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.

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.

RENESAS

Application Note

V850ES/Jx3-L

Sample Program (Interrupt)

External Interrupt Generated by Switch Input

This document summarizes the operations of the sample program and describes how to use the sample program and how to set and use the interrupt function. In the sample program, an interrupt is generated by detecting the falling edge of the switch input and an LED lighting pattern is displayed according to the number of switch inputs.

Target devices

V850ES/JF3-L microcontroller
V850ES/JG3-L microcontroller

CONTENTS

Document No. U19480EJ1V0AN00
Date Published December 2008 N

- **The information in this document is current as of October, 2008. The information is subject to change without notice. For actual design-in, refer to the latest publications of NEC Electronics data sheets or data books, etc., for the most up-to-date specifications of NEC Electronics products. Not all products and/or types are available in every country. Please check with an NEC Electronics sales representative for availability and additional information.**
- No part of this document may be copied or reproduced in any form or by any means without the prior written consent of NEC Electronics. NEC Electronics assumes no responsibility for any errors that may appear in this document.
- NEC 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 NEC Electronics products listed in this document or any other liability arising from the use of such products. No license, express, implied or otherwise, is granted under any patents, copyrights or other intellectual property rights of NEC Electronics or others.
- Descriptions of circuits, software and other related information in this document are provided for illustrative purposes in semiconductor product operation and application examples. The incorporation of these circuits, software and information in the design of a customer's equipment shall be done under the full responsibility of the customer. NEC Electronics assumes no responsibility for any losses incurred by customers or third parties arising from the use of these circuits, software and information.
- While NEC Electronics endeavors to enhance the quality, reliability and safety of NEC Electronics products, customers agree and acknowledge that the possibility of defects thereof cannot be eliminated entirely. To minimize risks of damage to property or injury (including death) to persons arising from defects in NEC Electronics products, customers must incorporate sufficient safety measures in their design, such as redundancy, fire-containment and anti-failure features.
- NEC Electronics products are classified into the following three quality grades: "Standard", "Special" and "Specific".

  The "Specific" quality grade applies only to NEC Electronics products developed based on a customer-designated "quality assurance program" for a specific application. The recommended applications of an NEC Electronics product depend on its quality grade, as indicated below. Customers must check the quality grade of each NEC Electronics product before using it in a particular application.

  "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.

  "Special": 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, life support systems and medical equipment for life support, etc.

The quality grade of NEC Electronics products is "Standard" unless otherwise expressly specified in NEC Electronics data sheets or data books, etc. If customers wish to use NEC Electronics products in applications not intended by NEC Electronics, they must contact an NEC Electronics sales representative in advance to determine NEC Electronics' willingness to support a given application.

(Note)

(1) "NEC Electronics" as used in this statement means NEC Electronics Corporation and also includes its majority-owned subsidiaries.

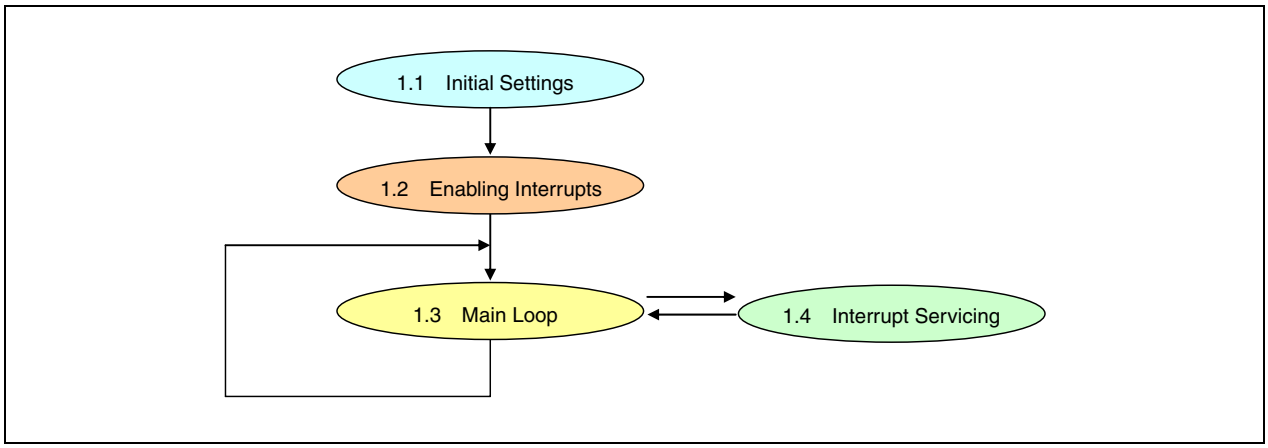
(2) "NEC Electronics products" means any product developed or manufactured by or for NEC Electronics (as defined above).

M8E 02. 11-1

CHAPTER 1 OVERVIEW

In this sample program, an example of using the interrupt function is presented. An LED lighting pattern is displayed according to the number of switch inputs, by detecting the falling edge of the switch input and servicing interrupts.

The main software contents are shown below.



1.1 Initial Settings

<Referencing option byte>

- Referencing the oscillation stabilization time after releasing reset

<Settings of on-chip peripheral functions>

- Setting wait operations <wait: 1> for bus access to on-chip peripheral I/O registers
- Setting on-chip debug mode <normal operation mode>
- Stopping the internal oscillator and watchdog timer
- Setting not to divide the CPU clock frequency
- Setting to PLL mode and setting to 20 MHz operation (5 MHz × 4).

<Pin settings>

- Setting unused pins
- Setting external interrupt pins (edge specification, priority specification, unmasking)
- Setting LED output pins (specifying values to turn off and output from LED1 and LED2)

<ROMization>

- ROMization processing (initialization of variables with initial values) (C language only)

1.2 Enabling Interrupts

- Enabling interrupts by using the EI instruction

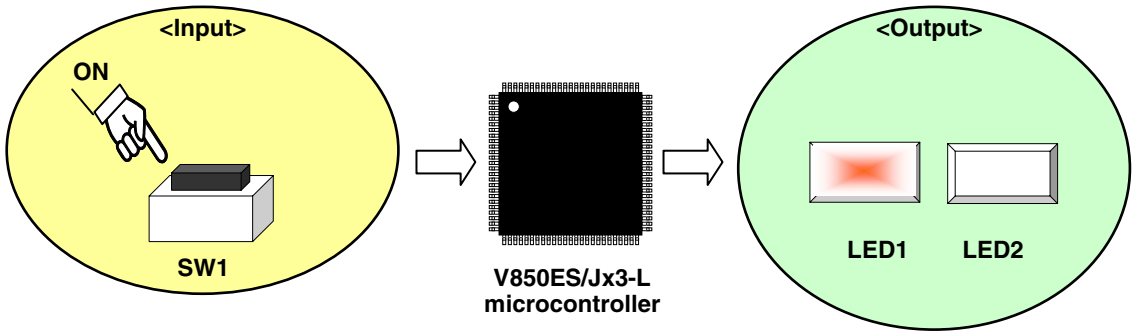
1.3 Main Loop

- Executing an infinite loop that executes no processing (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 LED lighting pattern 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 LED lighting pattern is not changed.



Switch input and a change the LED lighting pattern are repeated alternately.

Table 1-1. LED Lighting Patterns

Switch (SW1) Input count[Note]	LED1	LED2
0	OFF	OFF
1	ON	OFF
2	OFF	ON
3	ON	ON

Note Inputs 0 to 3 are repeated from the fourth input.

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

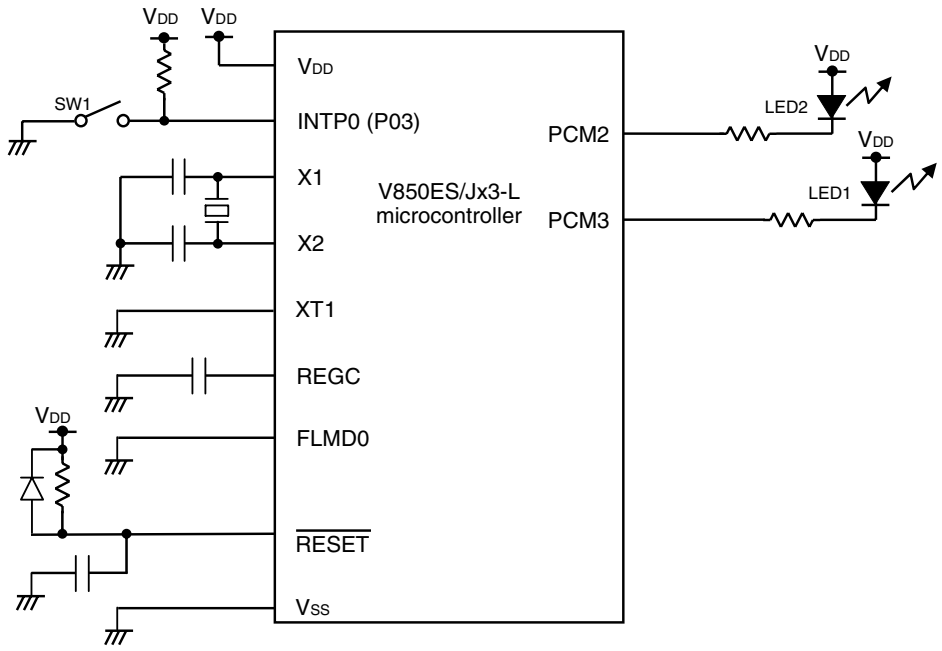
[Column] What is chattering?
Chattering is a phenomenon that an electric signal alternates between being on and off when a connection flip-flops mechanically immediately after a switch is switched.

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.



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 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 the LEDs.

(2) LEDs (LED1, LED2)

The LEDs are used as outputs corresponding to the number of switch inputs.

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.

[C language version]

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

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  : Only the source file is included.

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

[Assembler version]

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

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  : Only the source file is included.

 : 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.

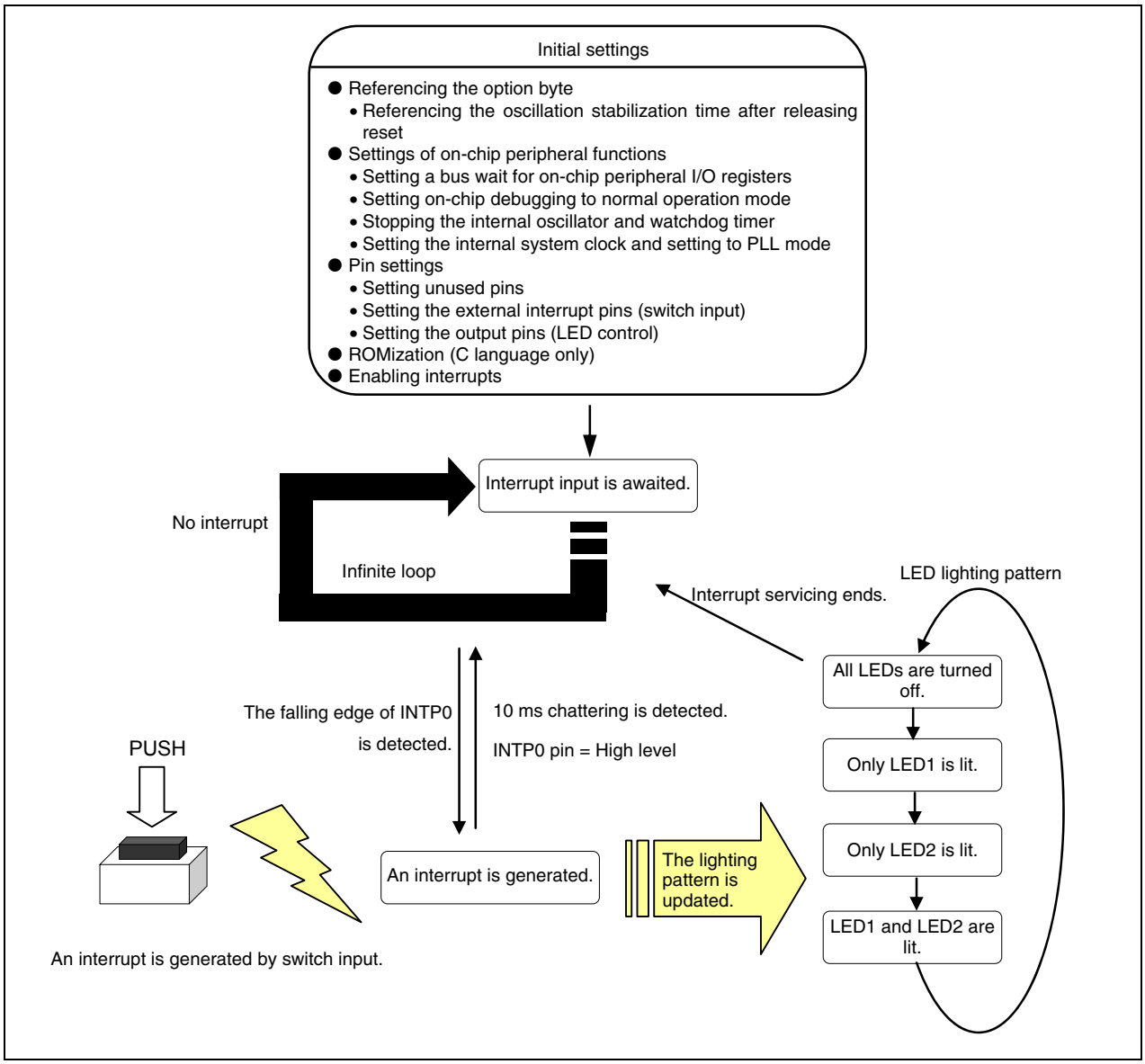
- External interrupt input (for switch input): INTP0 (SW1)
- Output ports (for lighting LEDs): PCM2 (LED2), PCM3 (LED1)

3.3 Initial Settings and Operation Overview

In this sample program, the selection of the clock frequency, setting for stopping the watchdog timer, setting of the I/O ports and external interrupt pins, and setting of interrupts are performed in the initial settings.

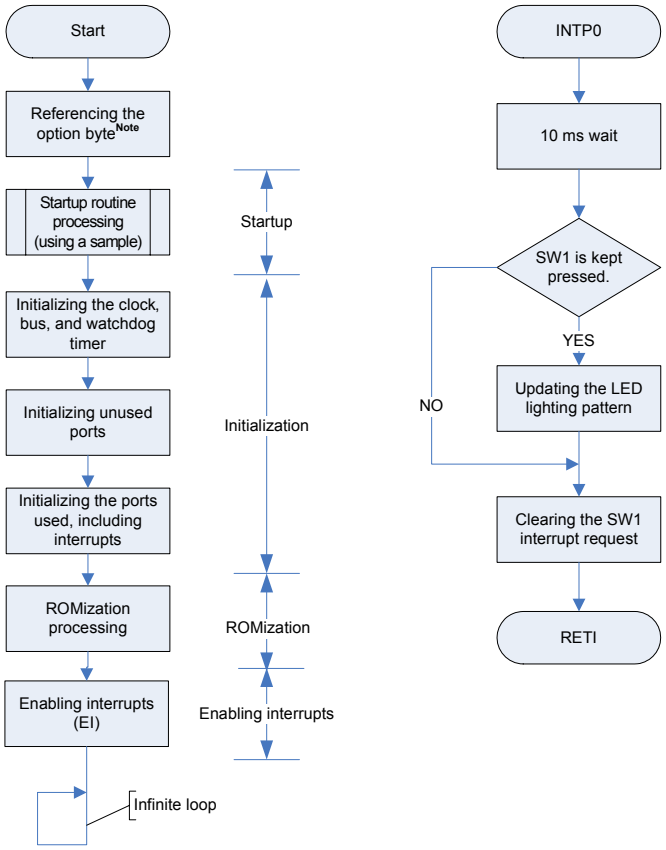
After completion of the initial settings, interrupts are serviced by detecting the falling edge of the switch input (SW1) and the lighting pattern of the two LEDs (LED1 and LED2) is controlled according to the number of switch inputs.

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



3.4 Flowchart

A flowchart for the sample program is shown below.



Note The option byte is automatically referenced by the microcontroller after reset is released. In this sample program, the oscillation stabilization time after releasing reset is set to 6.554 ms with the option byte.

[Column] Contents of the startup routine

The startup routine is a routine that is executed before executing the main function after the V850 is reset. Basically, the startup routine initializes the system after the system is reset.
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/JG3-L and V850ES/JF3-L

The V850ES/JG3-L is the V850ES/JF3-L with its functions, such as I/Os, timer/counters, and serial interfaces, expanded.

In this sample program, the initialization range of P7 in I/O initialization differs.

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

3.6 ROMization (C Language Only)

In this sample program (C language), ROMization information is copied after the on-chip peripheral functions are initialized.

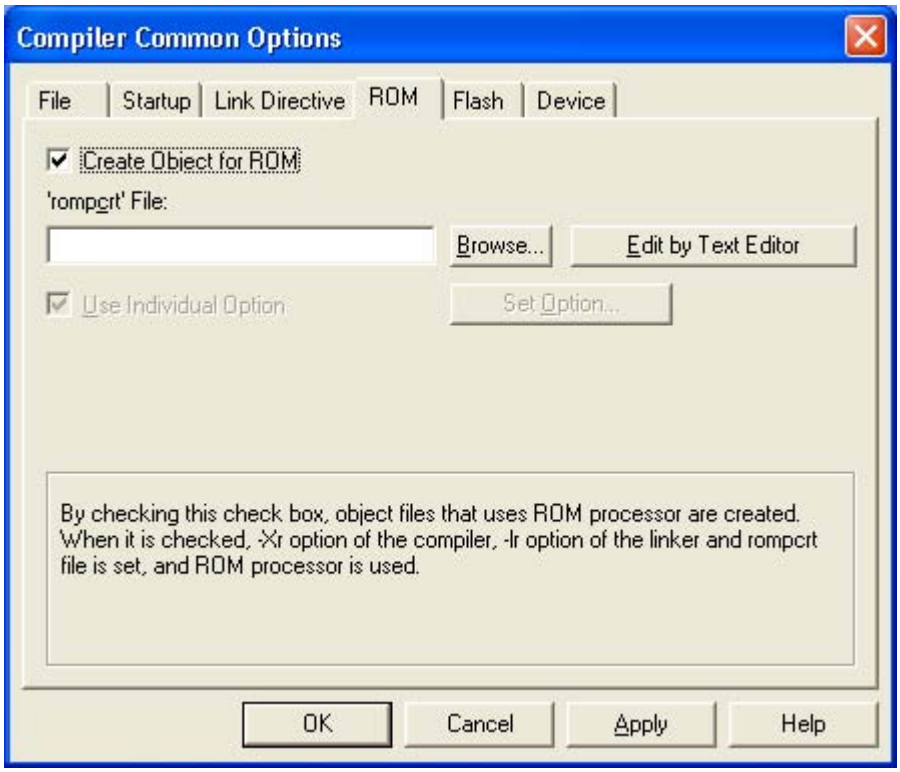
ROMization information is the information of the initial values of variables that have initial values (the section to which variables that have initial values are placed). Variables that have initial values (the section to which variables that have initial values are placed) will hold their software-based initial values for the first time by copying the ROMization information to the RAM[Note].

If a variable that has an initial value is used in the program to be created, ROMization information must be generated and copied. Furthermore, the ROMization information must be copied before using the variable that has an initial value.

Note The data allocated to a section that has a writable attribute is subject to packing by default in ROMization. Other data can also be packed. See the CA850 Help for details.

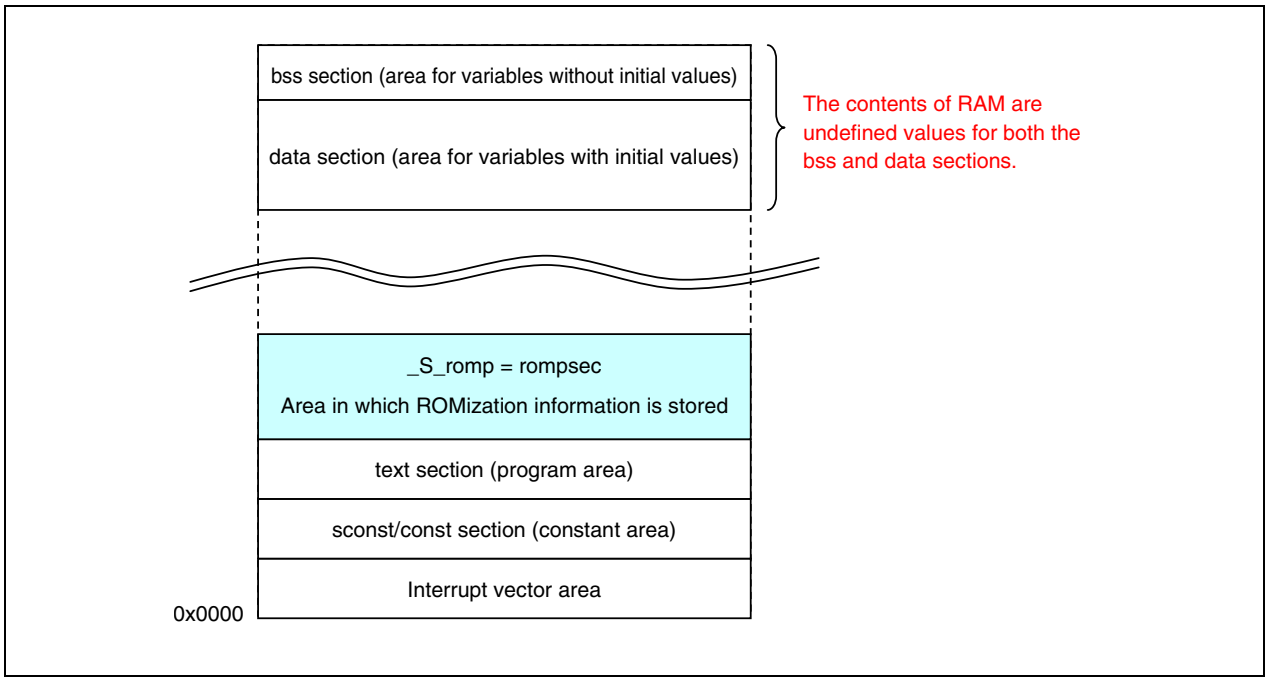
The ROMization procedure is described below.

Select the [ROM] tab, which is an option common to all PM+ compilers, and then check “Create Object for ROM”.



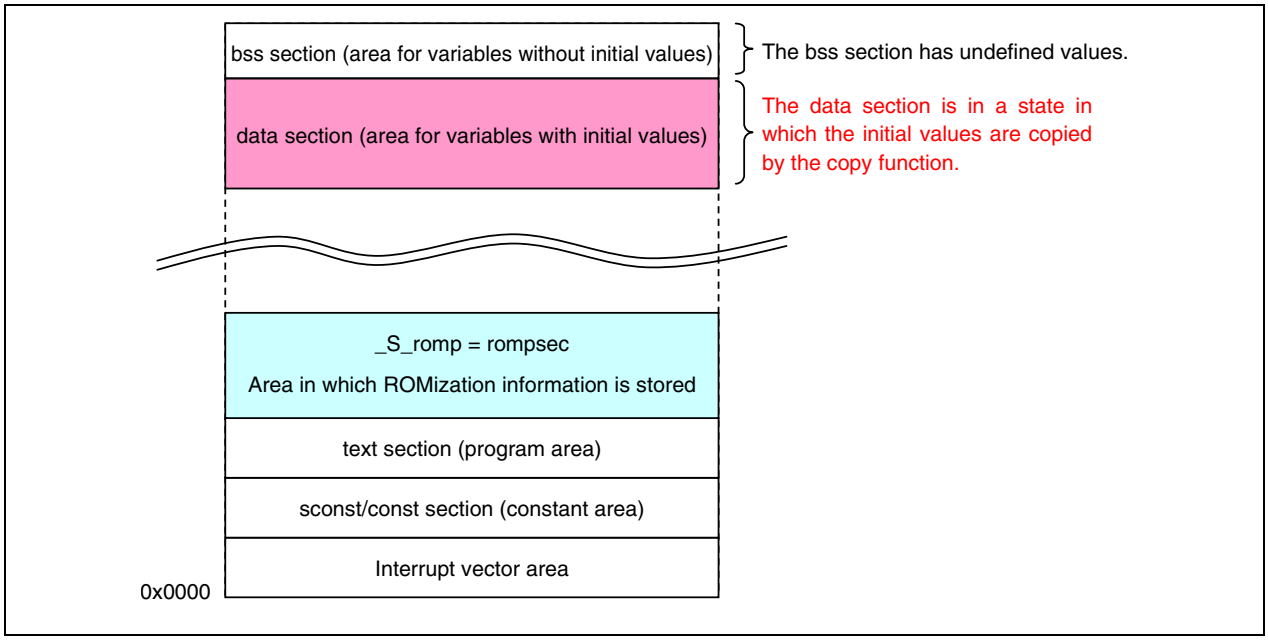
The section into which the ROMization information is to be stored (rompsec) will be automatically added immediately after the program area (.text) section. However, by checking “Create Object for ROM”, a code that indicates the same address as that of rompsec will be generated for the default label _S_romp defined by rompcrt.o, and the library libr.a, in which the copy function is stored, will be automatically linked.

An image of memory before the ROMization information is copied, which is created according to the procedure so far, is shown below.



The ROMization information must be copied, because the contents of the data section, which is the area for variables that have initial values will stay undefined if memory remains as is.

An image of the memory after the _rcopy() function is called to copy the ROMization information is shown below.



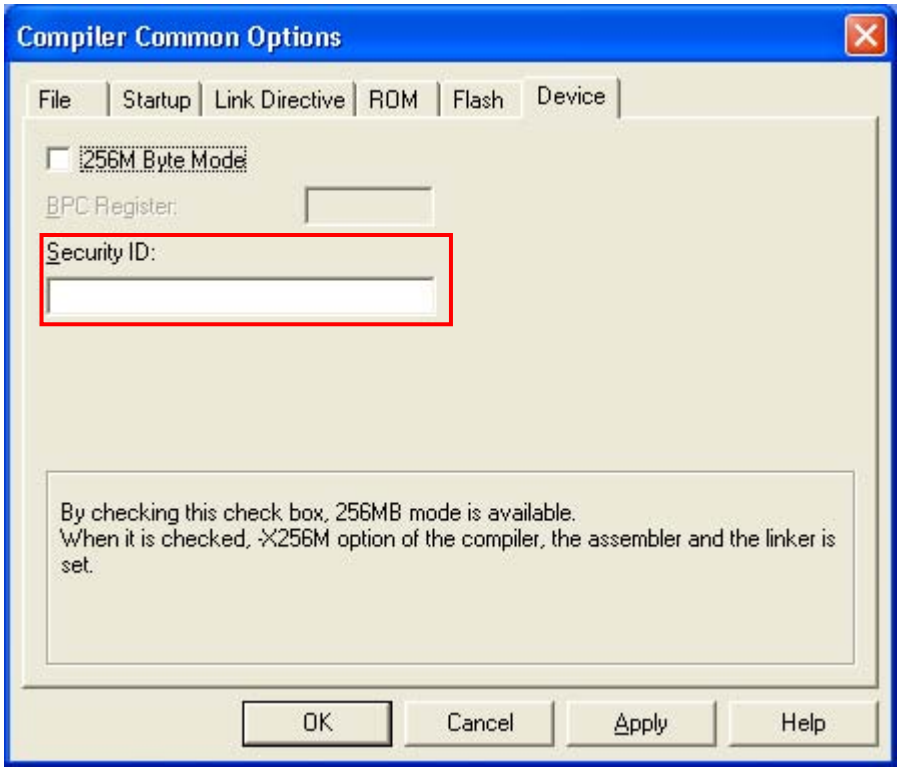
3.7 Security ID

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.

The debugger authorizes the ID by comparing it with the ID code preset to the 10 bytes from 0x0000070 to 0x0000079 in the internal flash memory area.

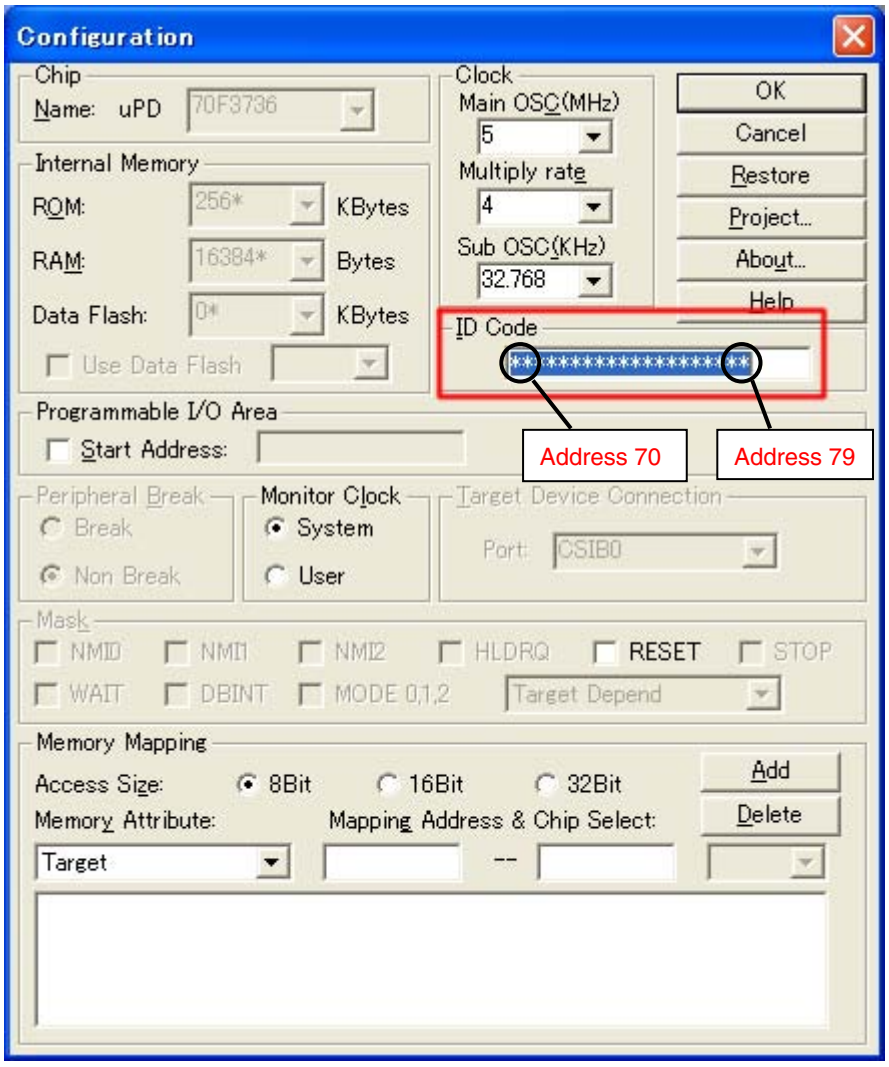
If the IDs match, the security code will be unlocked and reading flash memory and using the on-chip debug emulator will be enabled.

In this sample program (complete-environment version), the security ID is not set and the default security ID value 0xFFFF FFFF FFFF FFFF FFFF is applied.



Remark Set the security ID for a device provided with flash memory in the “Security ID” field, which is an option common to all compilers.
Specify the ID as a hexadecimal number of 10 bytes or less starting with 0x.
If specifying this option or specifying the security ID by using an assembly description (.section SECURITY_ID) is omitted, 0xFFFF FFFF FFFF FFFF FFFF will be assumed to have been specified.

If a program is downloaded and operated by using this sample program (complete-environment version), 0xFF will be set to the security ID area of the microcontroller. Caution is therefore required, because the on-chip debug emulator can be used only if 0xFFFF FFFF FFFF FFFF FFFF (default value) is set in the ID code entry area when the debugger is connected the next time.



- Bit 7 (0x0000079) of the 10 bytes of the ID code is the on-chip debug emulator use enable flag (0: Disables use, 1: Enables use).
- When the on-chip debug emulator is started, the debugger requests ID entry.
  The debugger will be started if the ID code entered in the debugger matches the ID code embedded in addresses 0x0000070 to 0x0000079.
- Even if the ID codes match, debugging cannot be executed if the on-chip debug emulator use enable flag is set to “0”.

CHAPTER 4 SETTING REGISTERS

This chapter describes how to set the interrupt pins and pins for lighting LED, as well as interrupt servicing.

For other initial settings, see the **V850ES/Jx3-L Sample Program (Initial Settings) LED Lighting Switch Control 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/JG3-L 32-bit Single-Chip Microcontroller
  Hardware User's Manual
- V850ES/JF3-L 32-bit Single-Chip Microcontroller
  Hardware User's Manual

See the following user's manuals for details of extended descriptions in C and assembly language.

- CA850 C Compiler Package C Language User's Manual
- CA850 C Compiler Package Assembly Language User's Manual

4.1 Setting Interrupt Pins and Pins for Lighting LEDs

In this sample program, the P03 pin is used as an external interrupt pin for SW1 input and interrupts are set to detect falling edges.

The PCM2 and PCM3 pins are set as output ports, because they are used as pins for lighting LEDs.

4.1.1 Setting port 0 mode register (PM0)

The PM0 register can be used to control the I/O mode of the P02 to P06 pins.

In this sample program, the I/O mode of the P03 pin is set to input mode, because the pin status is read by checking chattering.

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

Reset sets this register to FFH.

Figure 4-1. PM0 Register Format

Port 0 mode register (PM0)

Address: 0xFFFFF420

7	6	5	4	3	2	1	0
1	PM06	PM05	PM04	PM03	PM02	1	1

PM03	P03 pin I/O mode control
0	Output mode
1	Input mode

Remarks 1. The red values indicate the setting values used in the sample program.

2. If PM03 is set to output mode, the value of the output latch will be read by reading P03, so caution must be exercised.

The setting value of PM0 is 8BH.

• C language

```
PM0 = 0x8B;                    /* Connects P03 to the input latch. */
```

• Assembly language

```
SET_REG8    0x8B, PM0          -- Connects P03 to the input latch.
```

Remark The SET_REG8 macro is created by using the sample program.
It is a register access macro used to simplify the program descriptions for accessing 8-bit peripheral I/O registers.

```
--[8-bit peripheral I/O register access macro]
  .macro SET_REG8 val, reg
         mov     val, r6          -- Inputs a value to temporary r6 in advance and
         st.b    r6,(reg)[zero]   -- sets the value of r6 to a peripheral I/O register.
  .endm
```

4.1.2 Setting port 0 mode control register (PMC0)

The PMC0 register can be used to specify the operation mode of the P02 to P06 pins.

In this sample program, the operation mode of the P03 pin is set to INTP0 (external interrupt) input.

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

Reset sets this register to 00H.

Figure 4-2. PMC0 Register Format

Port 0 mode control register (PMC0)
Address:0xFFFFF440

7	6	5	4	3	2	1	0
0	PMC06	PMC05[Note]	PMC04	PMC03	PMC02	0	0

PMC03	P03 pin operation mode specification
0	I/O port
1	INTP0 input/ADTRG input

Note When the OCDM.OCDM0 bit is 1, the P05/INTP2/$\overline{\text{DRST}}$ pin is used as the $\overline{\text{DRST}}$ pin, regardless of the value of the PMC05 bit.

Remarks 1. The red values indicate the setting values used in the sample program.

2. The P02 to P06 pins have hysteresis characteristics if their alternate functions are set as inputs, but do not have hysteresis characteristics in port mode.

The setting value of PMC0 is 08H.

- C language

```
PMC0 = 0x08;                    /* Sets the input of the INTP0 pin. */
```

- Assembly language

```
SET_REG8    0x08, PMC0          -- Sets the input of the INTP0 pin.
```

4.1.3 Setting external interrupt falling and rising edge specification register 0 (INTF0, INTR0)

The INTF0 and INTR0 registers are 8-bit registers that are used to specify the detection of the NMI pin by using bit 2 and the detection of the rising and falling edges of external interrupt pins (INTP0 to INTP3) by using bits 3 to 6.

In this sample program, falling edges are set to be detected.

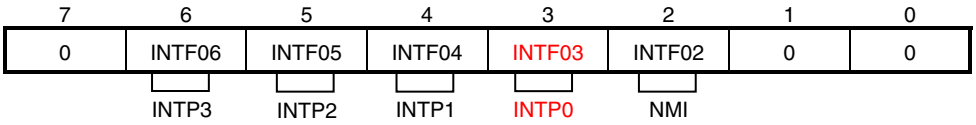
These registers can be read or written in 8-bit or 1-bit units.

Reset sets these registers to 00H.

Figure 4-3. INTF0 Register Format

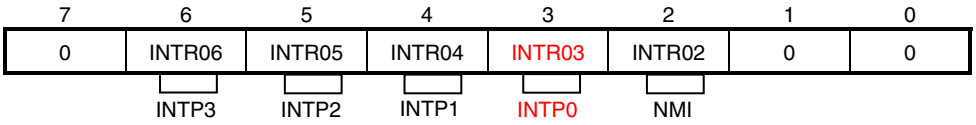
External interrupt falling edge specification register 0 (INTF0)

Address: 0xFFFFFC00



External interrupt rising edge specification register 0 (INTR0)

Address: 0xFFFFFC20



Remark See **Table 4-1** for details of valid-edge specification.

Table 4-1. Valid-Edge Specification

INTF03	INTR03	Valid-Edge (Falling) Specification
0	0	No edge detection
0	**1**	Detects the rising edge.
1	**0**	Detects the falling edge.
1	**1**	Detects both edges.

Remark The red values indicate the setting values used in the sample program.

The setting values of INTF0 and INTF0 are 08H and 00H, respectively.

- C language

```
INTF0 = 0x08;                /* Sets the input of the INTP0 pin. */
INTR0 = 0x00;
```

- Assembly language

```
SET_REG8    0x08, INTF0              -- Specifies the falling edge of INTP0.
SET_REG8    0x00, INTR0              --↓
```

4.1.4 Setting interrupt control register (PIC0)

The PIC0 register is assigned for each interrupt request signal (maskable interrupt) and is used to set the control conditions for each interrupt.

In this sample program, INTP0 is set to be used at the lowest priority level.

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

Reset sets this register to 47H.

Caution Read the xxICn.xxIFn bit while interrupts are disabled (DI) or masked. If the xxICn.xxIFn bit is read while interrupts are enabled (EI) or unmasked, a normal value may not be read if the timing of acknowledging interrupts and reading the bit conflict.

Figure 4-4. PIC0 Register Format

INTP0 interrupt control register (PIC0)

Address: 0xFFFFF112

7	6	5	4	3	2	1	0
PIF0	PMK0	0	0	0	PPR02	PPR01	PPR00

PIF0	Interrupt request flag
0	No interrupt request signal
1	An interrupt request signal is present

PMK0	Interrupt mask flag
0	Enables interrupt servicing
1	Disables (suspends) interrupt servicing

PPR02	PPR01	PPR00	Interrupt priority order specification bit
0	0	0	Specifies level 0 (highest level).
1	0	1	Specifies level 1.
0	1	0	Specifies level 2.
0	1	1	Specifies level 3.
1	0	0	Specifies level 4.
1	0	1	Specifies level 5.
1	1	0	Specifies level 6.
1	**1**	**1**	Specifies level 7 (lowest level).

Remark The red values indicate the setting values used in the sample program.

[Column] Interrupt request flag PIF0

Interrupt request flag PIF0 of the PIC0 register is set to 1 when an interrupt source is generated and automatically reset by hardware when an interrupt request signal is acknowledged.

The setting value of PIC0 is 07H.

• C language

```
PIC0 = 0x07;                    /* Sets the priority level of INTP0 to level 7 and unmasks
                                   INTP0. */
```

• Assembly language

```
SET_REG8    0x07, PIC0          -- Sets the priority level of INTP0 to level 7 and unmasks
                                   INTP0.
```

4.1.5 Setting port CM register (PCM) and port CM mode register (PMCM)

Port CM is a 4-bit port whose I/O can be controlled in 1-bit units.

In this sample program, the PCM2 and PCM3 pins are set to operate as output ports after the output to LED1 or LED2 is set not to light LED1 or LED2.

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

Reset sets the PCM register to 00H and the PMCM register to FFH.

Figure 4-5. PCM Register Format

Port CM register (PCM)

Address: 0xFFFFF00C

7	6	5	4	3	2	1	0
0	0	0	0	PCM3	PCM2	PCM1	PCM0

PCM2	PCM3	Output data control (in output mode)
0	0	Outputs 0.
1	**1**	Outputs 1.

Remark The red values indicate the setting values used in the sample program.

Figure 4-6. PMCM Register Format

Port CM mode register (PMCM)

Address: 0xFFFFF02C

7	6	5	4	3	2	1	0
1	1	1	1	PMCM3	PMCM2	PMCM1	PMCM0

PMCM2	PMCM3	I/O mode control
0	**0**	Output mode
1	1	Input mode

Remark The red values indicate the setting values used in the sample program.

[Column] Output port control

Generally, to initialize a port as an output port, set the output latch before setting the port mode (I/O) (set Pn, then PMn).

If the port mode is set first, the value set to the output latch at that time will be output from the pin and unintended pin output may be momentarily performed.

In this sample program, the output latches of PCM2 and PCM3 are preset to high-level output in the initial settings and then PCM2 and PCM3 are operated as output ports, in order to use PCM2 and PCM3 as output ports for lighting the LEDs.

The LEDs are lit when low level is output from PCM2 and PCM3 (see **2.1 Circuit Diagram**).

The setting values of PCM and PMCM are 0CH and F0H, respectively.

- C language

```
PCM = 0x0C;                    /* Sets values to turn off lighting to LED1 and LED2. */
PMCM = 0xF0;                   /* Sets the PCM pin to output.      */
```

- Assembly language

```
SET_REG8    0x0C, PCM          -- Sets values to turn off lighting to LED1 and LED2.
SET_REG8    0xF0, PMCM         -- Sets the PCM pin to output.
```

4.2 Interrupt Servicing

In this sample program, chattering is eliminated, the lighting pattern is updated, and interrupt requests are cleared in interrupt servicing.

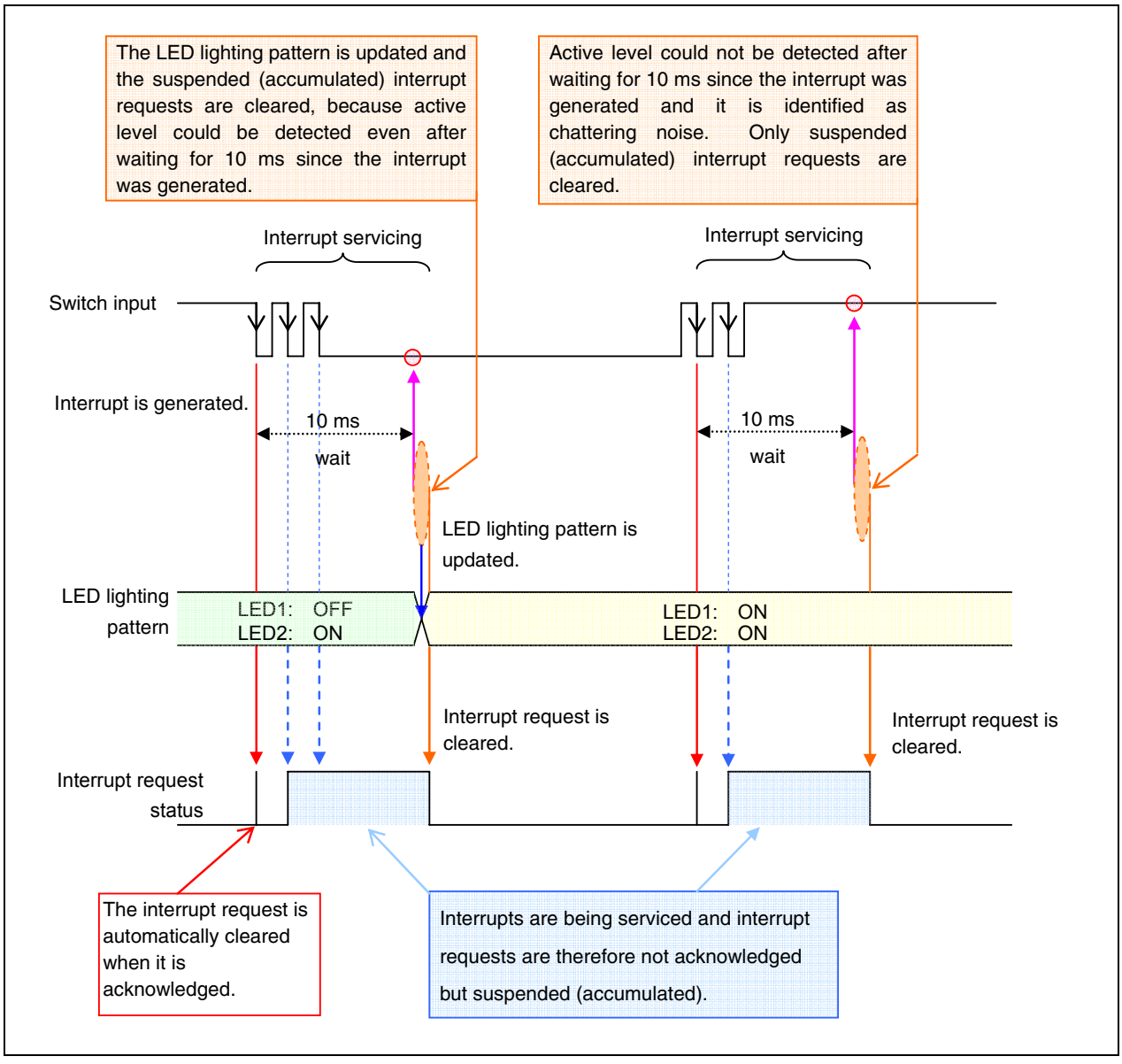
If an interrupt is generated, a 10 ms wait will be performed in interrupt servicing. If the switch input status is low level (active level) after the wait, switch input is assumed to be performed, and if the switch input status is high level (inactive level), it is identified as chattering noise and switch input is assumed not to be performed.

If switch input is performed, the current LED lighting pattern will be updated to the next LED lighting pattern.

See **Table 1-1 LED Lighting Patterns** for details of updating the LED lighting pattern.

After the LED lighting pattern is updated, the suspended (accumulated) interrupt requests that may have been generated due to chattering noise are cleared.

A timing chart of interrupt servicing based on switch input is shown below.



CHAPTER 5 RELATED DOCUMENTS

Document	English
V850ES/JF3-L User's Manual	PDF
V850ES/JG3-L User's Manual	PDF
PM+ Ver. 6.30 User's Manual	PDF
CA850 Ver. 3.20 C Compiler Package Operation	PDF
CA850 Ver. 3.20 C Compiler Package C Language	PDF
CA850 Ver. 3.20 C Compiler Package Assembly Language	PDF
CA850 Ver. 3.20 C Compiler Package Link Directive	PDF
V850ES Architecture	PDF

APPENDIX A PROGRAM LIST

The V850ES/Jx3-L microcontroller source program is shown below as a program list example.

- opt_b.s (common to the assembly language and C language versions)

```
#-------------------------------------------------------------------------------
#
#    NEC Electronics     V850ES/Jx3-L series
#
#-------------------------------------------------------------------------------
#    V850ES/JG3-L JF3-L  JF3-L sample program
#-------------------------------------------------------------------------------
#    Interrupts
#-------------------------------------------------------------------------------
#[History]
#    2008.7.--   Released
#-------------------------------------------------------------------------------
#[Overview]
#    This sample program sets the option byte.
#-------------------------------------------------------------------------------

     .section "OPTION_BYTES"
     .byte 0b00000101 -- 0x7a (5 MHz: Sets the oscillation stabilization time to 6.554 ms.)
     .byte 0b00000000 -- 0x7b            ↑
     .byte 0b00000000 -- 0x7c            ↑
     .byte 0b00000000 -- 0x7d 0x00 must be set to addresses 0x7b to 0x7f.
     .byte 0b00000000 -- 0x7e            ↓
     .byte 0b00000000 -- 0x7f            ↓
```

● minicube2.s (common to the assembly language and C language versions)

```
#-------------------------------------------------------------------------------
#
#    NEC Electronics     V850ES/Jx3-L series
#
#-------------------------------------------------------------------------------
#    V850ES/JG3-L JF3-L sample program
#-------------------------------------------------------------------------------
#    Interrupts
#-------------------------------------------------------------------------------
#[History]
#    2008.7.--   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_INT.dir (common to the assembly language and C language versions)

```
#   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         : uPD70F3738 (C:\Program Files\NEC Electronics Tools\DEV\DF3738.800)
#   Internal RAM   : 0x3ffb000 - 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 V0x03F800{
      MonitorROM = $PROGBITS ?A MonitorROM;
};
```

0x01F800 for products with 128 KB internal ROM

Difference from the default link directive file (additional code)

A reserved area for MINICUBE2 is secured.

```
SIDATA  : !LOAD ?RW V0x3ffb000 {
        .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 V0x3ffb100 {
        .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 (C language version)

```
/*---------------------------------------------------------------------------*/
/*
/*  NEC Electronics     V850ES/Jx3-L series
/*
/*---------------------------------------------------------------------------*/
/*  V850ES/JG3-L sample program
/*---------------------------------------------------------------------------*/
/*  Interrupts
/*---------------------------------------------------------------------------*/
/*[History]
/*  2008.07.--  Released
/*---------------------------------------------------------------------------*/
/*[Overview]
/*  This sample program presents an example of using the interrupt function.
/*  An LED lighting pattern that accords with the number of switch inputs is displayed
    by detecting the falling edge of the switch input and generating an interrupt.
/*  A chattering elimination time of 10 ms is provided for the switch inputs.
/*
/*  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
/*  • Performing prototype definitions
/*  • Defining the LED lighting pattern table
/*  • Setting a bus wait for on-chip peripheral I/O registers, stopping the watchdog
      timer, and setting the clock
/*  • Initializing unused ports
/*  • Initializing external interrupt ports (falling edge) and LED output ports
/*  • ROMization
/*  • Updating the LED lighting pattern in interrupt servicing
/*   (Chattering elimination time during switch input: 10 ms)
/*
```

```
/* <Switch input and LED lighting>
/*
/* +--------------–--------------–-----------------------+
/* |Number of times the switch is pressed  | LED2     | LED1     |
/* |      (P03/INTP0)                      |  (PCM2)  |  (PCM3)  |
/* |-------------------------------------------------------|
/* |        0 times                        |  OFF     |  OFF     |
/* |        1 time                         |  OFF     |  ON      |
/* |        2 times                        |  ON      |  OFF     |
/* |        3 times                        |  ON      |  ON      |
/* +--------------–--------------–-----------------------+
/*      *Inputs 0 to 3 are repeated from the fourth input.
/*
/*
/*[I/O port settings]
/*
/*  • Input port   : P03(INTP0)
/*  • Output ports : PCM2, PCM3
/*  • Unused ports : P02, P04 to P06, P10 and P11, P30 to P39, P50 to P55, P70 to P711,
/*                   P90 to P915, PCM0 and PCM1, PCT0, PCT1, PCT4, PCT6, PDH0 to PDH5,
/*                   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 (0x6EE7)       /* Defines the constant for a 10 ms wait
                                          adjustment.                          */
```

```
/*---------------------------*/
/*  Prototype definitions  */
/*---------------------------*/
        void main( void );                  /* Main function                          */
static    void f_init( void );              /* Initialization function                */
static    void f_init_clk_bus_wdt2( void ); /* Clock bus WDT initialization function */
static    void f_init_port_func( void );    /* Port/alternate-function initialization
                                               function    */


/*---------------------------------------*/
/* Setting the LED pattern table */
/*---------------------------------------*/
const unsigned char LED_TBL[] ={ 0x0C,      /* LED display pattern 0 [OFF][OFF]       */
                                 0x04,      /* LED display pattern 1 [OFF][ON]        */
                                  0x08,     /* LED display pattern 2 [ON] [OFF]       */
                                  0x00 };   /* LED display pattern 3 [ON] [ON]        */


/*****************************/
/*        Main module       */
/*****************************/
void main(void)
{
    extern unsigned int _S_romp;            /* Externally references ROMization
                                               symbols.                               */

    f_init();                               /* Executes initialization.               */

    _rcopy( &_S_romp, -1 );                 /* Executes ROMization processing.        */

    __EI();                                 /* Enables interrupts.                    */

    while(1);                               /* Main loop (infinite loop)              */

    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. */

    return;
}

/*--------------------------------------*/
/* Initializing clock bus WDT2 */
/*--------------------------------------*/
static void f_init_clk_bus_wdt2( void )
{
    VSWC  = 0x01;              /* Sets a bus wait for on-chip peripheral I/O
                                  registers. */

                               /* Specifies normal operation mode for OCDM. */

#pragma asm
    push r10
    mov 0x00, r10
    st.b r10, PRCMD
    st.b r10, OCDM
    pop r10
#pragma endasm

    RCM    = 0x01;             /* Stops the internal oscillator. */
    WDTM2  = 0x00;             /* Stops the watchdog timer.      */
                               /* Sets not to divide the clock.  */

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

    PLLCTL = 0x03;             /* Sets to PLL mode.              */

    return;
}
```

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

```
/*-----------------------------------------*/
/*   Setting the port/alternate function   */
/*-----------------------------------------*/
static void f_init_port_func( void )
{
    P0    = 0x00;                /* Sets P02 to P06 to output low level.          */
    PM0   = 0x8B;
    PMC0  = 0x08;                /* Sets INTP0 input to P03.                      */

    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;

#if(0) /* To use P4 as CSIB0 when using MINICUBE2,                                */
       /* P4 is not initialized as an unused pin (QB-V850ESJG3L-TB) */
    P4    = 0x00;                /* Sets P40 to P42 to output low level.          */
    PM4   = 0xF8;
    PMC4  = 0x00;
#endif

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

    P7H   = 0x00;                /* Sets P70 to P711 to output low level.         */
    P7L   = 0x00;
    PM7H  = 0xF0;
    PM7L  = 0x00;

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

    PCM   = 0x0C;                /* Sets PCM0 and PCM1 to output low level and values
                                    to turn off the LEDs to PCM2 and PCM3. */
    PMCM  = 0xF0;
    PMCCM = 0x00;

    PCT   = 0x00;                /* Sets PCT0, 1, 4, and 6 to output low level.   */
    PMCT  = 0xAC;
    PMCCT = 0x00;
```

With V850ES/JF3-L, the setting value is 0xFE.

With V850ES/JF3-L, only P10 is set.

With V850ES/JF3-L, the setting value is 0xFCC0.

With V850ES/JF3-L, P30 to P35, P38, and P39 are set.

With V850ES/JF3-L, these are not set because the registers do not exist.

With V850ES/JF3-L, P70 to P77 are set.

With V850ES/JF3-L, the setting value is 0x1C3C.

With V850ES/JF3-L, P90, P91, P96 to P99, and P913 to P915 are set.

```
    PDH    = 0x00;                  /* Sets PDH0 to PDH5 to output low level.        */
    PMDH   = 0xC0;
    PMCDH  = 0x00;

    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;
}
```

With V850ES/JF3-L, the setting value is 0xFC.

With V850ES/JF3-L, PDH0 and PDH1 are set.

```
/*****************************/
/*      Interrupt module     */
/*****************************/
__interrupt
void f_int_intp0( void )
{
    static unsigned int led_ptn_cnt = 0;
    unsigned int loop_wait;

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

    if( ( P0 & 0x08 ) == 0x00 )     /* Recognizes SW1 even after chattering is
                                       eliminated.*/
    {
        led_ptn_cnt++;              /* Changes the lighting pattern (4 types).       */
        if( led_ptn_cnt >= 4 )
        {
            led_ptn_cnt = 0;
        }

        PCM = LED_TBL[led_ptn_cnt]; /* Sets the updated lighting pattern.            */
    }

    PIC0 &= (unsigned char)~0x80;   /* FailSafe: Clears multiple requests.           */

    return;                         /* Goes to reti due to the __interrupt modifier. */
}
```

• main.s (Assembly language version)

```
#-------------------------------------------------------------------------------
#
#  NEC Electronics     V850ES/Jx3-L series
#
#-------------------------------------------------------------------------------
#  V850ES/JG3-L sample program
#-------------------------------------------------------------------------------
#  Interrupts
#-------------------------------------------------------------------------------
#[History]
#  2008.07.--  Released
#-------------------------------------------------------------------------------
#[Overview]
#  This sample program presents an example of using the interrupt function.
#  An LED lighting pattern that accords with the number of switch inputs is displayed
#  by detecting the falling edge of the switch input and generating an interrupt.
#  A chattering elimination time of 10 ms is provided for the switch inputs.
#
#  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>
#  • Setting interrupt handler processing
#  • Defining a wait adjustment value of 10 ms for chattering
#  • Defining the register access macro and module jump macro (to improve the
#    readability, maintainability, and portability of the program)
#  • Defining the LED lighting pattern table
#  • Setting a bus wait for peripheral I/O registers, stopping the watchdog timer, and
#    setting the clock
#  • Initializing unused ports
#  • Initializing external interrupt ports (falling edge) and LED output ports
#  • ROMization
#  • Updating the LED lighting pattern in interrupt servicing
#   (Chattering elimination time during switch input: 10 ms)
#
```

```
# <Switch input and LED lighting>
#
# +---------------—---------------—------------------------+
# |Number of times the switch is pressed   | LED2     | LED1     |
# |      (P03/INTP0)                       |  (PCM2)  |  (PCM3)  |
# |--------------------------------------------------------|
# |        0 times                         |  OFF     |  OFF     |
#*|        1 time                          |  OFF     |  ON      |
# |        2 times                         |  ON      |  OFF     |
# |        3 times                         |  OFF     |  ON      |
# +---------------—---------------—------------------------+
# *Inputs 0 to 3 are repeated from the fourth input.
#
#
#[I/O port settings]
#
#  • Input port     : P03(INTP0)
#  • Output ports   : P02 to P06, P10 and P11, P30 to P39, P50 to P55, P70 to P711,
                      P90 to P915, PCM0 and PCM1, PCT0, PCT1, PCT4, PCT6, PDH0 to PDH5,
#                     PDL0 to PDL15
#                     *Preset all unused ports as output ports.
#
#-------------------------------------------------------------------------------*/


#-----------------------------------------#
# Setting interrupt handler processing #
#-----------------------------------------#
    .section    "INTP0", text
    jr        F_INT_UPDATE_LED_PATTERN


#-----------------------------------------#
# Defining a wait adjustment value of 10 ms #
#-----------------------------------------#
    .set        LIMIT_10ms_WAIT, 0x6EE7
```

```
#-----------------------------------------#
# Macro for setting peripheral I/O registers #
#-----------------------------------------#

--[8-bit peripheral I/O register access macro]
    .macro    SET_REG8 val, reg
        mov    val, r6           -- Inputs a value to temporary r6 in advance and
        st.b    r6,(reg)[zero]   -- sets the value of r6 to a peripheral I/O register.
    .endm

--[16-bit peripheral I/O register access macro]
    .macro    SET_REG16 val, reg
        mov    val, r6           -- Inputs a value to temporary r6 in advance and
        st.h    r6,(reg)[zero]   -- sets the value of r6 to a peripheral I/O register.
    .endm

--[Special register access macro]
    .macro    SET_SP_REG val, reg
        mov        val, r6
        st.b    r6, (PRCMD)[zero]    -- Presets r6 to a command register.
        st.b    r6, (reg)[zero]      -- Sets the same value to a special register.
    .endm


#----------------------------------#
# Module jump macro #
#----------------------------------#

--[Module jump macro]
    .macro    CALL label
        push    lp              -- Saves the value of the lp register.
        jarl    label, lp       -- Stores the return destination PC to lp and
                                   transitions to label.
        pop    lp               -- Restores the lp register value that was saved.
    .endm

--[Return macro]
    .macro    RET
        jmp    [lp]             -- Jumps to the return destination PC.
    .endm
```

```
#--------------------------------------#
# Setting the LED pattern table #
#--------------------------------------#
    .const

T_LED_PATTERN:
    .byte    0x0C              -- LED display pattern 0 [OFF][OFF]
    .byte    0x04              -- LED display pattern 1 [OFF][ON]
    .byte    0x08              -- LED display pattern 2 [ON] [OFF]
    .byte    0x00              -- LED display pattern 3 [ON] [ON]


#****************************#
#       Main module        #
#****************************#
    .text
    .align 4
    .globl    _main

_main:

    mov    0x00, r7       -- Initializes the value of the LED lighting pattern control
                             register.

    CALL    F_INIT        -- Executes initialization.
                          -- Sets a bus wait for on-chip peripheral I/O
                             registers, stops WDT, and sets the clock.
                          -- Sets the port/alternate function.

    ei                    -- Enables interrupts.

  L_MAIN_LOOP:            -- ↑
    nop                   -- Infinite loop that executes no processing (waits for an
                             interrupt generated by switch input)
    jr    L_MAIN_LOOP     -- ↓
```

```
#---------------------------------#
#  Initialization module    #
#---------------------------------#
F_INIT:

    CALL    F_INIT_CLK_BUS_WDT2 -- Sets a bus wait for on-chip peripheral I/O registers,
                                   stops WDT, and sets the clock.

    CALL    F_INIT_PORT_FUNC    -- Sets the port/alternate function.

    RET

#-------------------------------------#
# Initializing clock bus WDT2 #
#-------------------------------------#
F_INIT_CLK_BUS_WDT2:

    SET_REG8    0x01, VSWC    -- Sets a bus wait for on-chip peripheral I/O registers.

    SET_SP_REG    0x00, OCDM  -- Specifies normal operation mode for OCDM.

    SET_REG8    0x01, RCM     -- Stops the internal oscillator.
    SET_REG8    0x00, WDTM2   -- Stops WDT2.

    SET_SP_REG    0x80, PCC   -- Sets not to divide the clock.

    SET_REG8    0x03, PLLCTL  -- Sets to PLL mode.

    RET
```

```
#--------------------------------------#
#  Setting the port/alternate function #
#--------------------------------------#
F_INIT_PORT_FUNC:

    SET_REG8    0x00, P0          -- Sets P02 to P06 to output low level.
    SET_REG8    0x8B, PM0         --
    SET_REG8    0x08, PMC0        -- Sets INTP0 to be input.

    SET_REG8    0x00, P1          -- Sets P10 and P11 to output low level.
    SET_REG8    0xFC, PM1         -- ↓

    SET_REG16    0x0000, P3       -- ↑
    SET_REG16    0xFC00, PM3      -- Sets P30 to P39 to output low level
    SET_REG16    0x0000, PMC3     -- ↓
-- P4 is not initialized as an unused pin, because P4 is used as CSIB0 when using
   MINICUBE2.
-- (QB-V850ESJG3L-TB)
--    SET_REG8    0x00, P4        -- ↑
--    SET_REG8    0xF8, PM4       -- Sets P40 to P42 to output low level.
--    SET_REG8    0x00, PMC4      -- ↓

    SET_REG8    0x00, P5          -- ↑
    SET_REG8    0xC0, PM5         -- Sets P50 to P55 to output low level.
    SET_REG8    0x00, PMC5        -- ↓

    SET_REG8    0x00, P7H         -- ↑
    SET_REG8    0x00, P7L         -- Sets P70 to P711 to output low level.
    SET_REG8    0xF0, PM7H        -- ↓
    SET_REG8    0x00, PM7L        -- ↓

    SET_REG16    0x0000, P9       -- ↑
    SET_REG16    0x0000, PM9      -- Sets P90 to P915 to output low level.
    SET_REG16    0x0000, PMC9     -- ↓

    SET_REG8    0x0C, PCM         -- Sets PCM0 and PCM1 to output low level and values
                                     to turn off the LEDs to PCM2 and PCM3.
    SET_REG8    0xF0, PMCM        -- Sets the PCM pin as an output port.
    SET_REG8    0x00, PMCCM       -- ↓

    SET_REG8    0x00, PCT         -- ↑
    SET_REG8    0xAC, PMCT        -- Sets PCT0, 1, 4, and 6 to output low level.
    SET_REG8    0x00, PMCCT       -- ↓
```

With V850ES/JF3-L, only P10 is set.

With V850ES/JF3-L, the setting value is 0xFE.

With V850ES/JF3-L, P30 to P35, P38, and P39 are set.

With V850ES/JF3-L, the setting value is 0xFCC0.

With V850ES/JF3-L, these are not set because the registers do not exist.

With V850ES/JF3-L, P70 to P77 are set.

With V850ES/JF3-L, P90, P91, P96 to P99, and P913 to P915 are set.

With V850ES/JF3-L, the setting value is 0x1C3C.

With V850ES/JF3-L, PDH0 and PDH1 are set.

With V850ES/JF3-L, the setting value is 0xFC.

```
    SET_REG8     0x00, PDH          -- ↑
    SET_REG8     0xC0, PMDH         -- Sets PDH0 to PDH5 to output low level.
    SET_REG8     0x00, PMCDH        -- ↓

    SET_REG16     0x0000, PDL       -- ↑
    SET_REG16     0x0000, PMDL      -- Sets PDL0 to PDL15 to output low level.
    SET_REG16     0x0000, PMCDL     -- ↓

-- Setting the interrupt function
    SET_REG8     0x08, INTF0        -- Specifies the falling edge of INTP0.
    SET_REG8     0x00, INTR0        -- ↓
    SET_REG8     0x07, PIC0         -- Sets the priority of INTP0 to level 7 and
                                       unmasks INTP0.

    RET
```

```
#*****************************#
#       Interrupt module      #
#*****************************#
F_INT_UPDATE_LED_PATTERN:

    mov        LIMIT_10ms_WAIT, r6      -- Sets the initial value of the chattering
time.

L_KEY_CHAT:
    sub        1, r6                    -- Loops without executing any processing
    nop                                 -- until 10 ms elapse to eliminate chattering.
    jnz        L_KEY_CHAT               -- ↓

    ld.b        (P0)[zero],r6           -- Checks whether SW1 is kept pressed
    and        0x08, r6                 -- even after 10 ms elapse.
    jnz        J_INT_P03_END            -- ↓

    add        1, r7                    -- Updates the LED output pattern to the
    and        3, r7                    -- following pattern among the four types.

    ld.b        (!T_LED_PATTERN)[r7], r6
    st.b        r6, (PCM)[zero]         -- Sets the LED lighting pattern.

J_INT_P03_END:

    clr1        7, (PIC0)[zero]     -- Clears the interrupt requests that may have been
                                       received multiple times by chattering.

    reti
```

For further information,
please contact:

NEC Electronics Corporation
1753, Shimonumabe, Nakahara-ku,
Kawasaki, Kanagawa 211-8668,
Japan
Tel: 044-435-5111
http://www.necel.com/

[America]

NEC Electronics America, Inc.
2880 Scott Blvd.
Santa Clara, CA 95050-2554, U.S.A.
Tel: 408-588-6000
800-366-9782
http://www.am.necel.com/

[Europe]

NEC Electronics (Europe) GmbH
Arcadiastrasse 10
40472 Düsseldorf, Germany
Tel: 0211-65030
http://www.eu.necel.com/

Hanover Office
Podbielskistrasse 166 B
30177 Hannover
Tel: 0 511 33 40 2-0

Munich Office
Werner-Eckert-Strasse 9
81829 München
Tel: 0 89 92 10 03-0

Stuttgart Office
Industriestrasse 3
70565 Stuttgart
Tel: 0 711 99 01 0-0

United Kingdom Branch
Cygnus House, Sunrise Parkway
Linford Wood, Milton Keynes
MK14 6NP, U.K.
Tel: 01908-691-133

Succursale Française
9, rue Paul Dautier, B.P. 52
78142 Velizy-Villacoublay Cédex
France
Tel: 01-3067-5800

Sucursal en España
Juan Esplandiu, 15
28007 Madrid, Spain
Tel: 091-504-2787

Tyskland Filial
Täby Centrum
Entrance S (7th floor)
18322 Täby, Sweden
Tel: 08 638 72 00

Filiale Italiana
Via Fabio Filzi, 25/A
20124 Milano, Italy
Tel: 02-667541

Branch The Netherlands
Steijgerweg 6
5616 HS Eindhoven
The Netherlands
Tel: 040 265 40 10

[Asia & Oceania]

NEC Electronics (China) Co., Ltd
7th Floor, Quantum Plaza, No. 27 ZhiChunLu Haidian
District, Beijing 100083, P.R.China
Tel: 010-8235-1155
http://www.cn.necel.com/

Shanghai Branch
Room 2509-2510, Bank of China Tower,
200 Yincheng Road Central,
Pudong New Area, Shanghai, P.R.China P.C:200120
Tel:021-5888-5400
http://www.cn.necel.com/

Shenzhen Branch
Unit 01, 39/F, Excellence Times Square Building,
No. 4068 Yi Tian Road, Futian District, Shenzhen,
P.R.China P.C:518048
Tel:0755-8282-9800
http://www.cn.necel.com/

NEC Electronics Hong Kong Ltd.
Unit 1601-1613, 16/F., Tower 2, Grand Century Place,
193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: 2886-9318
http://www.hk.necel.com/

NEC Electronics Taiwan Ltd.
7F, No. 363 Fu Shing North Road
Taipei, Taiwan, R. O. C.
Tel: 02-8175-9600
http://www.tw.necel.com/

NEC Electronics Singapore Pte. Ltd.
238A Thomson Road,
#12-08 Novena Square,
Singapore 307684
Tel: 6253-8311
http://www.sg.necel.com/

NEC Electronics Korea Ltd.
11F., Samik Lavied'or Bldg., 720-2,
Yeoksam-Dong, Kangnam-Ku,
Seoul, 135-080, Korea
Tel: 02-558-3737
http://www.kr.necel.com/

G0706