

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.

Application Note

V850ES/Jx3-L

Sample Program (Initial Settings)

LED Lighting Switch Control

This document summarizes the initial settings for the sample program of the V850ES/Jx3-L and describes the basic initial settings for the microcontroller. In the sample program, the lighting of two LEDs is controlled by using one switch input, after the basic initial settings for the peripheral functions of the microcontroller, such as selecting the clock frequency or I/O ports, have been performed.

Target devices

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

Document No. U19479EJ1V0AN00
Date Published November 2008 N

© NEC Electronics Corporation 2008
Printed in Japan

CONTENTS	
CHAPTER 1 OVERVIEW	3
1.1 Initial Settings	3
1.2 Contents of Main Processing Operation	3
CHAPTER 2 CIRCUIT DIAGRAM	5
2.1 Circuit Diagram	5
2.2 Peripheral Hardware	5
CHAPTER 3 SOFTWARE	6
3.1 File Configuration	6
3.2 On-Chip Peripheral Functions Used	7
3.3 Initial Settings and Operation Overview	8
3.4 Flowchart	9
3.5 Differences Between V850ES/JG3-L and V850ES/JF3-L	10
3.6 ROMization (C Language Only)	10
3.7 Security ID	12
CHAPTER 4 SETTING REGISTERS	14
4.1 Option Byte Setting	15
4.2 Setting System Wait Control Register (VSWC)	16
4.3 Setting Special Registers	17
4.3.1 Special registers	17
4.3.2 Setting data to special registers	17
4.3.3 Disabling DMA operations	17
4.4 Setting Normal Operation Mode for On-Chip Debugging	19
4.5 Setting Internal Oscillation Mode Register (RCM)	21
4.6 Setting Watchdog Timer 2	22
4.7 Clock Setting	23
4.7.1 Processor clock control register (PCC) setting	23
4.7.2 Lock register (LOCKR)	25
4.7.3 Setting PLL control register (PLLCTL)	26
4.8 Setting Ports	28
4.8.1 Port n register (Pn)	28
4.8.2 Port n mode register (PMn)	29
4.8.3 Port n mode control register (PMCn)	29
4.9 Main Processing	32
4.9.1 Chattering countermeasure	32
4.9.2 Main processing	34
CHAPTER 5 RELATED DOCUMENTS	39
APPENDIX A PROGRAM LIST	40

• **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).

CHAPTER 1 OVERVIEW

In this sample program, the basic initial settings for the V850ES/Jx3-L microcontroller, such as selecting the clock frequency and setting the I/O ports, are performed. In the main processing operation after completing the initial settings, the lighting of two LEDs is controlled by using one switch input.

1.1 Initial Settings

<Referencing option byte>

- Referencing the oscillation stabilization time after releasing reset

<Main contents of initial settings>

- Setting the system wait control register to one clock
- Setting on-chip debugging to normal operation mode
- Stopping the internal oscillator
- Stopping watchdog timer 2 operation
- Setting the system clock to 20 MHz by multiplying the input clock by 4 using the PLL
- Setting unused ports
- Setting the switch input and LED control ports

< ROMization >

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

1.2 Contents of Main Processing Operation

The lighting of two LEDs (LED1, LED2) is controlled according to the number of switch (SW1) inputs in the V850ES/Jx3-L microcontroller.

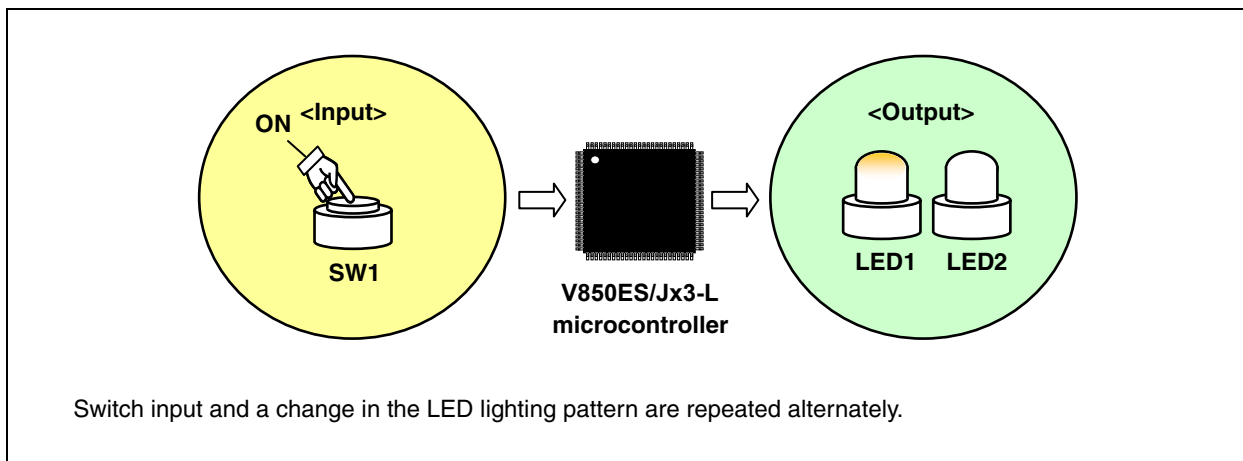


Table 1-1. LED Lighting Patterns

Switch (SW1) Input Count ^{Note}	LED1	LED2
0	OFF	OFF
1	ON	OFF
2	ON	ON
3	OFF	ON

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

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



[Column] What is chattering?

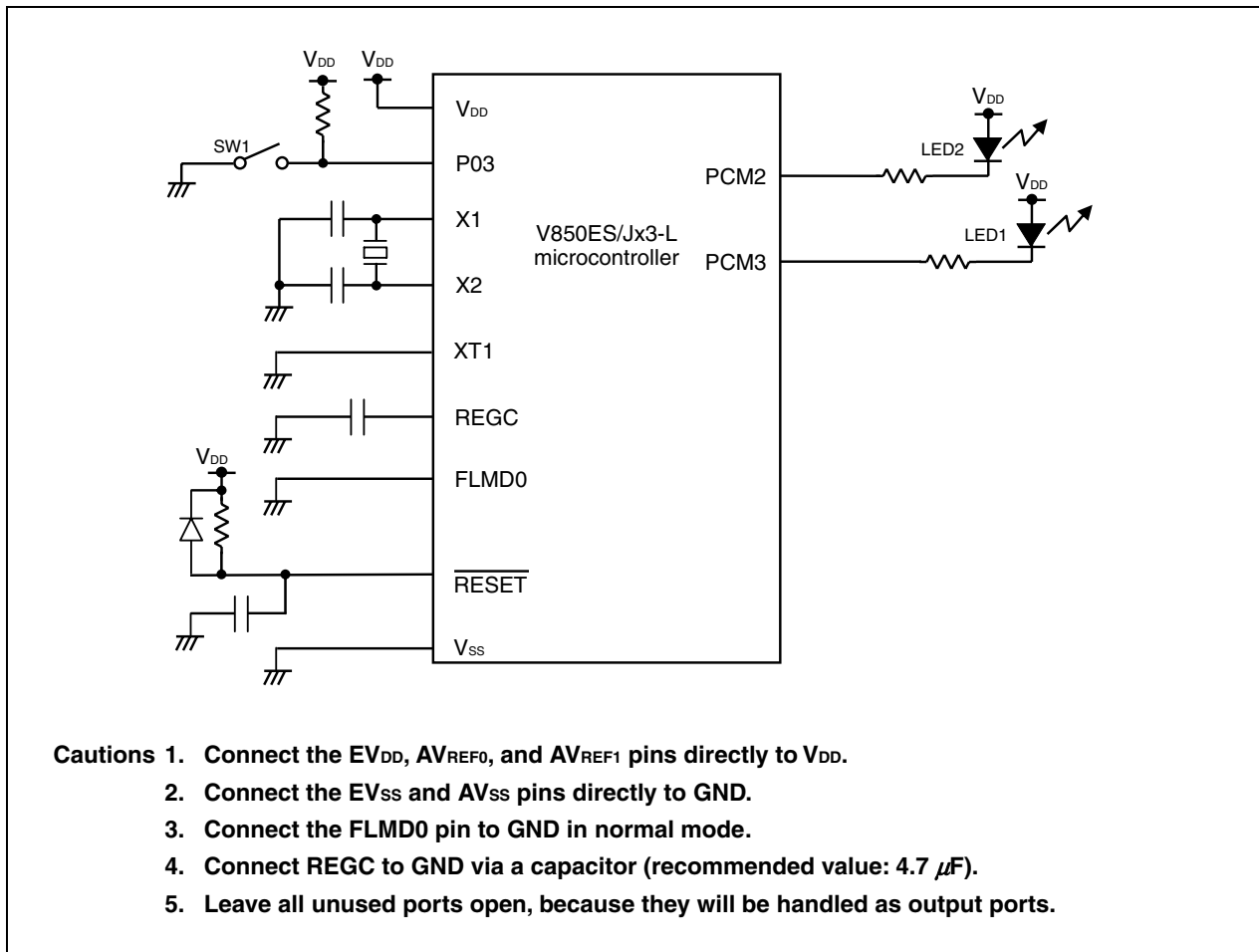
Chattering is a phenomenon that an electric signal alternates between being on and off when a connection flip-flops mechanically immediately after a switch is switched.

CHAPTER 2 CIRCUIT DIAGRAM

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

2.1 Circuit Diagram

The circuit diagram is shown below.



2.2 Peripheral Hardware

The peripheral hardware to be used is shown below.

(1) Switch (SW1)

This switch is used as an 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	
			
<pre> c ├── conf │ ├── crtE.s │ ├── AppNote_LED.dir │ ├── AppNote_LED.prj │ └── AppNote_LED.prw └── src ├── main.c ├── minicube2.s └── opt_b.s </pre>	Startup routine file ^{Note 1}	-	●
	Link directive file ^{Note 2}	●	●
	Project file for integrated development environment PM+	-	●
	Workspace file for integrated development environment PM+	-	●
	C language source file including descriptions of hardware initialization processing and main processing of microcontroller	●	●
	Source file for reserving area for MINICUBE2	●	●
	Source file for setting option byte	●	●

- 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.

[Assembly language version]

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

- Notes**
- 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.)
 - 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.

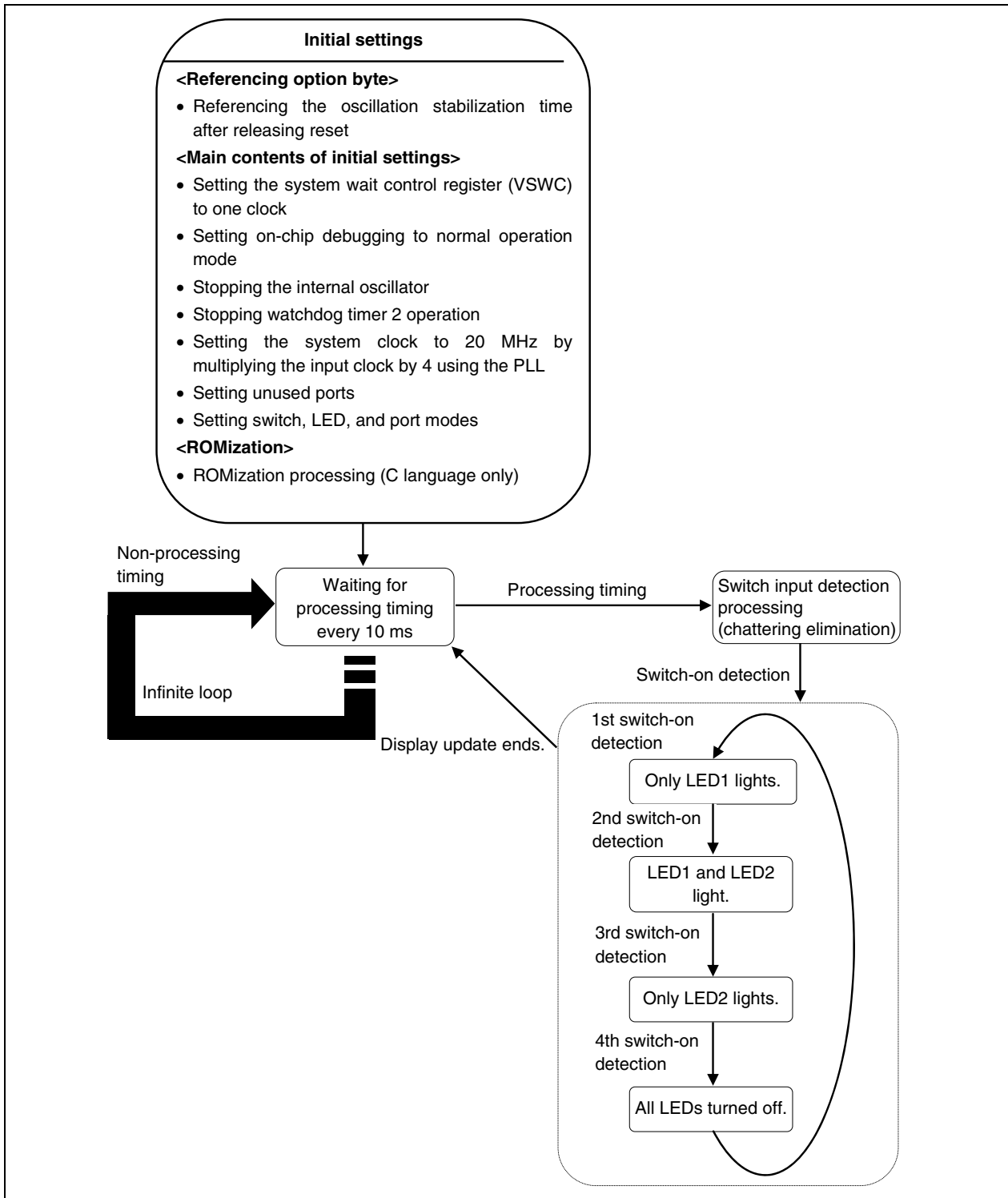
- Input ports (for switch input): P03 (SW1)
- Output ports (for lighting LEDs): PCM3 (LED1), PCM2 (LED2)

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 and the setting of the I/O ports are performed as the initial settings.

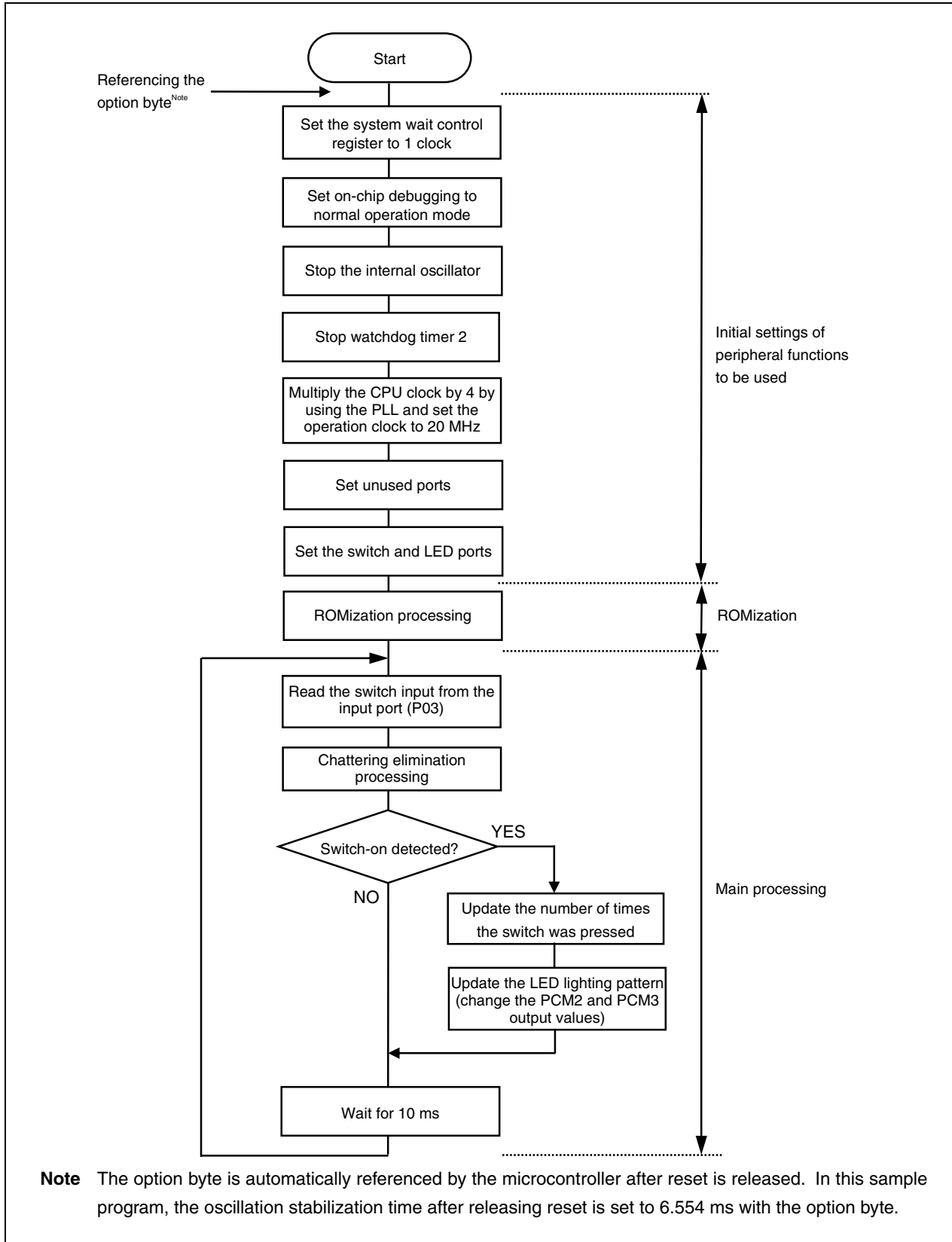
After completing the initial settings, the lighting of two LEDs (LED1 and LED2) is controlled according to the number of switch (SW1) inputs.

This is described in detail in the state transition diagram shown below.



3.4 Flowchart

A flowchart for the sample program is shown below.



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 port initialization range 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.

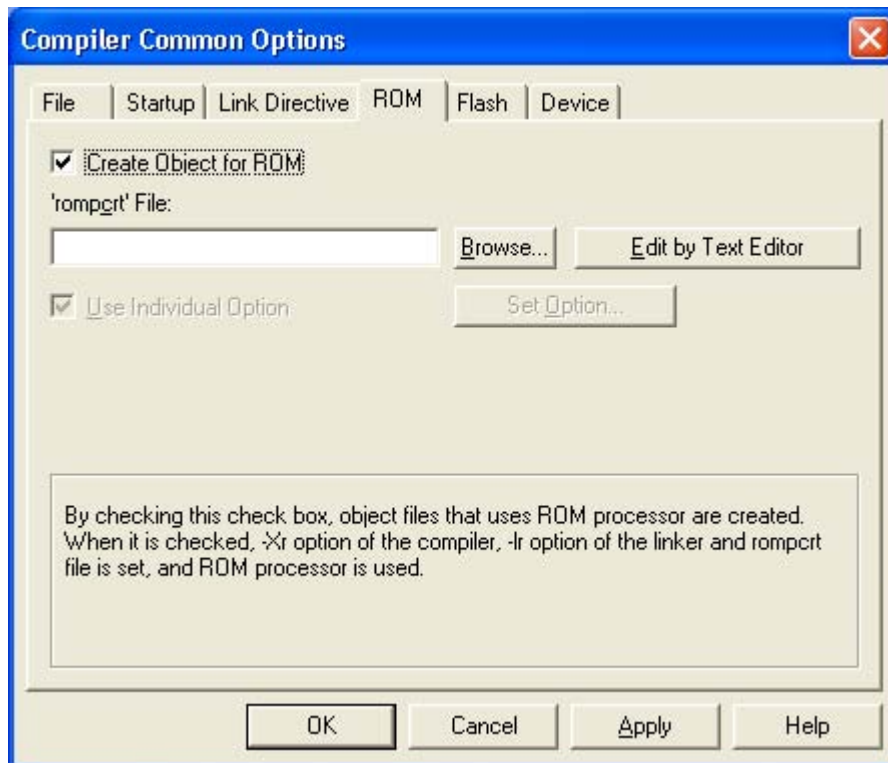
ROMization information is the information of the initial values of variables that have initial values (the section to which variables that have initial values are placed). Variables that have initial values (the section to which variables that have initial values are placed) will hold their software-based 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.

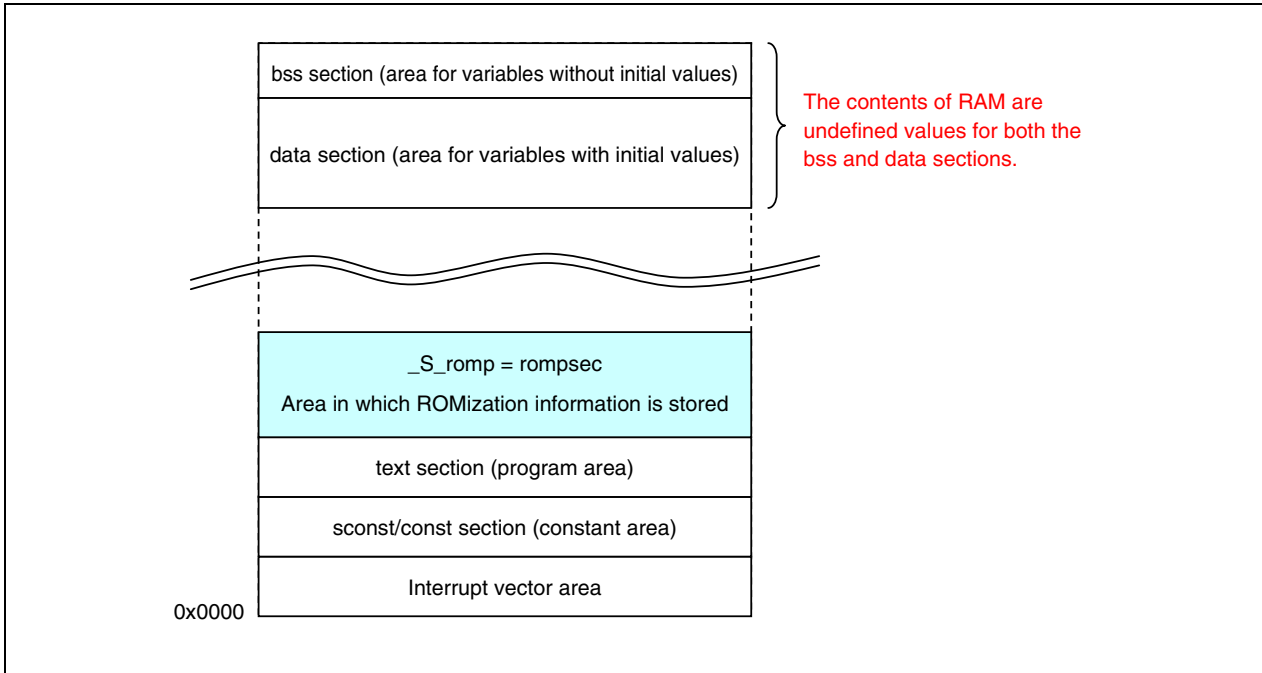
The ROMization procedure is described below.

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



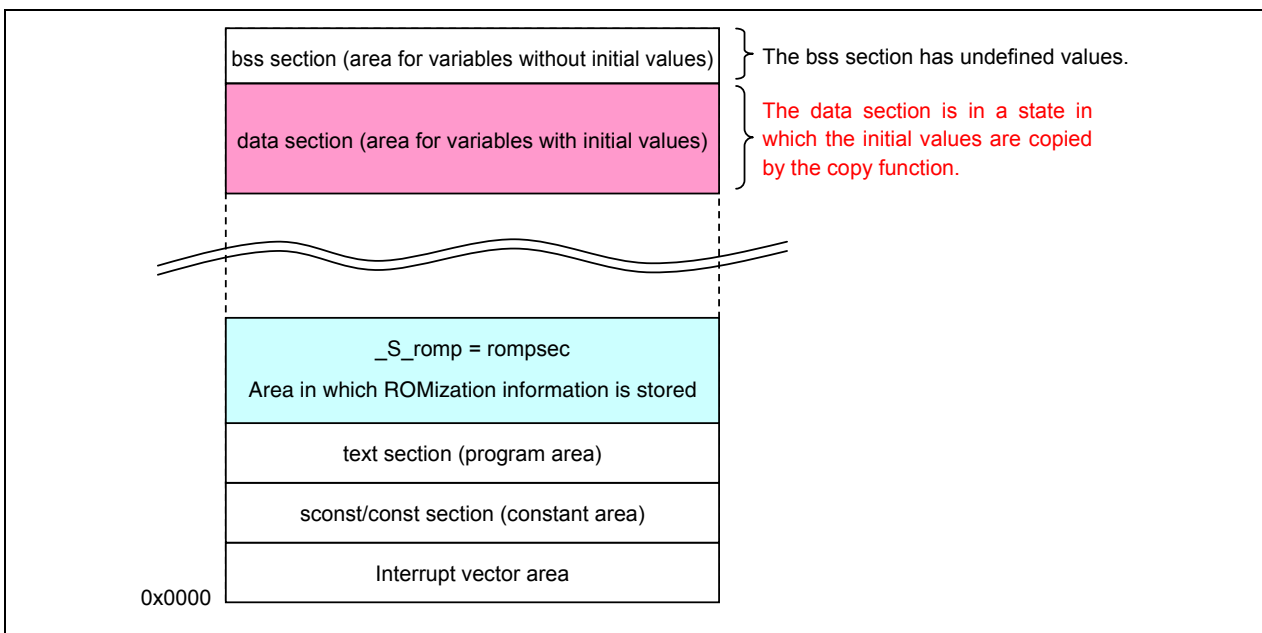
The section into which the ROMization information is to be stored (rompsec) will be automatically added immediately after the program area (.text) section. However, by checking “Create Object for ROM”, a code that indicates the same address as that of rompsec will be generated for the default label `_S_romp`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



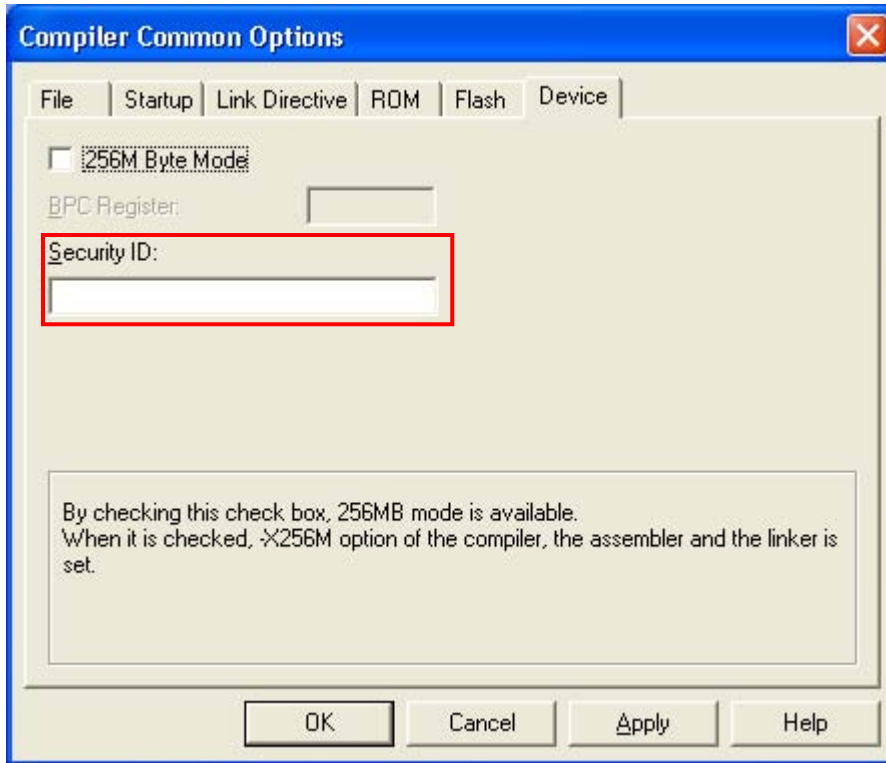
3.7 Security ID

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.

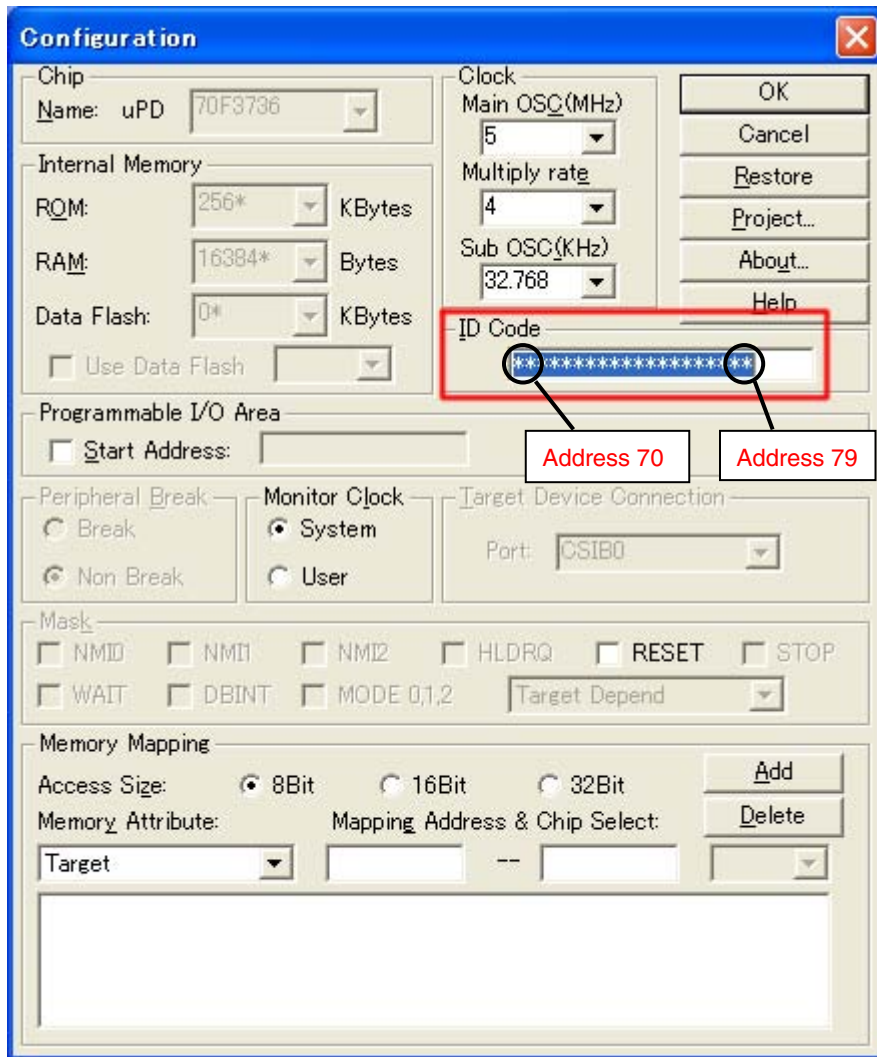


Remark Set the security ID for a device provided with flash memory in the “Security ID” field, which is an option common to all compilers.

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 details of the option byte, system wait control register, on-chip debugging, watchdog timer, DMA, internal system clock, PLL mode, pin function setting, and main processing.

See the following user's manuals for details of how to set registers.

- V850ES/JG3-L 32-bit Single-Chip Microcontrollers
Hardware User's Manual
- V850ES/JF3-L 32-bit Single-Chip Microcontrollers
Hardware User's Manual

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

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

4.1 Option Byte Setting

The option byte must be set. The option byte is stored at address 0x000007A of the internal flash memory (internal ROM area) as 8-bit data. This 8-bit data is used to set the oscillation stabilization time after reset is released. After reset is released, the oscillation stabilization time will be secured according to this value.

Set the oscillation stabilization time to a value that at least satisfies the oscillation stabilization time of the resonator by evaluating matching between the V850ES/JG3-L and resonator in an actual application.

In this sample program, the option byte is set by using opt_b.s.

Figure 4-1. Option Byte Format

Address: 0x000007A							
7	6	5	4	3	2	1	0
0	0	0	0	0	RESOSTS2	RESOSTS1	RESOSTS0

RESOSTS2	RESOSTS1	RESOSTS0	Oscillation stabilization time selection (logical value)	fx		
				2.5 MHz	5 MHz	10 MHz
				0	0	0
0	0	1	$2^{11}/fx$	819.2 μ s	409.6 μ s	Setting prohibited
0	1	0	$2^{12}/fx$	1.638 ms	819.2 μ s	409.6 μ s
0	1	1	$2^{13}/fx$	3.277 ms	1.638 ms	819.2 μ s
1	0	0	$2^{14}/fx$	6.554 ms	3.277 ms	1.638 ms
1	0	1	$2^{15}/fx$	13.11 ms	6.554 ms	3.277 ms
1	1	0	$2^{16}/fx$	26.21 ms	13.11 ms	6.554 ms
1	1	1	$2^{16}/fx$	26.21 ms	13.11 ms	6.554 ms

Caution Be sure to write 6 bytes of data to the option byte section.
If less than 6 bytes are written, an error will occur when linking is executed.

Remarks 1. Set addresses 0x0000007B to 0x0000007F to 0x00.
 2. The red values indicate the values set in the sample program.

- Common to C language and assembly language versions

```
.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  Addresses 0x7b to 0x7f must be set to 0x00.
.byte 0b00000000 -- 0x7e          ↓
.byte 0b00000000 -- 0x7f          ↓
```

4.2 Setting System Wait Control Register (VSWC)

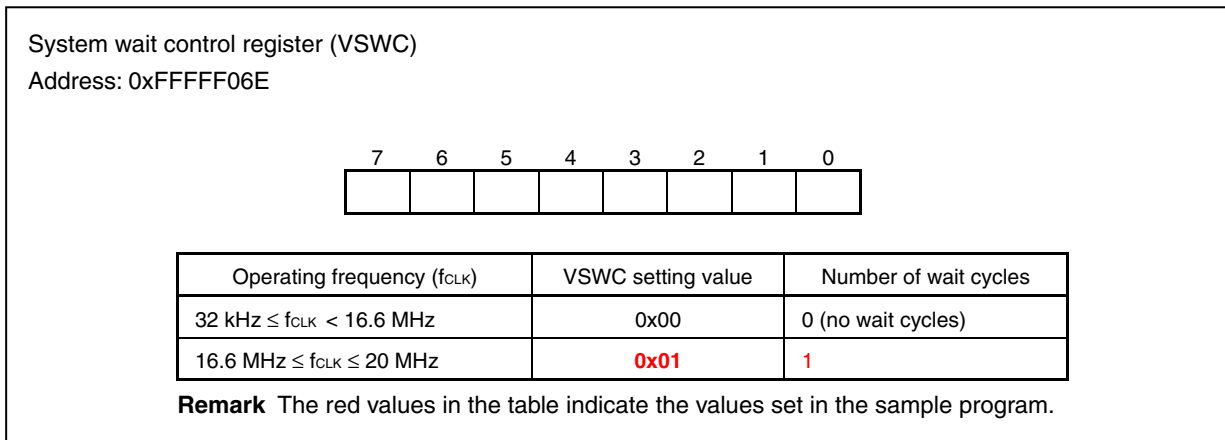
The VSWC register is used to control wait cycles for bus access to the on-chip peripheral I/O registers.

An on-chip peripheral I/O register can be accessed in three clocks (no wait cycles), but the V850ES/Jx3-L requires wait cycles according to the operating frequency. Set the following values to the VSWC register in accordance with the operating frequency used.

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

Reset sets this register to 0x77.

Figure 4-2. VSWC Register Format



The value set to VSWC is **0x01**.

- C language

```
VSWC = 0b00000001; /* Inserts one wait cycle when an on-chip peripheral I/O register is accessed.*/
```

- Assembly language

```
mov 0x01, r11      -- Inserts one wait cycle when an on-chip peripheral I/O register is accessed.
st.b r11, VSWC
```

4.3 Setting Special Registers

The on-chip debug mode register (OCDM) and processor clock control register (PCC) are set in the initial setting procedure. These registers are special registers and must be written in a specific sequence.

4.3.1 Special registers

Special registers are registers that are protected so that no illegal data will be written due to an infinite loop. The V850ES/Jx3-L is provided with the following seven special registers.

- Power-save control register (PSC)
- Clock control register (CKC)
- Processor clock control register (PCC)
- Clock monitor mode register (CLM)
- Reset source flag register (RESF)
- Low-voltage detection register (LVIM)
- On-chip debug mode register (OCDM)

The PRCMD register protects the special registers from being written so that application systems are not inadvertently stopped by an infinite loop. The special registers are accessed for writing via a special sequence and illegal store operations are reported to the SYS register.

4.3.2 Setting data to special registers

Write data to a special register in the following sequence.

- <1> Disable DMA operations.
- <2> Prepare the data to be written to the special register in any general-purpose register.
- <3> Write the data prepared in step <2> to the PRCMD register.
- <4> Write the data to the special register by using the following instructions.
 - Store instruction (ST/SST instruction)
 - Bit manipulation instruction (SET1/CLR1/NOT1 instruction)
 (<5> to <9>: Insert five NOP instructions.) (Only when the PSC.STP bit is set to 1.)
- <10> Enable DMA operations if required.

4.3.3 Disabling DMA operations

DMA operations must be disabled in order to write data to a special register.

DMA channel control registers 0 to 3 (DCHC0 to DCHC3) can be used to enable or disable DMA transfer for DMA channel n.

Set the DCHC.Enn bit (bit 0) to enable or disable DMA transfer for DMA channel n.

Remark After reset is released, the initial values of the DMA channel control registers are 0x00 and DMA operations are disabled. If it is clear that DMA operations are disabled, such as during the initial settings, steps <1> and <10> can be omitted. The sample program does not include step <1>.

Figure 4-3. DCHCn Register Format

DMA channel control registers 0 to 3 (DCHC0 to DCHC3)

Address: 0xFFFFF0E0 (DCHC0), 0xFFFFF0E2 (DCHC1), 0xFFFFF0E4 (DCHC2), 0xFFFFF0E6 (DCHC3)

7	6	5	4	3	2	1	0
TCn	0	0	0	0	INITn	STGn	Enn

Enn	Setting for enabling or disabling DMA transfer for DMA channel n
0	Disables DMA transfer.
1	Enables DMA transfer.

After reset is released, DMA stop processing can be omitted, because DMA transfer is disabled (DCHCn.Enn bit = 0).

4.4 Setting Normal Operation Mode for On-Chip Debugging

Use the OCDM register to switch between normal operation mode and on-chip debug mode and to specify whether to use the alternate-function pin to which the on-chip debug function is assigned as an on-chip debug pin or as a normal port/peripheral function alternate-function pin. At the same time, use this register to control disconnecting the on-chip pull-down resistor of the P05/INTP2/ $\overline{\text{DRST}}$ pin.

The OCDM register is a special register. It can be written only by using a combination of specific sequences (see **4.3.2 Setting data to special registers**).

Writing to the OCDM register is enabled only when the $\overline{\text{DRST}}$ pin is at low level.

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

The value of the OCDM register becomes 0x01 when data is input from the $\overline{\text{RESET}}$ pin. The value of OCDM register is retained in the case of reset by the watchdog timer, clock monitor, or low-voltage detector.

Figure 4-4. OCDM Register Format

On-chip debug mode register (OCDM)
Address: 0xFFFFF9FC

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	OCDM0

OCDM0	Operation mode
0	<p>[When using MINICUBE2 (QB-MINI2) in normal operation mode]</p> <p style="color: red;">Operates in normal operation mode (the alternate-function pin to which the on-chip debug function is assigned is used as a port pin or a peripheral function pin) and disconnects the on-chip pull-down resistor of the P05/INTP2/$\overline{\text{DRST}}$ pin.</p>
1	<p>[When using MINICUBE (QB-V850MINI)]</p> <p>When the $\overline{\text{DRST}}$ pin is at low level: Normal operation mode (uses the alternate-function pin to which the on-chip debug function is assigned as a port pin or peripheral function pin)</p> <p>When the $\overline{\text{DRST}}$ pin is at high level: On-chip debug mode (the alternate-function pin to which the on-chip debug function is assigned is used as an on-chip debug mode pin)</p>

Remark The red value indicates the value set in the sample program

The data set to the OCDM special register is 0x00.

- C language

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

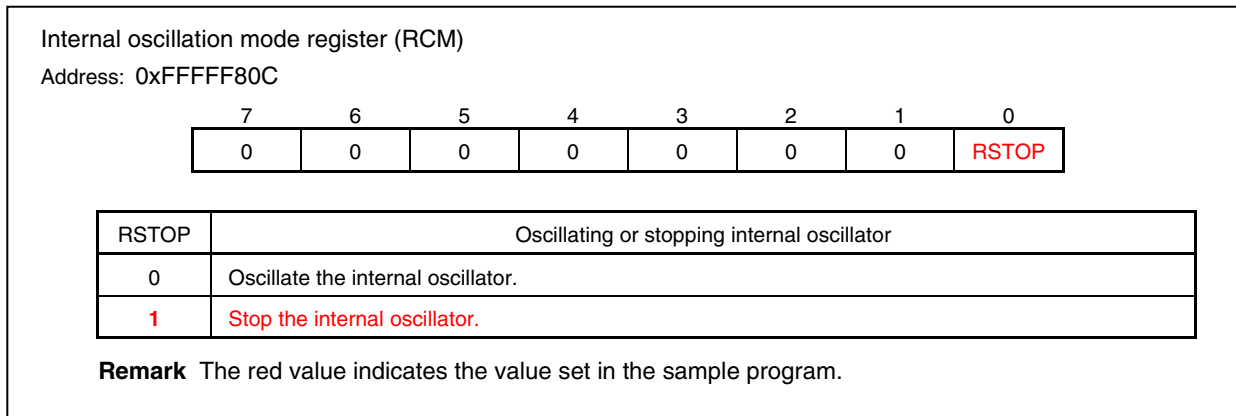
- Assembly language

```
st.b r0, PRCMD    -- Stores a dummy value to the PRCMD register for accessing OCDM.  
st.b r0, OCDM    -- Sets the OCDM register (sets to normal operation mode).
```

4.5 Setting Internal Oscillation Mode Register (RCM)

The RCM register is an 8-bit register that is used to set the operation mode of the internal oscillator. In this sample program, the internal oscillator is stopped, because the watchdog timer is not used. This register can be read or written in 8-bit or 1-bit units. Reset sets this register to 0x00.

Figure 4-5. RCM Register Format



The value set to RCM is **0x01**.

- C language

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

- Assembly language

```
setl RSTOP     -- Stop the internal oscillator.
```

4.6 Setting Watchdog Timer 2

The WDTM2 register is used to set the overflow time and operating clock of watchdog timer 2.

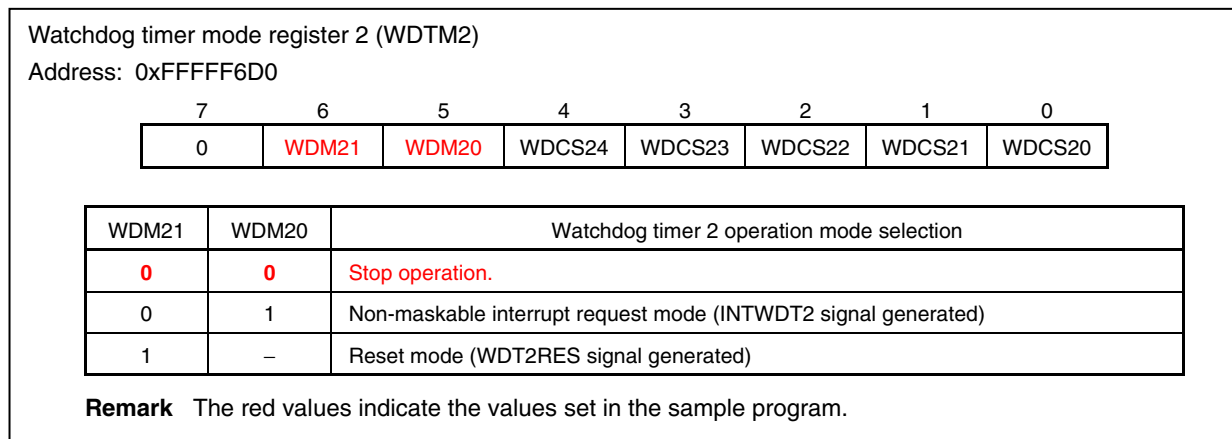
Watchdog timer 2 automatically starts in reset mode after reset is released. Write data to the WDTM2 register to specify the operation of watchdog timer 2.

In this sample program, watchdog timer 2 is stopped, because no watchdog timer is used for detecting infinite loop.

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

Reset sets this register to 0x67.

Figure 4-6. WDTM2 Register Format



The value set to WDTM2 is 0x00.

- C language

```
WDTM2 = 0b00000000; /* Stop watchdog timer 2 operation. */
```

- Assembly language

```
st.b    r0, WDTM2        -- Stop watchdog timer 2 operation.
```


4.7 Clock Setting

In this sample program, an example in which a 5 MHz ceramic or crystal resonator is connected to the X1 and X2 pins and the clock of the resonator is multiplied by 4 in PLL mode and used as the internal system clock (20 MHz) is shown. The subclock is not used.

4.7.1 Processor clock control register (PCC) setting

The PCC register is used to select the internal feedback resistor of the main clock and subclock, control the main clock oscillator, and select the internal system clock.

This register is a special register and can be written only by using a combination of specific sequences.

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

Reset sets this register to 0x03.

Figure 4-7. PCC Register Format

Processor clock control register (PCC)
Address: 0xFFFFF828

7	6	5	4	3	2	1	0
FRC	MCK	MFRC	CLS	CK3	CK2	CK1	CK0

FRC	Subclock internal feedback resistor selection
0	Use internal feedback resistor (subclock connected).
1	Do not use internal feedback resistor (subclock not connected).

MFRC	Main clock internal feedback resistor selection
0	Use internal feedback resistor (when using a ceramic or crystal resonator).
1	Does not use internal feedback resistor (when using an external clock).

CK3	CK2	CK1	CK0	Clock selection (f_{CLK}/f_{CPU})
0	0	0	0	f_{xx}
0	0	0	1	$f_{xx}/2$
0	0	1	0	$f_{xx}/4$
0	0	1	1	$f_{xx}/8$
0	1	0	0	$f_{xx}/16$
0	1	0	1	$f_{xx}/32$
0	1	1	X	Setting prohibited
1	X	X	X	f_{XT}

Remark The red values indicate the values set in the sample program.

The value set to PCC is 0x80.

- C language

```
/* Set to not divide the clock. */  
#pragma asm  
    push r10  
    mov 0x80, r10  
    st.b r10, PRCMD  
    st.b r10, PCC  
    pop r10  
#pragma endasm
```

- Assembly language

```
mov 0x80, r10    -- Sets the data to be set to the special register to the general-  
                 purpose register.  
st.b r10, PRCMD -- Stores a dummy value to the PRCMD register for accessing PCC.  
st.b r10, PCC   -- Sets the PCC register and selects main clock oscillation (fxx).
```

4.7.2 Lock register (LOCKR)

The LOCKR register is used as a flag to check whether the PLL has stabilized (has been locked).

This register is read-only, in 8-bit or 1-bit units.

Reset sets this register to 0x01. This register becomes 0x00 when the oscillation stabilization time has elapsed after reset is released.

Figure 4-8. LOCKR Register Format

Lock register (LOCKR)							
Address: 0xFFFF824							
7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	LOCK

LOCK	Operation mode
0	Locked
1	Unlocked

The LOCK bit does not reflect the lock state of PLL in real time.

Remark After reset is released and the oscillation stabilization time has elapsed, the lock register is locked (LOCKR = 0x01). When shifting to PLL mode without stopping the PLL, such as during the initial settings, **checking the lock register can be omitted.**

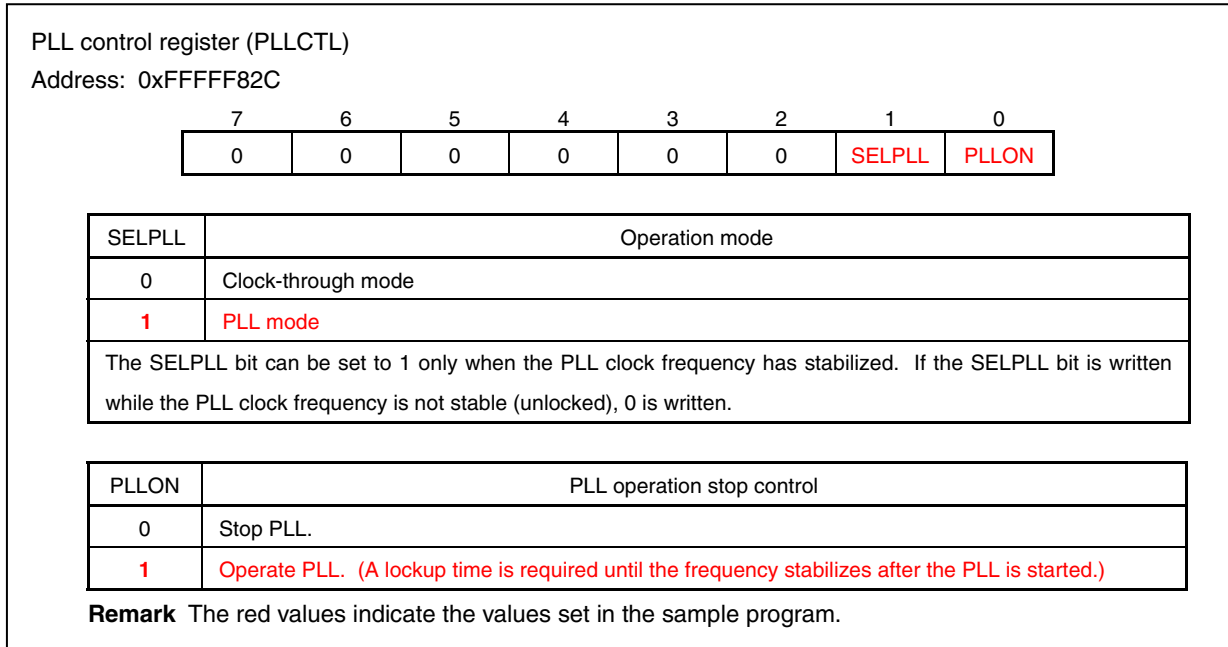
4.7.3 Setting PLL control register (PLLCTL)

The PLL control register (PLLCTL) is used to select the CPU operation clock.

This register is an 8-bit register that controls the PLL. It can be read or written in 8-bit or 1-bit units.

Reset sets this register to 0x01.

Figure 4-9. PLLCTL Register Format



Usage:

- After reset is released, the PLL operates (PLLCTL.PLLON bit = 1), but the mode must be changed to PLL mode (SELPLL bit = 1), because clock-through mode (PLLCTL.SELPLL bit = 0) is set by default.
- To operate the PLL after it has been stopped, set the PLLON bit to 1 and then set the SELPLL bit to 1 after the LOCKR.LOCK bit becomes 0. Stop the PLL (PLLON bit = 0) at least eight clocks after setting the mode to clock-through mode (SELPLL bit = 0).

- C language

```
/* CPU operation clock PLL mode: fx = 2.5 to 5 MHz (fxx = 10 to 20 MHz) selected. */  
  
    PLLON = 1;           /* Enables PLL operation. */  
  
    SELPLL = 1;         /* Sets to PLL mode. */
```

- Assembly language

```
-- CPU operation clock PLL mode: fx = 2.5 to 5 MHz (fxx = 10 to 20 MHz) selected.  
  
set1    PLLON           -- Enables PLL operation.  
  
set1    SELPLL         -- Sets to PLL mode.
```

4.8 Setting Ports

The ports to be set vary, because the on-chip ports differ for each product.

	V850ES/JF3-L	V850ES/JG3-L
Port 0	P02 to P06	P02 to P06
Port 1	P10	P10 to P11
Port 3	P30 to P35, P38, P39	P30 to P39
Port 4	P40 to P42	P40 to P42
Port 5	P50 to P55	P50 to P55
Port 7	P70 to P77	P70 to P711
Port 9	P90, P91, P96 to P99, P913 to P915	P90 to P915
Port CM	PCM0 to PCM3	PCM0 to PCM3
Port CT	PCT0, PCT1, PCT4, PCT6	PCT0, PCT1, PCT4, PCT6
Port DH	PDH0, PDH1	PDH0 to PDH5
Port DL	PDL0 to PDL15	PDL0 to PDL15

4.8.1 Port n register (Pn)

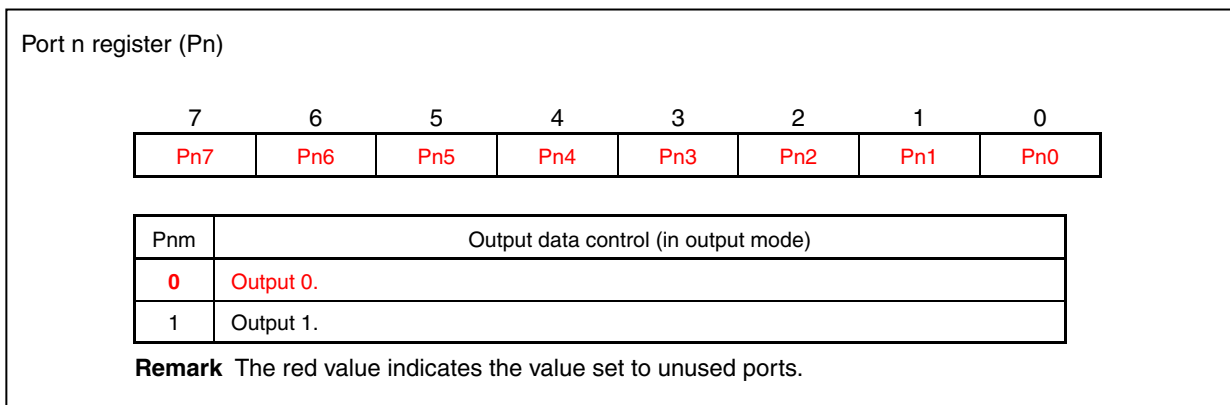
Inputting data from and outputting data to external devices is performed by writing to and reading from the Pn register. The Pn register is configured of an output latch that retains the output data and a circuit that reads the pin statuses.

Each bit of the Pn register corresponds to one pin of port n and can be read or written in 1-bit units.

In this sample program, port 0 is set as [Example 1] and port CM is set as [Example 2] described later. Unused port pins are set as output ports.

Reset sets this register to 0x00.

Figure 4-10. Pn Register Format



[Column] Handling unused pins

Port pins are set as input pins by reset. Consequently, it is recommended to connect unused pins individually to V_{DD} or GND via a resistor.

Note that unused pins set to output mode can be left open in order to reduce the number of resistors.

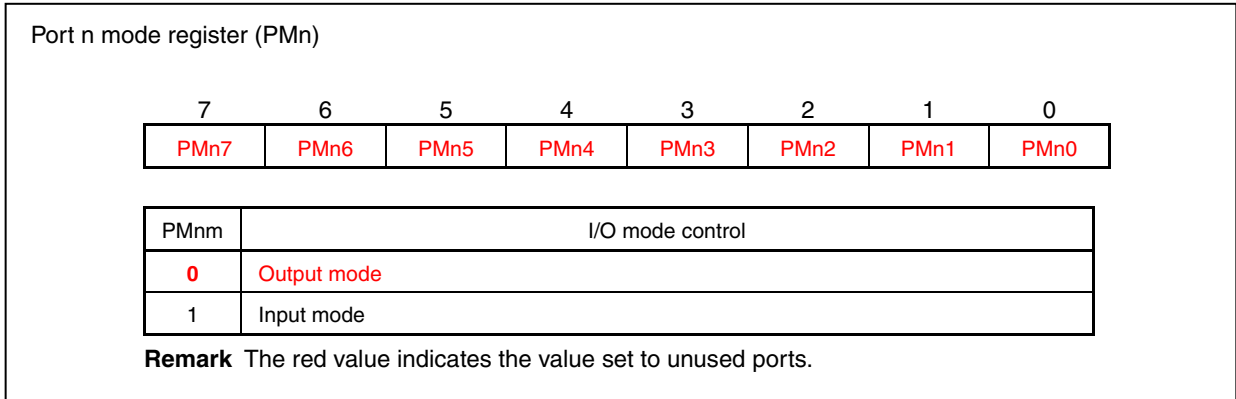
4.8.2 Port n mode register (PMn)

The PMn register is used to specify input mode or output mode for each port.

Each bit of the PMn register corresponds to one pin of port n and can be specified in 1-bit units.

Reset sets this register to 0xFF.

Figure 4-11. PMn Register Format



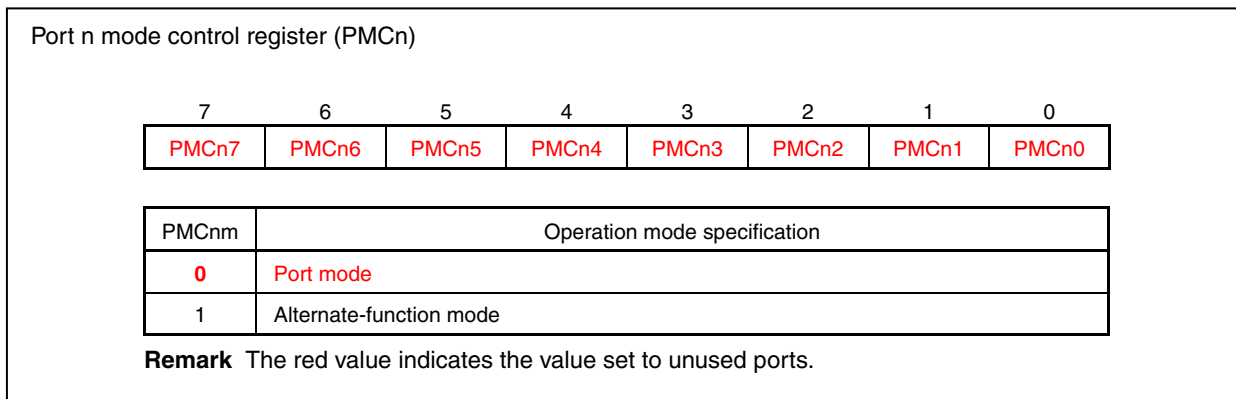
4.8.3 Port n mode control register (PMcn)

The PMcn register is used to specify port mode or alternate-function mode.

Each bit of the PMcn register corresponds to one pin of port n and can be specified in 1-bit units.

Reset sets this register to 0x00.

Figure 4-12. PMcn Register Format



[Column] Writing to and reading from the Pn register

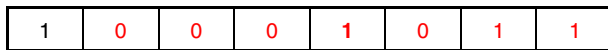
Writing to the Pn register results in writing to an output latch.

For pins set to input mode by the PMn register, the input pin status is not affected, regardless of the value written to the Pn register.

The value written to the output latch is retained until a value is written to the output latch again.

[Example 1] • Setting P03 as an input port

PM0



P03 pin I/O mode setting



The value set to PM0 is **0x8B**.

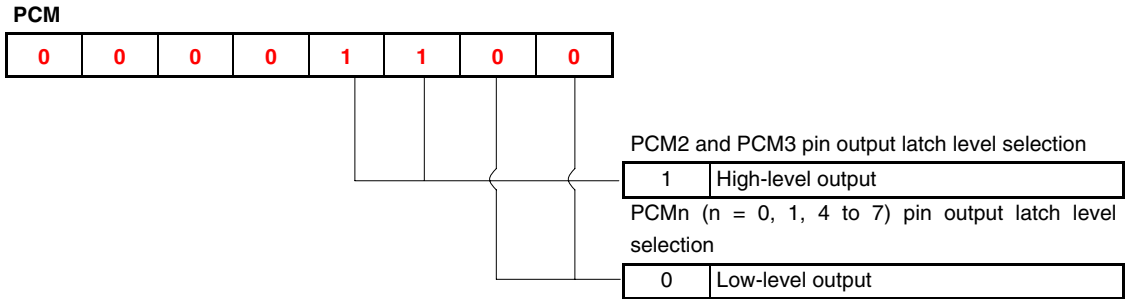
• C language

```
PM0 = 0b10001011;    /* Sets P02 to P06 to low-level output (except P03). */
```

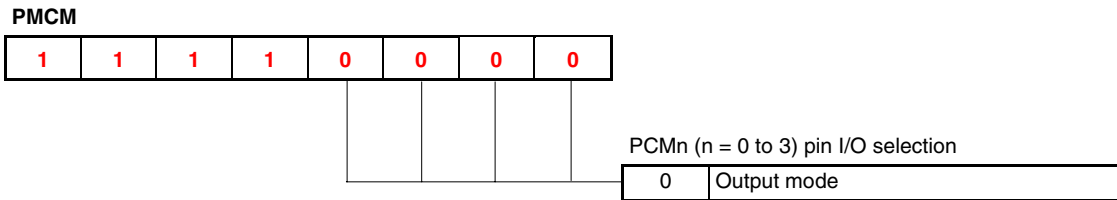
• Assembly language

```
mov 0x8b, r11        -- Sets P02 to P06 to low-level output (except P03).
st.b r11, PM0
```


- [Example 2]**
- Setting the output latches of PCM2 and PCM3 to high-level output
 - Setting PCM2 and PCM3 as output ports.



* In this sample program, PCM2 and PCM3 are used as output ports for lighting LEDs, so the output latch levels of PCM2 and PCM3 are preset to high-level output in the initial settings. (The LED lights up when a low level is output from PCM2 and PCM3 (see 2.1 **Circuit Diagram**.)



* In this sample program, PCM2 and PCM3 are used as output ports for lighting LEDs, so port CM is set as an output port.

The value set to PCM is **0x0C** and the value set to PMCM is **0xF0**.

• C language

```
PCM = 0b00001100;      /* Sets the output latches of PCM2 and PCM3 to high-level output. */
PMCM = 0b11110000;     /* Sets PCM0 to PCM3 as output ports. */
```

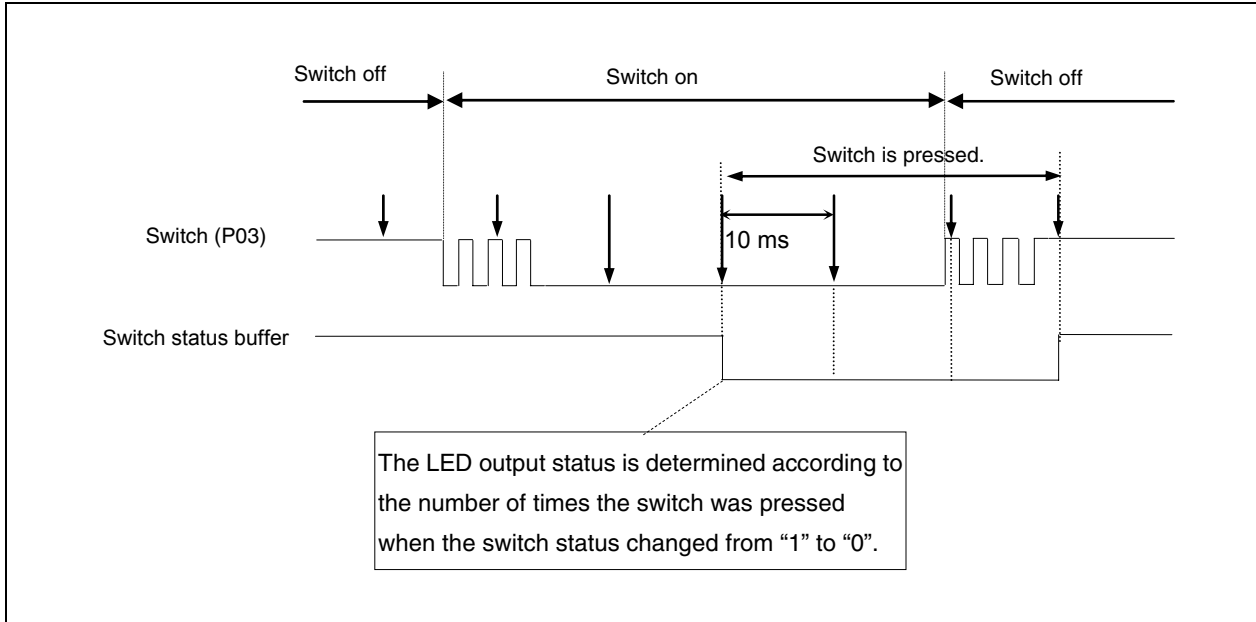
• Assembly language

```
mov 0x0c, r11          -- Sets the output latches of PCM2 and PCM3 to high-level output.
st.b r11, PCM
mov 0xF0, r11          -- Sets PCM0 to PCM3 as output ports.
st.b r11, PMCM
```

4.9 Main Processing

4.9.1 Chattering countermeasure

To eliminate chattering, a change in the switch status is determined by reading inputs every 10 ms and detecting the same level for the switch status two times in succession.



In the 10 ms wait processing, the following operation is performed.

- C language

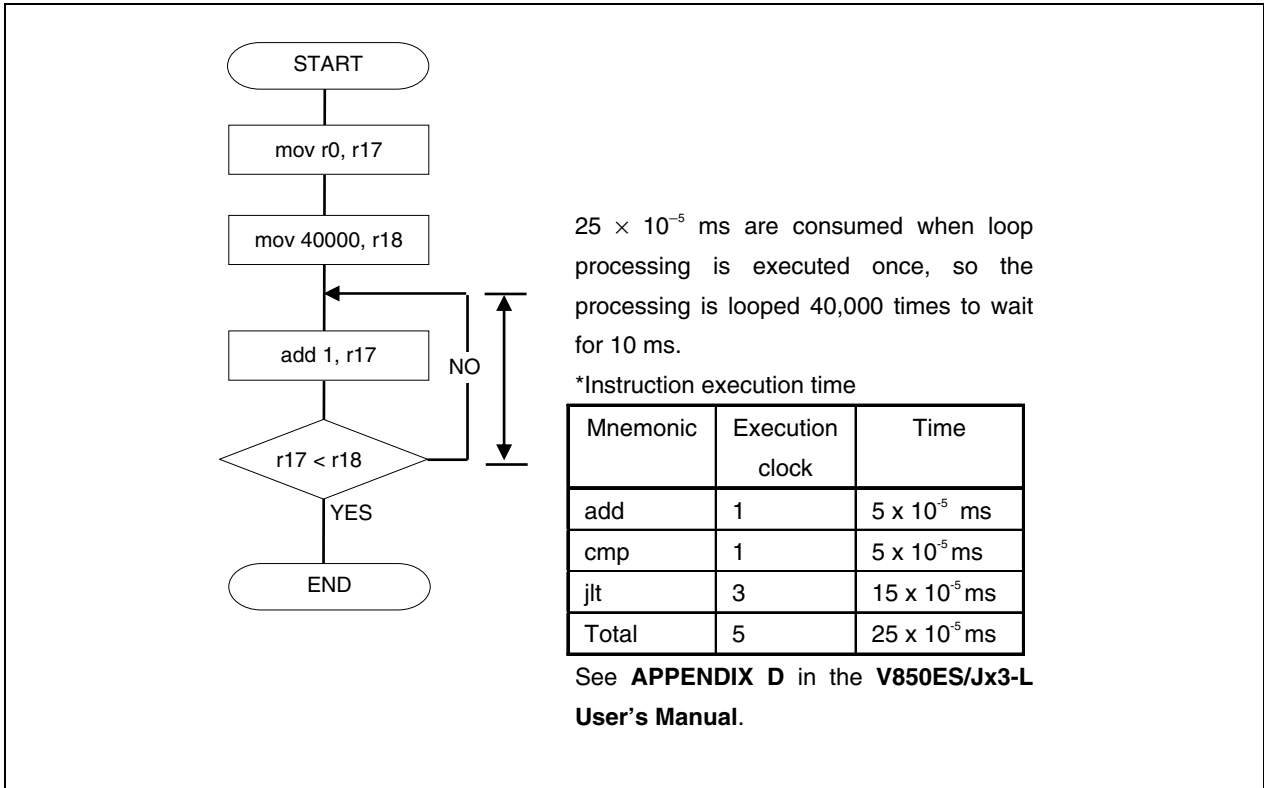
```

unsigned long loop_wait;          /* Counter for loop */

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

```

• Assembly language

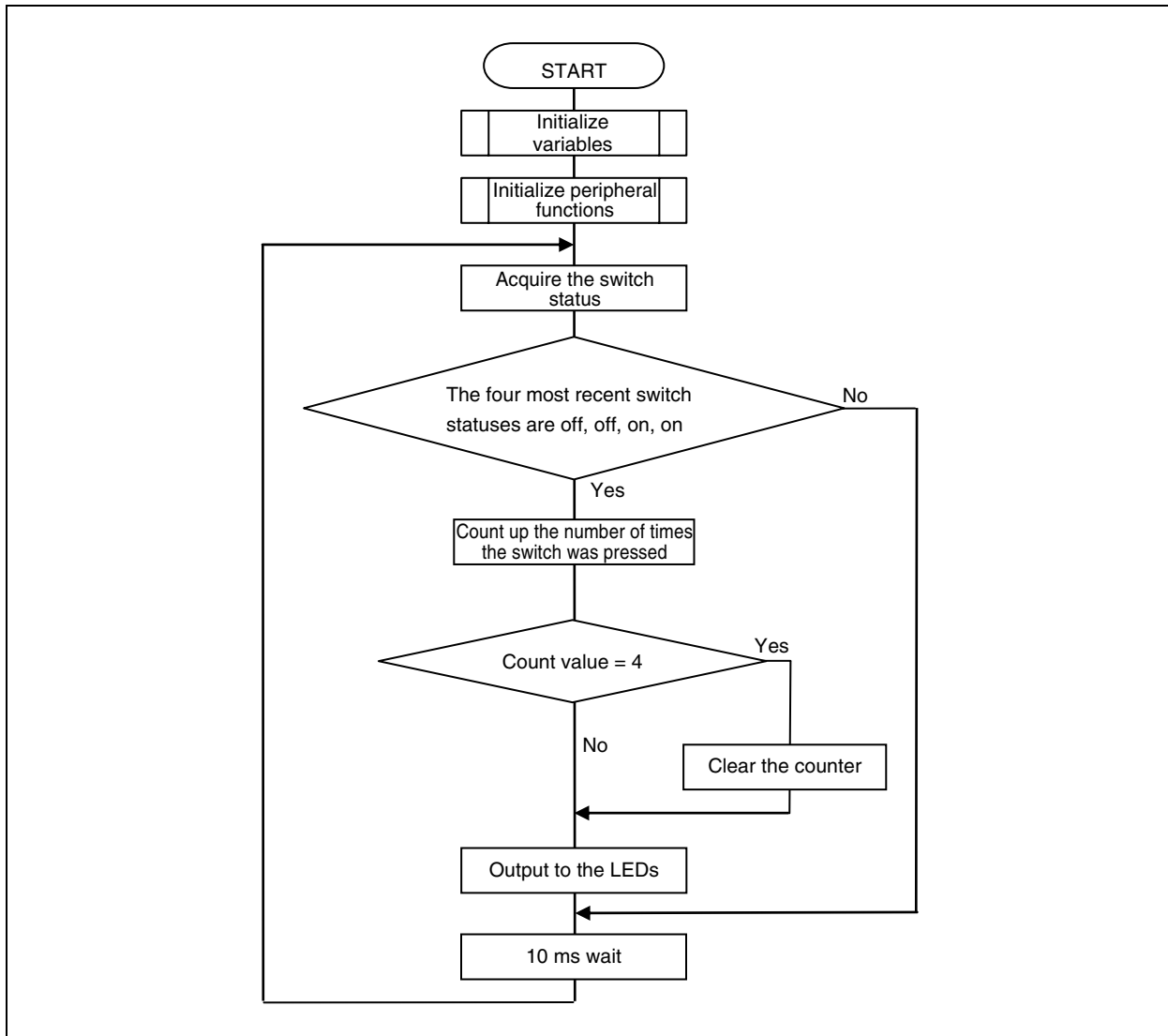


```
.wait10ms:
-- 10 ms wait
    mov    r0, r17
    mov    40000, r18
.not_equal_10ms:
    add    1, r17
    cmp    r18, r17
    jlt    .not_equal_10ms
```

4.9.2 Main processing

In the main processing in C language, the following operation is performed.

In C language, the correspondence between the input and output data is set in an array.



```

/*****
Main processing
*****/
void main( void )
{
extern unsigned int _S_romp;          /* External reference of ROMization symbol */

/*-----*/
/* Variable declaration and initial variable setting */
/*-----*/
const unsigned char outdata[] = {     /* Array for the data display pattern */
    0x0c,                               /* Turns off all LEDs. */
    0x04,                               /* Lights LED1. */
    0x00,                               /* Lights LED1 and LED2 */
    0x08                               /* Lights LED2. */
};

unsigned char indata = 0b00000001;    /* To memorize the pressed status of the switch
(initialized if the previous value is "off") */
unsigned char count;                  /* Number of times the switch was pressed */
unsigned long loop_wait;              /* Counter for loop */

count = VAL_RST_COUNT;               /* Initializes the number of times the switch was pressed */

/*-----*/
/* Peripheral-function initialization */
/*-----*/
f_init_vswc();                        /* Sets the VSWC register */
f_init_ocdm();                        /* Sets on-chip debugging to normal operation
mode */
f_init_rcm();                          /* Disables the internal oscillator */
f_init_wdtm2();                        /* Sets watchdog timer 2 */
f_init_lock();                         /* Sets the CPU operation clock to PLL mode */

f_init_blank_port();                  /* Sets unused ports */
f_init_use_port();                    /* Sets the SW1 and LED ports */

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

/*-----*/
/* LED lighting processing */
/*-----*/
while ( 1 )
{
    indata <<= 1;                      /* Updates the previous switch status value */
    indata |= P0.3;                    /* Updates the current switch status value */
}

```

Four units of data are defined in the braces, within the output data is set.

```
if ( ( indata & 0b00001111 ) == 0b00001100 )
{
    count++;          /* Updates the number of times the switch was pressed */
    count &= 0b00000011
    PCM = outdata[count];    /* Displays the display data read from the table */;
}

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

return;
}
```

The correspondence between the input and output data is shown below.

Switch input count	COUNT	OUTDATA	LED lighting
0	0	0b00001100	All LEDs turned off.
1	1	0b00000100	Only LED1 lights.
2	2	0b00000000	LED1 and LED2 light.
3	3	0b00001000	Only LED 2 lights.

In the main processing in assembly language, the same operation as that in C language is performed.

- <1> Set the previous switch status to bit 1 of the switch status memory register (r28).
- <2> Read P03 data and set the latest switch status to bit 0 of the switch status memory register (r28).
- <3> Compare the most recent switch status (r28) with 0xC (status in which “on” is detected two times in succession after “off” is detected two times in succession).
- <4> If the comparison in step <3> results in a match, change the LED output according to the number of times the switch was pressed and proceed to step <6>.
- <5> If the comparison in <3> does not result in a match, proceed to step <6>.
- <6> Wait for 10 ms.
- <7> Return to step <1>.

.data	-- Data section	ROM area setting
.align 4		
data_area:		
.byte 0x0C	-- 0 times, 4 times	--> Turns off all LEDs
.byte 0x04	-- 1 time	--> Lights LED1.
.byte 0x00	-- 2 times	--> Lights LED1 and LED2.
.byte 0x08	-- 3 times	--> Lights LED2.

```

.globl _main
_main:
-- Peripheral-function initialization
jarl    _init, lp

-- Variable initialization
mov     1, r28          -- indata  bit0: Current value /bit1: Previous value
mov     r0, r29         -- count   Number of times the switch was pressed

.main_loop:
-- LED lighting processing
-- Update of indata
shl     1, r28          -- indata.1 = indata.0
tstl    3, P0          -- Acquires the switch input value
setfnz  r14
or      r14, r28        -- indata.0 = P03

.sw_on_chk:
-- Switch-on check
andi    0xF, r28, r17
cmp     0xC, r17        -- Proceed to switch-on detection if switch-on is detected two times in succession
jne     .wait10ms      -- Proceed to a 10 ms wait if switch-on is not detected
-- Switch-on detection
-- Update of count
add     1, r29          -- count = count + 1
and     3, r29
-- Update of LED lighting status
mov     r29, r15
add     #data_area, r15
ld.b    [r15], r16      -- r16 = PCM port output value
st.b    r16, PCM        -- Output to the PCM ports (LEDs).

.wait10ms:
-- 10 ms wait
mov     r0, r17
mov     40000, r18

.not_equal_10ms:
add     1, r17
cmp     r18, r17
jlt     .not_equal_10ms

jr     .main_loop

```


CHAPTER 5 RELATED DOCUMENTS

Document	English
V850ES/JF3-L Hardware User's Manual	PDF
V850ES/JG3-L Hardware 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
#-----
#   LED lighting switch control
#-----
# [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
#-----
#   LED lighting switch control
#-----
#[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_LED.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;
};

```

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 (C language version)
/*-----*/
/*
/* NEC Electronics      V850ES/Jx3-L series
/*
/*-----*/
/* V850ES/JG3-L sample program
/*-----*/
/* LED lighting switch control
/*-----*/
/* [History]
/* 2008.07.-- Released
/*-----*/
/* [Overview]
/* This sample program selects the clock frequency, sets the port I/Os, and performs
/* the basic initial settings of the V850ES/JG3-L microcontroller.
/* The main processing operation performed after the completion of the initial
/* settings controls the lighting of two LEDs by using one switch input.
/*
/* Of the peripheral functions that are stopped after reset is released, those that
/* are not used in this sample program are not set.
/*
/*
/* <Main contents of initial settings>
/* • Setting the system wait control register to one clock
/* • Setting on-chip debugging to normal operation mode
/* • Stopping the internal oscillator
/* • Stopping watchdog timer 2 operation
/* • Setting the system clock to 20 MHz by multiplying the input clock by 4 using the
PLL
/* • Setting unused ports
/* • Setting the switch input and LED control ports /*
/*
/* <Main contents of main processing>
/* • Detecting the number of switch inputs
/* • Lighting the LEDs
/*

```

```

/* <Switch input and LED lighting>
/*
/* +-----+
/* | Number of times the switch is pressed | LED1 | LED2 |
/* | (P03) | (PCM3) | (PCM2) |
/* |-----|-----|-----|
/* | 0 times | OFF | OFF |
/* | 1 time | ON | OFF |
/* | 2 times | ON | ON |
/* | 3 times | OFF | ON |
/* +-----+
/* *Inputs 0 to 3 are repeated from the fourth input.
/*
/*
/* [I/O port settings]
/*
/* • Input port : P03
/* • Output ports : PCM2, PCM3
/* • Unused ports : P02, P04 to P06, P10 and P11, P3H0 and P3H1, P3L0 to P3L7,
/* P40 to P42, P50 to P55, P7H0 to P7H3, P7L0 to P7L7, P9H0 to P9H7,
/* P9L0 to P9L7, PCM0 and PCM1, PCT0,1,4,6, PDH0 to PDH5,
/* PDLH0 to PDLH7, PDLL0 to PDLL7
/* *Preset all unused ports as output ports (low-level output).
/*
/*-----*/

/*-----*/
/* pragma directives */
/*-----*/
#pragma ioreg /* Specifies enabling the names of the peripheral
I/O registers. */

/*-----*/
/* Constant definitions */
/*-----*/
#define VAL_RST_COUNT (0) /* Initial value of the number of times the switch
was pressed */
#define VAL_TIMER_WAIT (28700) /* 10 ms wait */

```

```

/*-----*/
/* Prototype declarations      */
/*-----*/
static void f_init_vswc( void );      /* VSWC register setting processing */
static void f_init_ocdm( void );      /* On-chip debugging normal operation
mode setting processing */
static void f_init_rcm( void );       /* Internal oscillator stop setting */
static void f_init_wdtm2( void );     /* Watchdog timer 2 setting processing */
static void f_init_lock( void );      /* CPU operation clock setting processing */
static void f_init_blank_port( void ); /* Unused port setting initialization
processing */
static void f_init_use_port( void );   /* SW1 and LED port setting initialization
processing */
void main( void );                   /* Main processing */

/*****/
/* Initial settings of peripheral functions */
/*****/
/*-----*/
/* Setting the VSWC register */
/*-----*/
static void f_init_vswc( void )
{
    VSWC = 0b00000001;                /* Inserts one wait cycle when the on-chip
peripheral I/O register is accessed. */

    return;
}

/*-----*/
/* Setting on-chip debugging to normal operation mode */
/*-----*/
static void f_init_ocdm( void )
{
    /* Specifies normal operation mode for OCDM. */

#pragma asm
    st.b    r0, PRCMD
    st.b    r0, OCDM
#pragma endasm

    return;
}

```



```
/*-----*/
/* Setting the internal oscillation mode register (RCM) */
/*-----*/
static void f_init_rcm( void )
{
    RSTOP = 1;                /* Stops the internal oscillator. */

    return;
}

/*-----*/
/* Setting watchdog timer 2 (WDTM2) */
/*-----*/
static void f_init_wdtm2( void )
{
    WDTM2 = 0b00000000;      /* Stops watchdog timer 2 operation. */

    return;
}

/*-----*/
/* Setting the CPU operation clock to PLL mode */
/*-----*/
static void f_init_lock( void )
{
    /* Setting to PLL mode and setting PLL operation (PLLCTL register setting) */
    /* Selecting CPU operation clock PLL mode: fx = 2.5 to 5 MHz (fxx = 10 to 20 MHz) */
    PLLON = 1;                /* Enables PLL operation. */
    SELPLL = 1;                /* Sets to PLL mode. */

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

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

    return;
}
```

```

}
/*-----*/
/* Setting unused ports */
/*-----*/
static void f_init_blank_port( void )
{
    P0 = 0b00000000;          /* Sets P02 to P06 to low-level output (except P03)
*/
    PM0 = 0b10001011;

    P1 = 0b00000000;          /* Sets P10 and P11 to low-level output.          */
    PM1 = 0b11111100;

    P1 = 0b00000000;          /* Sets P10 to low-level output.          */
    PM1 = 0b11111110;

    P3H = 0b00000000;          /* Sets P3H0 and P3H1 to low-level output.          */
    PM3H = 0b11111100;
    P3L = 0b00000000;          /* Sets P3L0 to P3L7 to low-level output.          */
    PM3L = 0b00000000;

    P3H = 0b00000000;          /* Sets P3H0 and P3H1 to low-level output.          */
    PM3H = 0b11111100;
    P3L = 0b00000000;          /* Set P3L0 to P3L5 to low-level output.          */
    PM3L = 0b11000000;

    P4 = 0b00000000;          /* Sets P40 to P42 to low-level output.          */
    PM4 = 0b11111000;

    P5 = 0b00000000;          /* Sets P50 to P55 to low-level output.          */
    PM5 = 0b11000000;

    P7H = 0b00000000;          /* Sets P7H0 to P7H3 to low-level output.          */
    PM7H = 0b11100000;
    P7L = 0b00000000;          /* Sets P7L0 to P7L7 to low-level output.          */
    PM7L = 0b00000000;

    P7L = 0b00000000;          /* Sets P7L0 to P7L7 to low-level output.          */
    PM7L = 0b00000000;

```

In the V850ES/JF3-L, sets only P10 as P1.

In the V850ES/JF3-L, sets only P3H0, P3H1, and P3L0 to P3L5 as P3.

In the V850ES/JF3-L, sets only P7L0 to P7L7 as P7.

APPENDIX A PROGRAM LIST

```
P9H = 0b00000000; /* Sets P9H0 to P9H7 to low-level output. */
PM9H = 0b00000000;
P9L = 0b00000000; /* Sets P9L0 to P9L7 to low-level output. */
PM9L = 0b00000000;

/* In the V850ES/JF3-L, sets only P9H0, 1, 5, 6, 7 and P9L0, 1, 6, 7 as P9. */
P9H = 0b00000000; /* Sets P9H0, P9H1, and P9H5 to P9H7 to low-level output. */
PM9H = 0b00011100;
P9L = 0b00000000; /* Sets P9L0, P9L1, P9L6, and P9L7 to low-level output. */
PM9L = 0b00111100;

PCM = 0b00000000; /* Sets PCM0 and PCM1 to low-level output. */
PMCM = 0b11111100;

PCT = 0b00000000; /* Sets PCT0, 1, 4, and 6 to low-level output. */
PMCT = 0b10101100;

PDH = 0b00000000; /* Sets PDH0 to PDH5 to low-level output. */
PMDH = 0b11000000;

/* In the V850ES/JF3-L, sets only PDH0 and PDH1 as PDH. */
PDH = 0b00000000; /* Sets PDH0 and PDH1 to low-level output. */
PMDH = 0b11111100;

PDLH = 0b00000000; /* Sets PDLH0 to PDLH7 to low-level output. */
PMDLH = 0b00000000;
PDLL = 0b00000000; /* Sets PDLL0 to PDLL7 to low-level output. */
PMDLL = 0b00000000;

return;
}
```

```

/*-----*/
/* Setting the switch input and LED control ports */
/*-----*/
/static void f_init_use_port( void )
{
    /* Setting the switch output port */
    P0 = 0b00000000;          /* Sets P03 as an input.          */
    PM0 = 0b10001011;

    /* Setting the LED output port */
    PCM = 0b00001100;        /* Sets PCM2 and PCM3 to high-level output. */
    PMCM = 0b11110000;

    return;
}

/*****/
/* Main module      */
/*****/
void main( void )
{
    extern unsigned int _S_romp; /* External reference of ROMization symbol*/

    /*-----*/
    /* Variable declaration and initial variable setting */
    /*-----*/
    const unsigned char outdata[] = { /* Array for the data display pattern */
        0x0c, /* Turns off all LEDs. */
        0x04, /* Lights LED1. */
        0x00, /* Lights LED1 and LED2. */
        0x08 /* Lights LED2. */
    };
    unsigned char indata = 0b00000001; /* To memorize the pressed status of the switch
                                         (initialized if the previous value is "off") */
    unsigned char count; /* Number of times the switch was pressed */
    unsigned long loop_wait; /* Counter for loop */

    count = VAL_RST_COUNT; /* Initializes the number of times the switch was
                             pressed */
}

```

```

/*-----*/
/* Peripheral-function initialization */
/*-----*/
f_init_vswc();          /* Sets the VSWC register */
f_init_ocdm();         /* Sets on-chip debugging to normal operation mode */
f_init_rcm();          /* Disables the internal oscillator */
f_init_wdtm2();        /* Sets watchdog timer 2 */
f_init_lock();         /* Sets the CPU operation clock to PLL mode */
f_init_blank_port();   /* Sets unused ports */
f_init_use_port();     /* Sets the SW input and LED control ports */

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

/*-----*/
/* LED lighting processing */
/*-----*/
while ( 1 )
{
    indata <<= 1;          /* Updates the previous switch status value */
    indata |= P0.3;       /* Updates the current switch status value */
    if ( ( indata & 0b00001111 ) == 0b00001100 )
    {
        count++;          /* Updates the number of times the switch was pressed */
        count &= 0b00000011;
        PCM = outdata[count]; /* Displays the display data read from the table */
    }

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

return;
}

```

```

● main.s (assembly language version)
#-----
#
# NEC Electronics      V850ES/Jx3-L series
#
#-----
# V850ES/JG3-L sample program
#-----
# LED lighting switch control
#*-----
# [History]
# 2008.07.-- Released
#-----
# [Overview]
# This sample program selects the clock frequency, sets the port I/Os, and performs
# the basic initial settings of the V850ES/JG3-L microcontroller.
# The main processing operation performed after the completion of the initial
# settings controls the lighting of two LEDs by using one switch input.
#
# Of the peripheral functions that are stopped after reset is released, those that
# are not used in this sample program are not set.
#
#
#
# <Main contents of initial settings>
# • Setting the system wait control register to one clock
# • Setting on-chip debugging to normal operation mode
# • Stopping the internal oscillator
# • Stopping watchdog timer 2 operation
# • Setting the system clock to 20 MHz by multiplying the input clock by 4 using the
PLL
# • Setting unused ports
# • Setting the switch input and LED control ports
#
# <Main contents of main processing>
# • Detecting the number of switch inputs
# • Lighting the LEDs
#

```

```

/* <Switch input and LED lighting>
/*
/* +-----+
/* | Number of times the switch is pressed | LED1 | LED2 |
/* | (P03) | (PCM3) | (PCM2) |
/* |-----|-----|-----|
/* | 0 times | OFF | OFF |
/* | 1 time | ON | OFF |
/* | 2 times | ON | ON |
/* | 3 times | OFF | ON |
/* +-----+
/* *Inputs 0 to 3 are repeated from the fourth input.
/*
/*
/*[I/O port settings]
/*
/* • Input port : P03
/* • Output ports : PCM2, PCM3
/* • Unused ports : P02, P04 to P06, P10 and P11, P3H0 and P3H1, P3L0 to P3L7,
/* P40 to P42, P50 to P55, P7H0 to P7H3, P7L0 to P7L7, P9H0 to P9H7,
/* P9L0 to P9L7, PCM0 and PCM1, PCT0,1,4,6, PDH0 to PDH5,
/* PDLH0 to PDLH7, PDLL0 to PDLL7
/* *Preset all unused ports as output ports (low-level output).
/*
/*-----

#-----#
# Defining the display data pattern array and ROM #
#-----#
.data -- Data section
    .align 4
data_area:
    .byte 0x0C -- 0 times, 4 times --> Turns off all LEDs
    .byte 0x04 -- 1 time --> Lights LED1.
    .byte 0x00 -- 2 times --> Lights LED1 and LED2.
    .byte 0x08 -- 3 times --> Lights LED2.

#-----#
# Initial settings of peripheral functions to be used #
#-----#
.text
_init:

```

```

/*-----*/
/* Setting the VSWC register */
/*-----*/
    mov    0x01, r11    -- Inserts one wait cycle when the on-chip peripheral I/O
                        register is accessed.

    st.b   r11, VSWC

#-----#
# Setting on-chip debugging to normal operation mode #
#-----#
    st.b   r0, PRCMD    -- Stores a dummy value to the PRCMD register for accessing
                        OCDM.

    st.b   r0, OCDM     -- Sets the OCDM register (to normal operation mode).

#-----#
# Setting the internal oscillation mode register (RCM) #
#-----#
    setl   RSTOP        -- Stops the internal oscillator.

#-----#
# Setting watchdog timer 2 (WDTM2) #
#-----#
    st.b   r0, WDTM2    -- Stops watchdog timer 2 operation.

#-----#
# Setting the CPU operation clock to PLL mode #
#-----#
    -- [Setting to PLL mode and setting PLL operation (PLLCTL register setting)]
    -- Selecting CPU operation clock PLL mode: fx = 2.5 to 5 MHz (fxx = 10 to 20 MHz)
    setl   PLLON        -- Enables PLL operation.
    setl   SELPLL       -- Sets to PLL mode.

-- [Setting the PCC register]
    mov    0x80, r10     -- Sets the data to be set to the special register to a
                        general-purpose register.

    st.b   r10, PRCMD   -- Stores a dummy value to the PRCMD register for accessing
                        PCC.

    st.b   r10, PCC     -- Sets the PCC register and selects main clock oscillation
                        (fxx).

#-----#
# Setting unused ports #
#-----#

```



```
-- [Setting port 0]
   mov    0x00, r11          -- Sets P02 to P06 to low-level output (except P03)
   st.b   r11, P0
   mov    0x8b, r11
   st.b   r11, PM0
```

```
-- [Setting port 1]
   mov    0x00, r11          -- Sets P10 and P11 to low-level output.
   st.b   r11, P1
   mov    0xfc, r11
   st.b   r11, PM1
```

In the V850ES/JF3-L, sets only P10 as P1.

```
-- [Setting port 1]
   mov    0x00, r11          -- Sets P10 to low-level output.
   st.b   r11, P1
   mov    0xfc, r11
   st.b   r11, PM1
```

```
-- [Setting port 3]
   mov    0x00, r11          -- Sets P3H0 and P3H1 to low-level output.
   st.b   r11, P3H
   mov    0xfc, r11
   st.b   r11, PM3H
```

```
   mov    0x00, r11          -- Sets P3L0 to P3L7 to low-level output.
   st.b   r11, P3L
   mov    0x00, r11
   st.b   r11, PM3L
```

In the V850ES/JF3-L, sets only P3H0, P3H1, and P3L0 to P3L5 as P3.

```
--[Setting port 3]
   mov    0x00, r11          -- Sets P3H0 and P3H1 to low-level output.
   st.b   r11, P3H
   mov    0xfc, r11
   st.b   r11, PM3H

   mov    0x00, r11          -- Sets P3L0 to P3L5 to low-level output.
   st.b   r11, P3L
   mov    0xc0, r11
   st.b   r11, PM3L
```

```
--[Setting port 4]
  mov    0x00, r11          -- Sets P40 to P42 to low-level output.
  st.b   r11, P4
  mov    0xfc, r11
  st.b   r11, PM4

--[Setting port 5]
  mov    0x00, r11          -- Sets P50 to P55 to low-level output.
  st.b   r11, P5
  mov    0xc0, r11
  st.b   r11, PM5

-- [Setting port 7]
  mov    0x00, r11          -- Sets P7H0 to P7H3 to low-level output.
  st.b   r11, P7H
  mov    0xf0, r11
  st.b   r11, PM7H

  mov    0x00, r11          -- Sets P7L0 to P7L7 to low-level output.
  st.b   r11, P7L
  mov    0x00, r11
  st.b   r11, PM7L
```

In the V850ES/JF3-L, sets only P7L0 to P7L7 as P7.

```
--[Setting port 7]
  mov    0x00, r11          -- Sets P7L0 to P7L7 to low-level output.
  st.b   r11, P7L
  mov    0x00, r11
  st.b   r11, PM7L
```

```
--[Setting port 9]
  mov    0x00, r11      -- Sets P9H0 to P9H7 to low-level output.
  st.b   r11, P9H
  mov    0x00, r11
  st.b   r11, PM9H

  mov    0x00, r11      -- Sets P9L0 to P9L7 to low-level output.
  st.b   r11, P9L
  mov    0x00, r11
  st.b   r11, PM9L
```

In the V850ES/JF3-L, sets only P9H0, 1, 5, 6, 7 and P9L0, 1, 6, 7 as P9.

```
--[Setting port 9]
  mov    0x00, r11      -- Sets P9H0, P9H1, and P9H5 to P9H7 to low-level output.
  st.b   r11, P9H
  mov    0x1c, r11
  st.b   r11, PM9H

  mov    0x00, r11      -- Sets P9L0, P9L1, P9L6, and P9L7 to low-level output.
  st.b   r11, P9L
  mov    0x3c, r11
  st.b   r11, PM9L
```

```
--[Setting port CM]
  mov    0x00, r11      -- Sets PCM0 and PCM1 to low-level output.
  st.b   r11, PCM
  mov    0xfc, r11
  st.b   r11, PMCM
```

```
--[Setting port CT]
  mov    0x00, r11      -- Sets PCT0, 1, 4, and 6 to low-level output.
  st.b   r11, PCT
  mov    0xac, r11
  st.b   r11, PMCT
```

```
--[Setting port DH]
mov    0x00, r11          -- Sets PDH0 to PDH5 to low-level output.
st.b   r11, PDH
mov    0xc0, r11
st.b   r11, PMDH
```

In the V850ES/JF3-L, sets only PDH0 and PDH1 as PDH.

```
--[Setting port DH]
mov    0x00, r11          -- Sets PDH0 and PDH1 to low-level output.
st.b   r11, PDH
mov    0xfc, r11
st.b   r11, PMDH
```

```
--[Setting port DL]
mov    0x00, r11          -- Sets PDLH0 to PDLH7 to low-level output.
st.b   r11, PDLH
mov    0x00, r11
st.b   r11, PMDLH
```

```
mov    0x00, r11          -- Sets PDLL0 to PDLL7 to low-level output.
st.b   r11, PDLL
mov    0x00, r11
st.b   r11, PMDLL
```

```
#-----#
# Setting the switch input and LED control ports #
#-----#
```

```
--[Setting port 0]
mov    0x00, r11          -- Sets P03 as an input.
st.b   r11, P0
mov    0x8b, r11
st.b   r11, PM0
```

```
--[Setting port CM]
mov    0x0c, r11          -- Sets PCM2 and PCM3 to high-level output.
st.b   r11, PCM
mov    0xf0, r11
st.b   r11, PMCM
```

```
jmp [lp]
```

```

#-----#
# Main processing                                     #
#-----#
    .globl _main
_main:
    -- Peripheral-function initialization
    jarl    _init, lp

    -- Variable initialization
    mov     1, r28          -- indata  bit0: Current value /bit1: Previous value/bit2: Second
                           most recent value/bit3: Third most recent value
    mov     r0, r29        -- count    Number of times the switch was pressed

.main_loop:
    -- LED lighting processing
    -- Update of indata
    shl     1, r28          -- indata.1 = indata.0
    tstl    3, P0          -- Acquires the switch input value
    setfnz  r14
    or      r14, r28       -- indata.0 = P03

.sw_on_chk:
    -- Switch-on check
    andi    0xF, r28, r17
    cmp     0xC, r17       -- Proceed to switch-on detection if switch-on is detected two times
                           in succession
    jne     .wait10ms     -- Proceed to a 10 ms wait if switch-on is not detected
    -- Switch-on detection
    -- Update of count
    add     1, r29         -- count = count + 1
    and     3, r29
    -- Update of LED lighting status
    mov     r29, r15
    add     #data_area, r15
    ld.b    [r15], r16     -- r16 = PCM port output value
    st.b    r16, PCM       -- Output to the PCM ports (LEDs).

.wait10ms:
    -- 10 ms wait
    mov     r0, r17
    mov     40000, r18

.not_equal_10ms:
    add     1, r17
    cmp     r18, r17
    jlt     .not_equal_10ms

    jr     .main_loop

```

*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 Esplandiú, 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/>