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

Sample Program (Low-Voltage Detector (LVI))

Reset Generation When Low Voltage Detected

This document summarizes the operations of the sample program and describes how to use the sample program and how to set and use the low-voltage detector (LVI). In the sample program, an internal reset (LVI reset) signal is generated by detecting that V_{DD} is less than V_{LVI} (V_{LVI} = 2.95 V (typ.) ± 0.10 V). With an LVI reset, the LED lighting pattern immediately before the LVI reset is retained and then restored after LVI reset release.

Target devices

V850ES/JG3 microcontroller

V850ES/JJ3 microcontroller

CONTENTS

CHAPTER 1 OVERVIEW 3
1.1 Initial Settings 4
1.2 Operation of Low-Voltage Detector (LVI) 5
1.3 Main processing 5
1.4 Interrupt Servicing 6
CHAPTER 2 CIRCUIT DIAGRAM 7
2.1 Circuit Diagram 7
2.2 Peripheral Hardware 7
CHAPTER 3 SOFTWARE 8
3.1 File Configuration 8
3.2 On-Chip Peripheral Functions Used 9
3.3 Initial Settings and Operation Overview 9
3.4 Flowchart 11
3.5 Differences Between V850ES/JJ3 and V850ES/JG3 14
3.6 ROMization 14
3.7 Security ID 16
3.8 Notes on Operating Sample Program When Using MINICUBE2 16
CHAPTER 4 SETTING REGISTERS 17
4.1 Setting Low-Voltage Detector (LVI) 18
4.1.1 Settings regarding low-voltage detection 18
4.1.2 Setting regarding low-voltage detection level 19
4.1.3 Controls regarding RAM retention voltage detection 19
4.2 Checking Detection of LVI Reset 22
4.2.1 Reset source flag register (RESF) 22
CHAPTER 5 RELATED DOCUMENTS 24
APPENDIX A PROGRAM LIST 25

Document No. U20019EJ1V0AN00 (1st edition)
Date Published September 2009 NS

- The information in this document is current as of September, 2009. 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.

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

(M8E0909)

CHAPTER 1 OVERVIEW

In this sample program, an example of using the low-voltage detector (LVI) is presented.

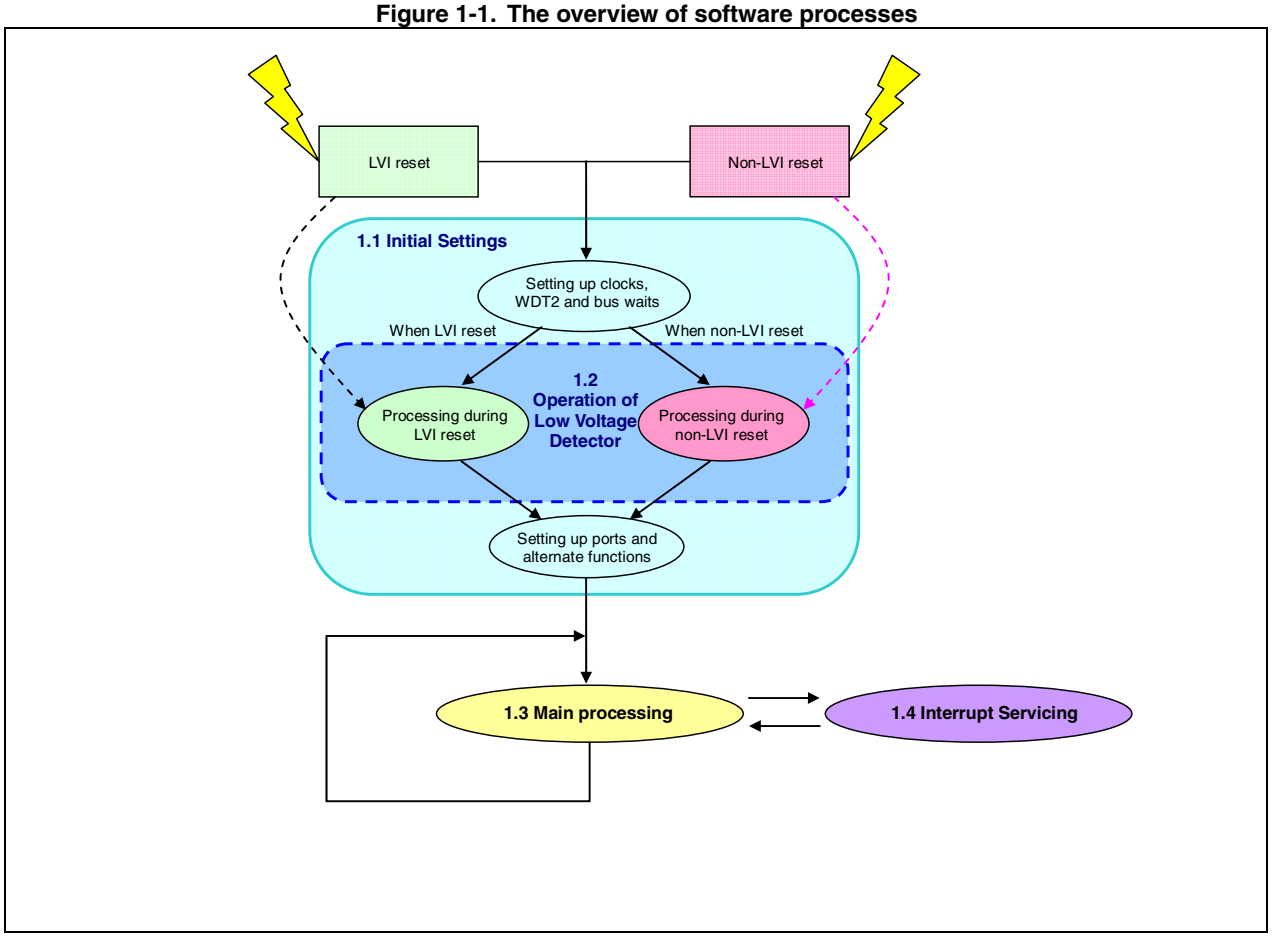
The low-voltage detector is used to detect that V_{DD} is less than V_{LVI} (V_{LVI} = 2.95 V (typ.)±0.10 V), and it is set to generate an internal reset (LVI reset) signal.

After completion of the initial setup, 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. (This processing is the same as that described in the **V850ES/Jx3 Sample Program (Interrupt) External Interrupt Generated by Switch Input Application Note**.)

When a non-LVI reset is generated, the program is used to initialize the LED lighting pattern data held in the internal RAM. When an LVI reset is generated, the LED lighting pattern data immediately before reset generation is retained and then restored and displayed after the LVI reset is released. In a precise sense, it should be checked whether the internal RAM is valid or not by using LVI. In this sample program, that check is executed by RAM retention voltage detection operation.

The main software processes are shown below.

Figure 1-1. The overview of software processes



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 normal operation mode
- Stopping the internal oscillator and watchdog timer 2
- Setting not to divide the CPU clock frequency
- Setting to PLL mode and setting to 32 MHz operation (4 MHz × 8)
- Setting V_{LVI} (low-voltage detection voltage) to 2.95 V (typ.) ±0.10 V and setting <reset> operation when low-voltage is detected[Note]

Note No setting is made when the system is restarted by an LVI reset.

<Pin settings>

- Setting unused pins
- Setting external interrupt pins (edge specification, priority specification, unmasking)
- Setting LED output pins (specifying the value at which LED1 and LED2 are turned off when a non-LVI reset occurs and specifying the output value prior to reset when an LVI reset occurs)

<ROMization>

- ROMization processing (initialization of variables with initial values)

1.2 Operation of Low-Voltage Detector (LVI)

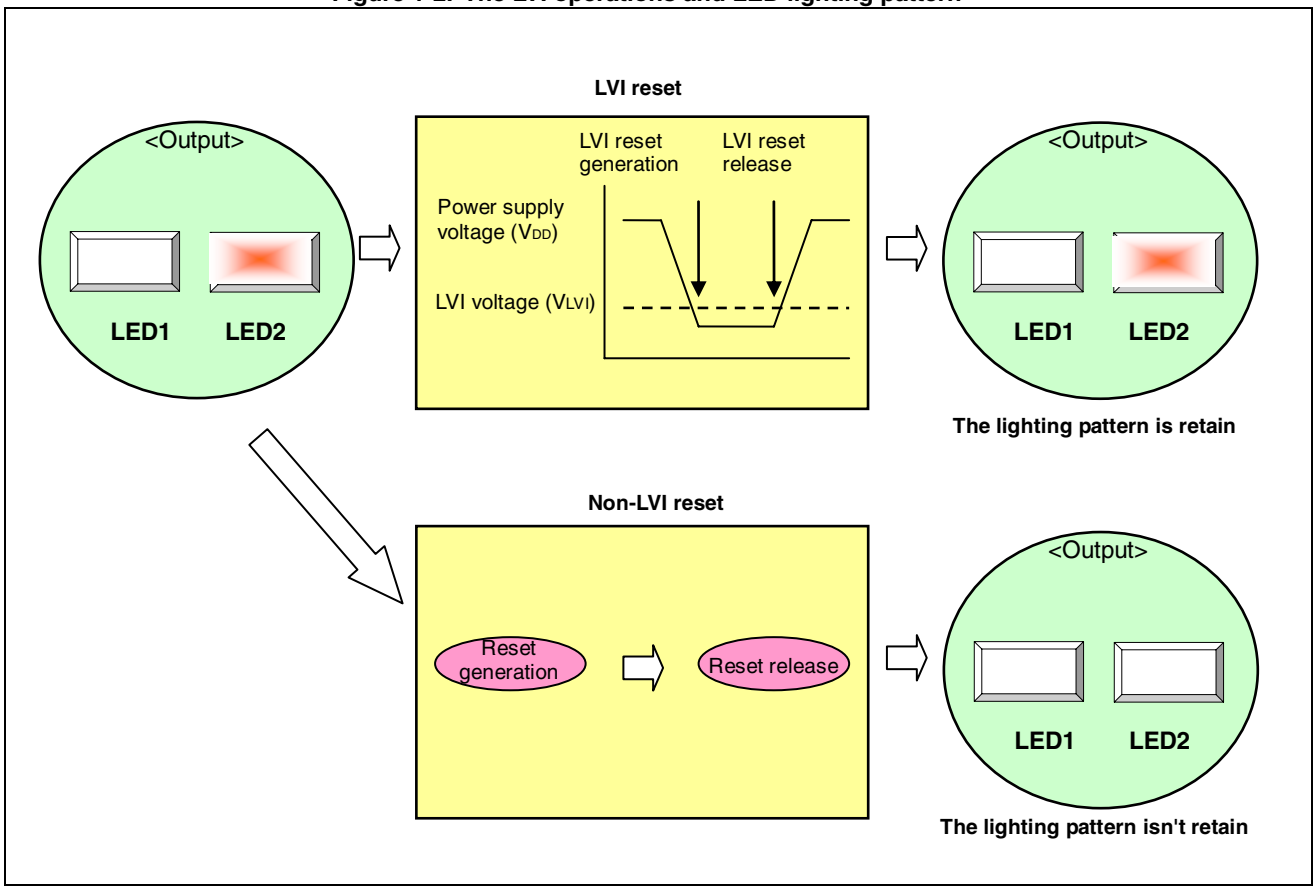
The low-voltage detector (LVI) occurs an internal reset (LVI reset) or an internal interrupt (INTLVI) when V_{DD} becomes less than V_{LVI} (The interrupt occurrence is selected, then the internal interrupt is also generated when V_{DD} becomes more than V_{LVI} after the low-voltage ($V_{DD} < V_{LVI}$) is detected).

In this sample program, an internal reset (LVI reset) occurrence is selected as the low-voltage detector (LVI) operation when V_{DD} becomes less than V_{LVI}.

When low voltage is detected, register values are initialized, but the internal RAM retains the data immediately before the LVI reset [Note]. Therefore, the LED lighting pattern immediately before the LVI reset is retained, enabling it to be restored after LVI reset release. When a reset is generated by other than LVI, the program initializes the LED lighting pattern and all LEDs are turned off.

Note It is necessary to check the whether the internal RAM is valid or not by referring to Internal RAM data status register (RAMS).

Figure 1-2. The LVI operations and LED lighting pattern



1.3 Main processing

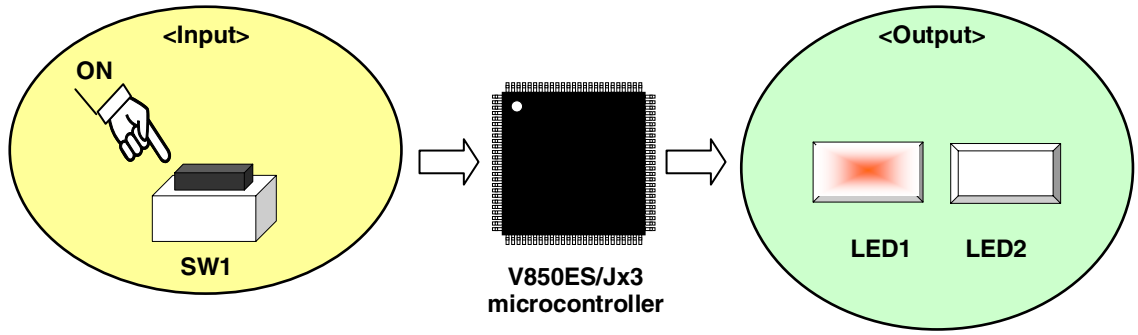
- Enabling interrupts by using the EI instruction
- Executing an infinite loop that executes no processing (waiting for an interrupt generated by switch input)

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.

Figure 1-3. Switch input and LED lighting pattern



Switch input and a change in 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.

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

[Column] Chattering

Chattering is a phenomenon in which the electric signal repeats turning on and off due to a mechanical flip-flop of the contacts, immediately after the switch has been pressed.

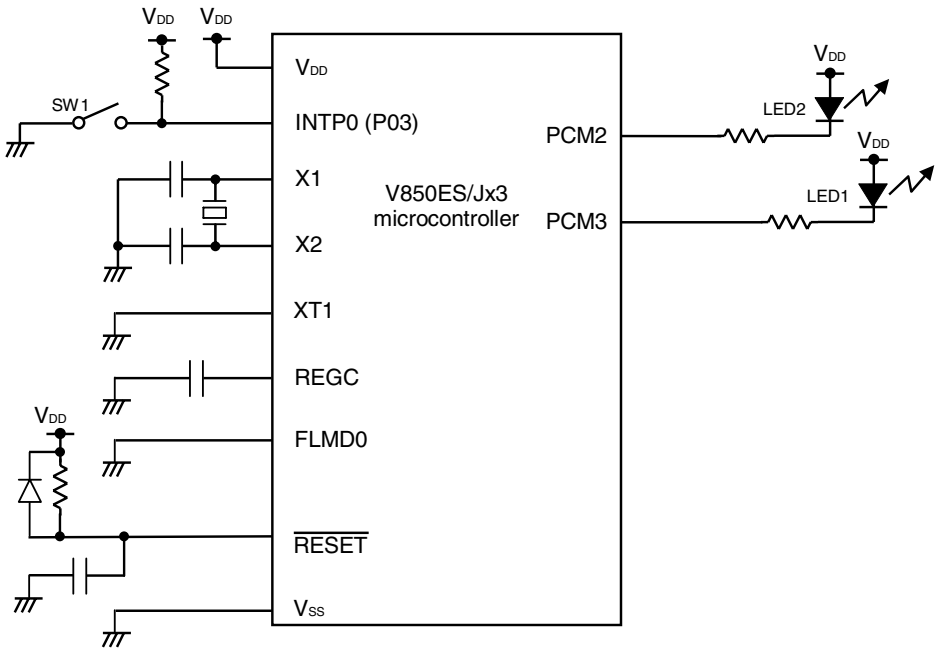
CHAPTER 2 CIRCUIT DIAGRAM

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

2.1 Circuit Diagram

The circuit diagram is shown below.

Figure 2-1. The 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 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.

File Name (Tree Structure)	Description	Compressed (*.zip) Files Included	
			
conf ─ crtE.s	Startup routine file[Note 1]	–	●
├ AppNote_LVI.dir	Link directive file[Note 2]	●	●
├ AppNote_LVI.prj	Project file for integrated development environment PM+	–	●
└ AppNote_LVI.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 and Use the Sample file” 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\CA850*Version used*\lib850\r32\crtE.s.)

2. This is the link directive file automatically generated when “Create and Use the Sample file” is selected and “Memory Usage: Use Internal memory only” 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.

- Low-voltage detector ($V_{DD} < V_{LVI}$): LVI
- 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 and settings such as the setting for stopping the watchdog timer 2, the setting of the I/O ports and external interrupt pins, setting of interrupts, and setting of LVI are performed as the initial settings.

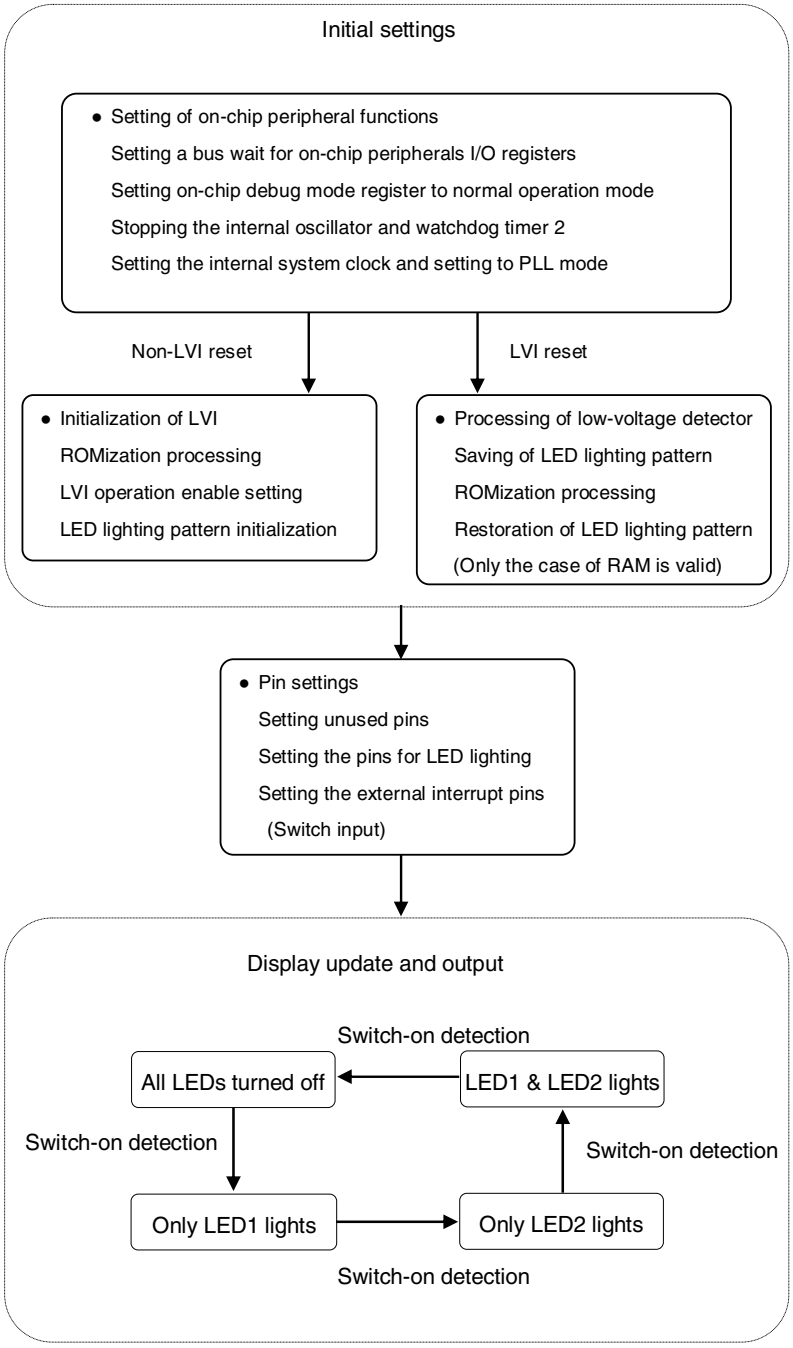
After completion of the initial settings, the lighting pattern of two LEDs (LED1 and LED2) is controlled according to the number of switch (SW1) inputs, by detecting the falling edge of the switch input (SW1) and performing interrupt servicing (This processing is the same as that described in the **V850ES/Jx3 Sample Program (Interrupt) External Interrupt Generated by Switch Input Application Note**.)

When the system is reset, the internal peripherals are initialized, but the internal RAM is not initialized and retain the data [Note]. However the variables without initial value are cleared by startup routine which boot up after the release of reset. In the case of LVI reset is generated, the variables with initial values, it is not initialized by startup routine, should be used to avoid initialize of retaining LED lighting pattern. Concretely the LED lighting pattern values defined by variables with initial values are saved to variables without initial values immediately, then the variables with initial values are initialized (ROMization processing) and return from variables without initialized values (For details of ROMization, see the **3.6 ROMization**).

When a reset is generated by other than LVI, the program initializes the LED lighting pattern and all LEDs are turned off.

This is described in detail in the state transition diagram shown on the next page.

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)

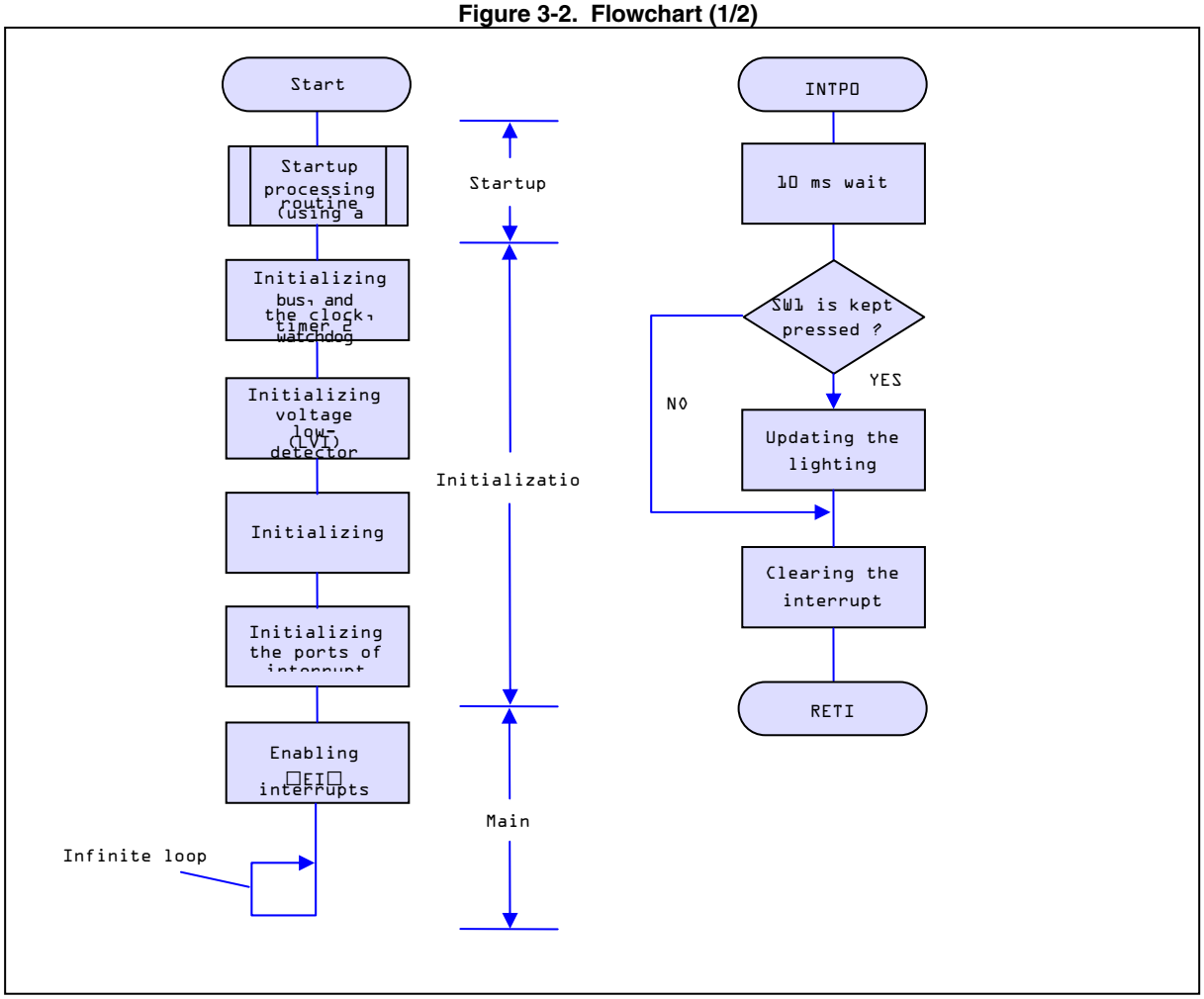
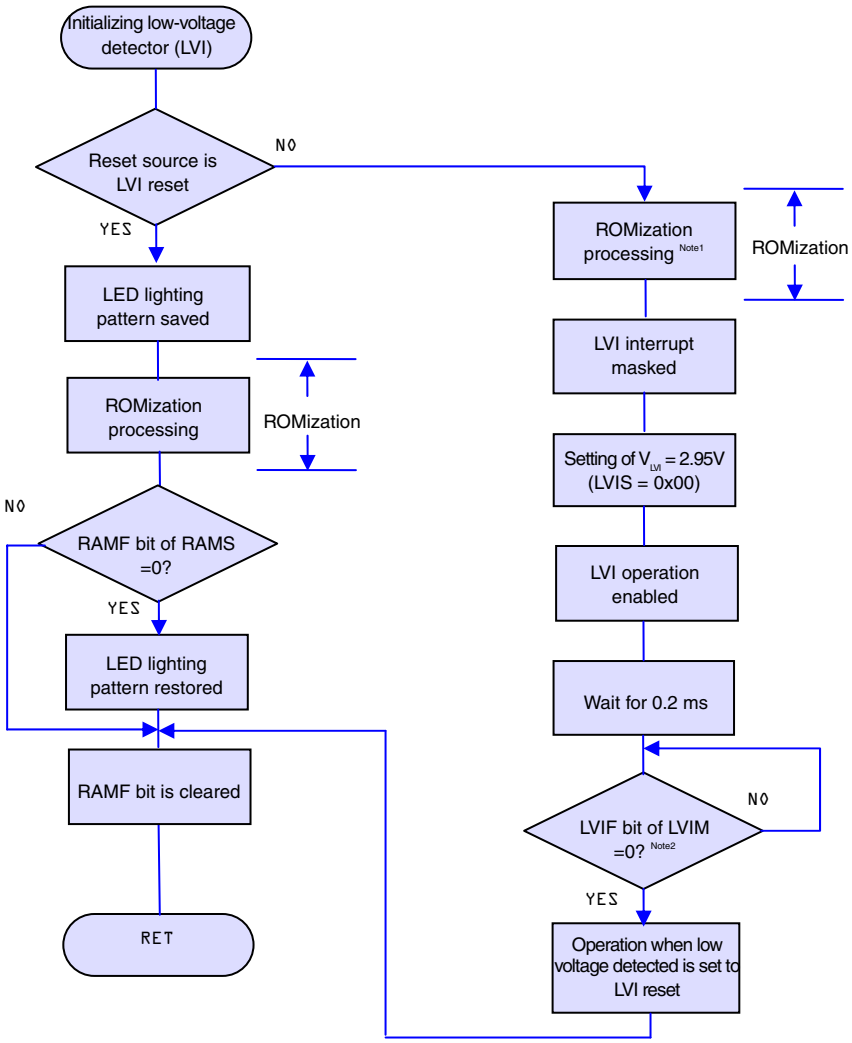


Figure 3-3. Flowchart (2/2)



Notes 1. If the reset source is a non-LVI reset, the LED lighting pattern is initialized as part of the ROMization processing.

2. The LVIF bit of LVIM is clear when V_{DD} rises to greater than V_{LVI}.

[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 port initialization range of ports in I/O initialization differs.

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

3.6 ROMization

In this sample program, the ROMization information is copied during initialization of LVI.

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].

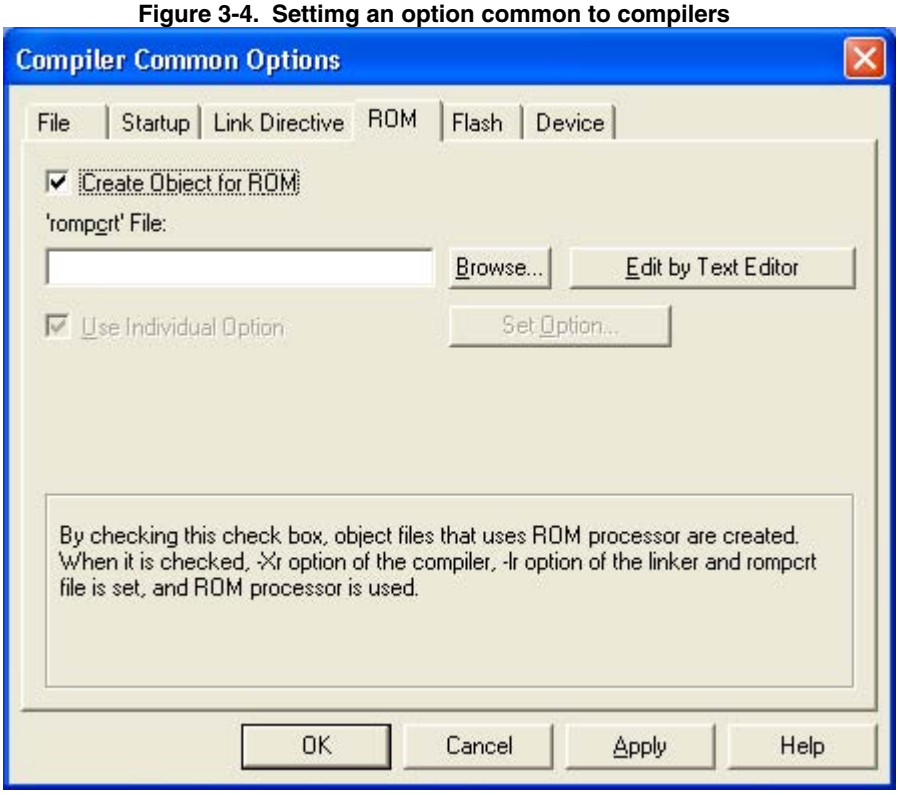
If a variable that has an initial value is used in the program to be created, ROMization information must be generated and copied. Furthermore, the ROMization information must be copied before using the variable that has an initial value.

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”.

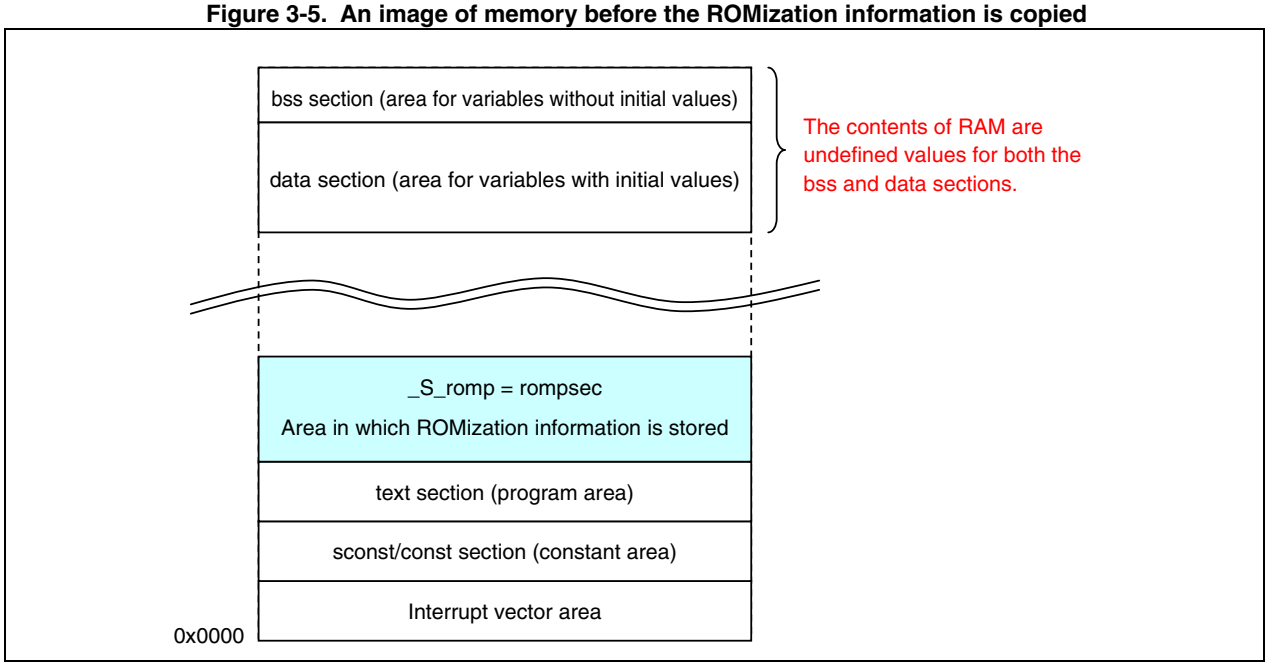
Figure 3-4. Settimg an option common to compilers



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.

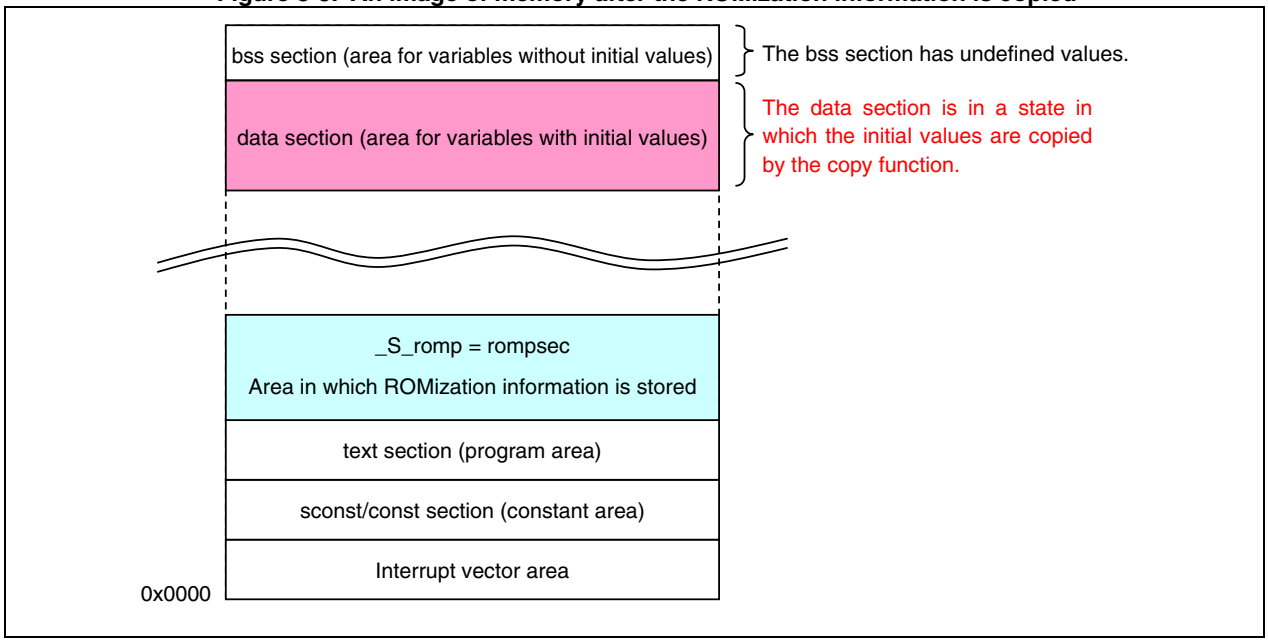
Figure 3-5. An image of memory before the ROMization information is copied



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.

Figure 3-5. An image of memory after the ROMization information is copied



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.

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

3.8 Notes on Operating Sample Program When Using MINICUBE2

When operating the sample program via the on-chip debug emulator, be aware that a program break will occur when an LVI reset is generated. Also, when the operating voltage of MINICUBE2 is less than 2.7 V, MINICUBE2 and the target device may not be able to communicate properly, which may lead to a malfunction.

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 low-voltage detector (LVI) 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 Setting Low-Voltage Detector (LVI)

The low-voltage detector has the following two types of operation modes (In this sample program, using it as a reset).

- Using it as a reset (see **[Example 1]**)
- Using it as an interrupt (see **[Example 2]**)

The low-voltage detector is mainly controlled by the following three types of registers.

- Low-voltage detection register (LVIM)
- Low-voltage detection level select register (LVIS)
- Internal RAM data status register (RAMS)

4.1.1 Settings regarding low-voltage detection

The low-voltage detection register (LVIM) is used to set the low-voltage detection operation mode and control the operation.

In this sample program, the LVIM register is set so that the internal reset signal is generated when a low voltage is detected.

The LVIM register can be read or written in 8-bit or 1-bit units. However, the LVIF flag can only be read.

The value of the LVIM register after reset is 0x82 in the case of a reset generated by the detection of a low voltage, and 0x00 in the case of a reset generated by another source.

Figure 4-1. Format of LVIM Register

Low-voltage detection register (LVIM)
Address: 0xFFFFF890

7	6	5	4	3	2	1	0
LVION	0	0	0	0	0	LVIMD	LVIF

LVION	Enabling or disabling of low-voltage detection operation
0	Disables operation
1	Enables operation

LVIMD	Low-voltage detection operation mode selection
0	The interrupt request signal INTLVI is generated when the power supply voltage drops or rises across the detection voltage value
1	The internal reset signal LVIRES is generated when the power supply voltage falls below the detection voltage.

LVIF	Low-voltage detection flag (read only)
0	When power supply voltage > detection voltage or when operation is prohibited
1	When connected power supply voltage < detection voltage

Remarks 1. The red values (LVION, LVIMD) indicate the setting values used in the sample program.

2. The blue values (LVIF) indicate the bits to be checked in the sample program.

4.1.2 Settings regarding low-voltage detection level

The low-voltage detection level select register (LVIS) is used to set the low-voltage detection level.

The low-voltage detection level of V850ES/JJ3 and V850ES/JG3 is fixed to 2.95 V (typ.) ± 0.10 V.

The LVIS register can be read or written in 8-bit units.

This register cannot be written until a reset request due to something other than low-voltage detection is generated after the LVIM.LVION and LVIM.LVIMD bits are set to 1.

Figure 4-2 Format of LVIS Register

Low-voltage detection level select register (LVIS)
Address: 0xFFFFF891

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	LVIS0

LVIS0	Control of low-voltage detection level
0	2.95 V (typ.) ± 0.10 V
1	Reserved (setting prohibited)

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

4.1.3 Controls regarding RAM retention voltage detection

The internal RAM data status register (RAMS) is a flag register that indicates whether the internal RAM is valid or not.

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

The RAMS register is a special register. This can be written only in a special combination of sequences.

The set/clear conditions for the RAMF bit are shown below.

Setting conditions : Detection of voltage lower than specified level
Writing of 1 in specific sequence
Clearing conditions: Writing of 0 in specific sequence

Figure 4-3 Format of RAMS Register

Internal RAM data status register (RAMS)
Address: 0xFFFFF892

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	RAMF

RAMF	Internal RAM voltage detection
0	Voltage lower than RAM retention voltage is not detected.
1	Voltage lower than RAM retention voltage is detected.

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

[Example 1] Using the low-voltage detection as a reset by setting the low-voltage detection level (V_{LVI}) to 2.95 V (TYP.) ± 0.10 V (same content as the sample program)

- Setting procedure
  - <1> Mask the LVI interrupt.
  - <2> Set the detection voltage using the LVIS.LVIS0 bit.
  - <3> Set the LVIM.LVION bit to 1 (operation enabled).
  - <4> Use software to insert a wait of at least 0.2 ms (a wait of 0.25 ms is set in the program example).
  - <5> Check the LVIM.LVIF bit to see if the power supply voltage is higher than the detection voltage.
  - <6> Set the LVIMD bit to 1 (internal reset generated).

Caution Once the LVIMD bit has been set to 1, the LVIM and LVIS registers cannot be changed until a non-LVI reset occurs.

- Program example (same content as the sample program)

```
LVIMK = 1;       /* Mask the LVI interrupt                          */   <1>
LVIS = 0x00;     /* Set the low-voltage detection level to 2.95 V±0.10 V  */   <2>
/* Enable the low-voltage detector operation */
#pragma asm
    push r10
    mov 0x80, r10
    st.b r10, PRCMD
    set1 LVION                                                           <3>
    pop r10
#pragma endasm

/* Wait of 250 μs(0.25 ms) */
for( loop_wait = 0 ; loop_wait < LIMIT_250us_WAIT ; loop_wait++ )        <4>
{
       __nop();
}

while( LVIF == 1 );   /* Note that the program enters an infinite    */   <5>
                      /* loop until the power supply voltage rises   */
                      /* above the detection voltage                 */
/* Setting operation when low voltage is detected to reset operation */
#pragma asm
    push r10
    mov 0x82, r10
    st.b r10, PRCMD                                                      <6>
    set1 LVIMD
    pop r10
#pragma endasm
```

[Example 2] Using the low-voltage detection function as an interrupt by setting the low-voltage detection level (V_{LVI}) to 2.95 V (TYP.) ± 0.10 V.

- Setting procedure
  - <1> Mask the LVI interrupt.
  - <2> Set the detection voltage using the LVIS.LVIS0 bit.
  - <3> Set the LVIM.LVION bit to 1 (operation enabled).
  - <4> Use software to insert a wait of at least 0.2 ms (a wait of 0.25 ms is set in the program example).
  - <5> Check the LVIM.LVIF bit to see if the power supply voltage is higher than the detection voltage.
  - <6> Clear the interrupt request flag of LVI.
  - <7> Release the interrupt mask of LVI

- Program example

```
LVIMK = 1;       /* Mask the LVI interrupt                          */    <1>
LVIS = 0x00;     /* Set the low-voltage detection level to 2.95 V±0.10 V  */    <2>
/* Enables the low-voltage detector operation /
#pragma asm
    push r10
    mov 0x80, r10
    st.b r10, PRCMD
    set1 LVION                                                           <3>
    pop r10
#pragma endasm

/* Wait of 250 μs(0.25 ms) */
for( loop_wait = 0 ; loop_wait < LIMIT_250us_WAIT ; loop_wait++ )        <4>
{
       __nop();
}

while( LVIF == 1 );   /* Note that the program enters an infinite     */    <5>
                      /* loop until the power supply voltage rises    */
                      /* above the detection voltage                  */

LVIIF = 0;            /* Clear the interrupt request flag of LVI        */    <6>

LVIMK = 0;            /* Release the interrupt mask of LVI              */    <7>
```

4.2 Checking Detection of LVI Reset

When an LVI reset occurs, the LVIRF 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 LVIRF bit is cleared. Therefore, by checking the LVIRF bit after the reset is released, it is possible to ascertain whether the reset source was an LVI reset or an input to the $\overline{\text{RESET}}$ pin.

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

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

Remarks 1. 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.

[Clearing the reset source flag]
As mentioned in Remark 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 low-voltage detector (LVI) is being 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 series
#
#-------------------------------------------------------------------------------
#    V850ES/JJ3 JG3 sample program
#-------------------------------------------------------------------------------
#    Reset Generation When Low Voltage Detected
#-------------------------------------------------------------------------------
#[History]
#    2009.9.--   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;
};
```

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

A reserved area for MINICUBE2 is secured.

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

● main.c

```
/*---------------------------------------------------------------------------*/
/*
/*  NEC Electronics     V850ES/Jx3 series
/*
/*---------------------------------------------------------------------------*/
/*  V850ES/JJ3 sample program
/*---------------------------------------------------------------------------*/
/*  Reset Generation When Low Voltage Detected
/*---------------------------------------------------------------------------*/
/*[History]
/*  2009.09.--  Released
/*---------------------------------------------------------------------------*/
/*[Overview]
/* This sample program presents an example of using the low-voltage detector (LVI).
/* The low-voltage detector (LVI) is set so that an internal reset (LVI reset) signal
/* is generated when LVI detects that VDD is less than VLVI (2.95 V (typ.)± 0.10 V).
/* After completion of the initial settings, interrupt servicing is executed at the
/* falling edge of the switch input and the LED lighting pattern is displayed in
/* accordance with the number of switch inputs.
/* If a reset is generated by other than LVI, the LED lighting pattern held in the
/* internal RAM is initialized. If a reset is generated by LVI, the LED lighting
/* pattern immediately before the LVI reset is retained in the internal RAM and then
/* restored and displayed after the LVI reset is released.
/*
/*
/*  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 a wait adjustment value of 0.2 ms for setting the low-voltage detection
/*    register (LVIM)
/*  • Performing prototype definitions
/*  • Defining the LED lighting pattern table
/*  • Setting a bus wait for on-chip peripheral I/O registers, stopping the watchdog
/*    timer 2, and setting the clock
/*  • Executing low-voltage detector initialization
/*  • Initializing unused ports
/*  • Initializing external interrupt ports (falling edge) and LED output ports
/*  • ROMization
/* <Interrupt servicing>
/*  • Updating the LED lighting pattern
/*   (Chattering elimination time during switch input: 10 ms)
```

```
/*
/* <Switch input and LED lighting pattern>
/*
/*  ---------------—---------------—-----------+
/*  Switch input count      LED2        LED1
/*     (P03/INTP0)       (PCM2)      (PCM3)
/*  ----------------------------------------
/*          0              OFF         OFF
/*          1              OFF         ON
/*          2              ON          OFF
/*          3              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 : 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 (0x9D58)       /* Defines the constant for a 10 ms wait
                                          adjustment.                          */
#define LIMIT_250us_WAIT (0x3EF)       /* Defines the constant for a 250 us (0.25 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 WDT2 initialization function */
static    void f_init_lvi( void );          /* Low-voltage detector 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]       */


/*------------------------------*/
/* Setting global variables */
/*------------------------------*/
static unsigned char g_led_ptn_cnt = 0;   /* Variables for saving LED lighting pattern */


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

    __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_lvi();                 /* Sets LVI                          */

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

    return;
}


/*--------------------------------------*/
/* Initializing clock bus WDT2          */
/*--------------------------------------*/
static void f_init_clk_bus_wdt2( void )
{
    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

    RCM    = 0x01;              /* Stops the internal oscillator. */
    WDTM2  = 0x00;              /* Stops the watchdog timer 2.    */

    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

    PLLON = 1;                  /* Enable PLL operation          */

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

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

```
      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;
}


  /*-----------------------------------------*/
  /*   Initializing the low-voltage detector   */
  /*-----------------------------------------*/
  static void f_init_lvi( void )
  {
      extern unsigned int _S_romp;          /* Externally references the ROMization
                                               symbols      */
      static unsigned char save_led_ptn;    /* Variables for saving LED lighting
                                               pattern      */
      unsigned int loop_wait;

      /* LVI reset */
      if( RESF.0 == 1 )
      {
          save_led_ptn = g_led_ptn_cnt;     /* Saves the LED lighting pattern      */

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

          if(RAMF == 0)
          {
             g_led_ptn_cnt = save_led_ptn;  /* Restores the LED lighting pattern   */
          }
      }
```

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

```
    /* Reset generated by other than LVI */
    else
    {
        _rcopy( &_S_romp, -1 ); /* Executes ROMization processing                */
                                /* ↑LED lighting pattern specifications
                                       initialized by ROMization                 */

        LVIMK = 1;              /* Masks LVI interrupt                           */
        LVIS = 0x00;            /* Sets low-voltage detection level to 2.95 V±0.10V  */

/* Enabling low-voltage detection operation */
#pragma asm
   push   r10
   mov    0x80, r10
   st.b   r10, PRCMD
   set1   LVION
   pop    r10
#pragma endasm
```

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

```
        /* Inserting 250 us (0.25 ms) wait */
        for( loop_wait = 0 ; loop_wait < LIMIT_250us_WAIT ; loop_wait++ )
        {
            __nop();
        }

        while( LVIF == 1 );
```

/* Note that the program enters an infinite loop until the */
/* power supply voltage rises above the detection voltage */

```
/* Setting operation when low voltage is detected to reset operation */
#pragma asm
    push   r10
    mov    0x82, r10
    st.b   r10, PRCMD
    set1   LVIMD
    pop    r10
#pragma endasm

    }
```

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

```
  /* Clear the flag of the internal RAM retention voltage detected */
#pragma asm
    st.b   r0, PRCMD
    st.b   r0, RAMS
#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;

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

    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;

    PCM    = LED_TBL[g_led_ptn_cnt];        /* Sets PCM0,PCM1, PCM4 and PCM5 to output low
                                            level and values to turn off the LEDs to PCM2
                                            and PCM3. */
    PMCM   = 0xC0;
    PMCCM  = 0x00;

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

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

    PDH    = 0x00;                          /* Sets PDH0 to PDH7 to output low level.    */
    PMDH   = 0x00;
    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/JG3, the setting value is 0xF0.

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

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

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

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

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

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

```
/*****************************/
/*      Interrupt module    */
/*****************************/
__interrupt
void f_int_intp0( void )
{
    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 )   /* Identifies that SW1 is being pressed after wait */

    {
        g_led_ptn_cnt++;           /* Changes the lighting pattern (4 types).       */
        g_led_ptn_cnt &= 3;
        PCM = LED_TBL[g_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. */
}
```

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