RENESAS

Application Note

V850ES/Jx3

Sample Program (Initial Settings)

LED Lighting Switch Control

This document summarizes the initial settings for the sample program of the V850ES/Jx3 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/JG3 microcontroller
V850ES/JJ3 microcontroller

CONTENTS

Document No. U19837EJ1V0AN00
Date Published June 2009 NS

CHAPTER 1 OVERVIEW

In this sample program, the basic initial settings for the V850ES/Jx3 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

<Main contents of initial settings>

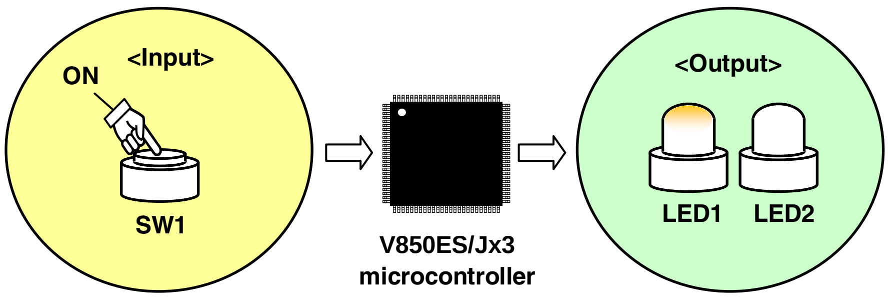
- Setting the system wait control register to two clock
- Setting on-chip debug mode register to normal operation mode
- Stopping the internal oscillator
- Stopping watchdog timer 2 operation
- Setting the system clock to 32 MHz by multiplying the input clock by 8 using the PLL
- Setting unused ports
- Setting the switch input and LED control ports

< ROMization >

- ROMization processing (initialization of variables with initial values)

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 microcontroller.



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	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) for cautions when using the device.

[Column] What is chattering?

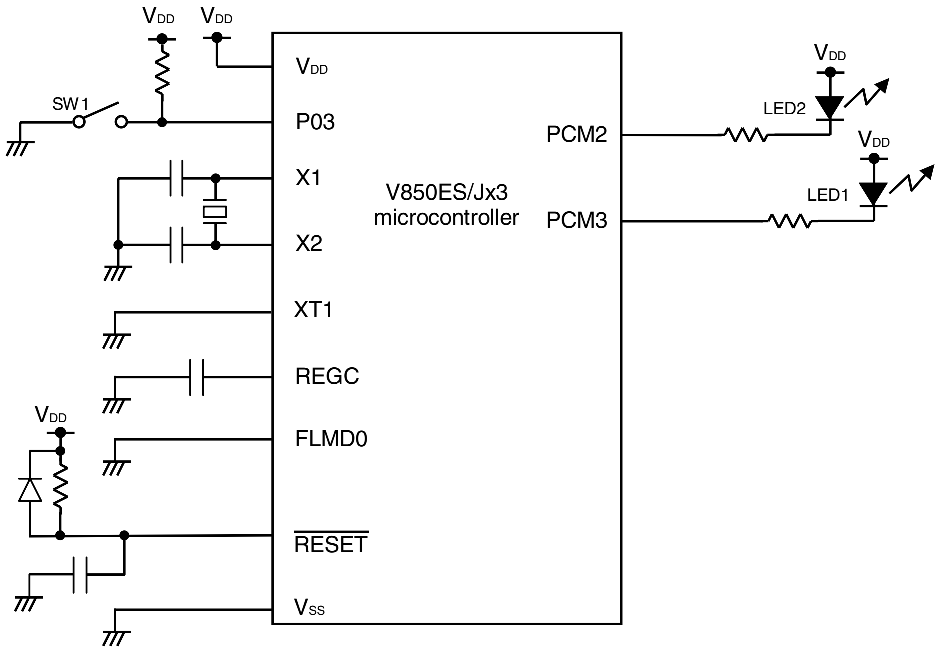
Chattering is a phenomenon that an electric signal alternates between being on and off when a connection flip-flops mechanically immediately after a switch is switched.

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.



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.

6. See the following user's manuals for circuit example with on-chip debug.

•V850ES/JJ3 32-bit Single-Chip Microcontrollers
Hardware User's Manual
•V850ES/JG3 32-bit Single-Chip Microcontrollers
Hardware User's Manual
•QB-MINI2 On-Chip Debug Emulator with Programming Function
User's Manual

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.

File Name (Tree Structure)		Description	Compressed (*.zip) Files Included	
			ZIP	PM 32
conf	crtE.s	Startup routine file[Note 1]	–	●
	AppNote_LED.dir	Link directive file[Note 2]	●	●
	AppNote_LED.prj	Project file for integrated development environment PM+	–	●
	AppNote_LED.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 ZIP : Only the source file is included.

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

3.2 On-Chip Peripheral Functions Used

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

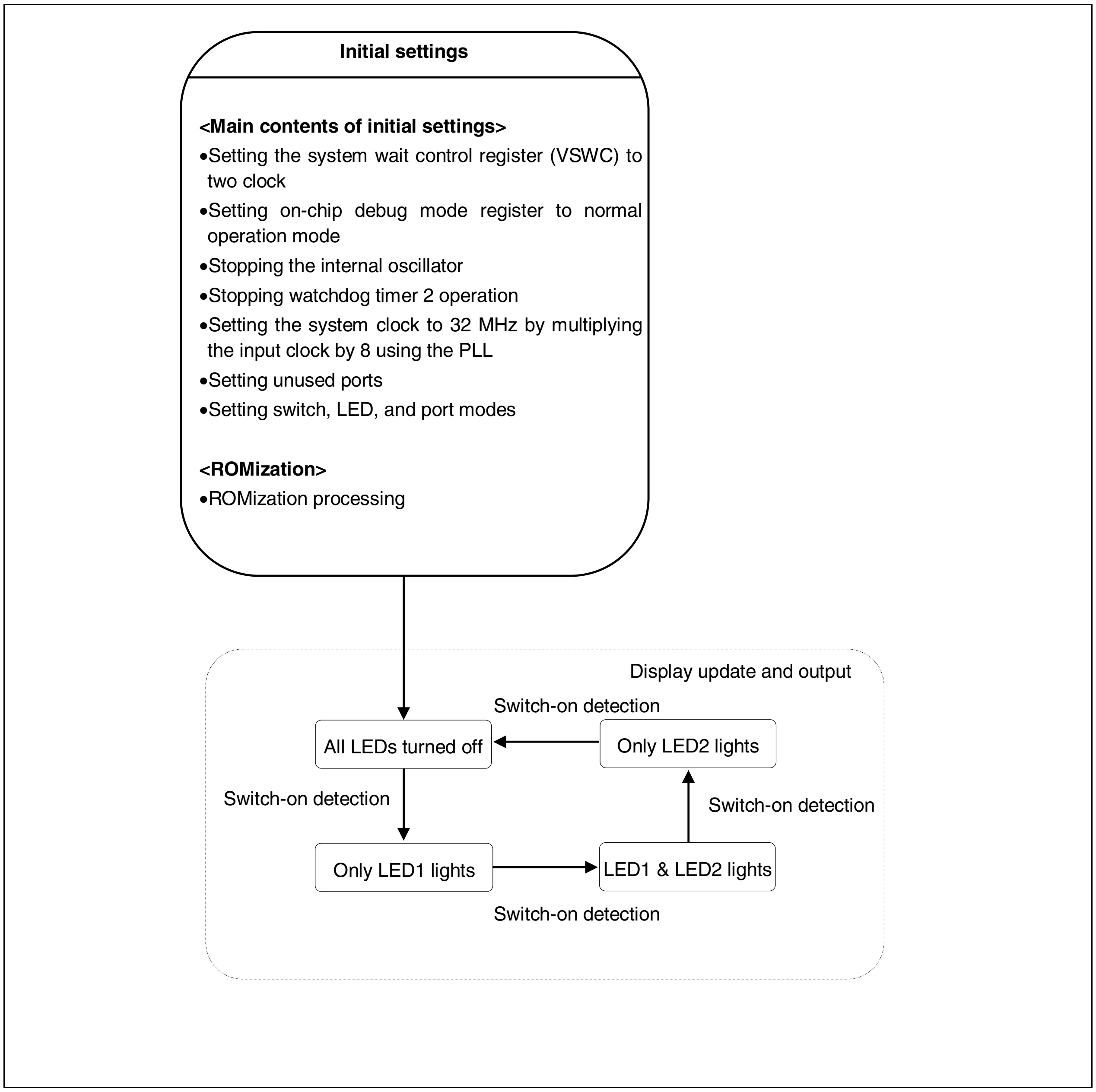
• Input ports (for switch input):	P03 (SW1)
• Output ports (for lighting LEDs):	PCM3 (LED1), PCM2 (LED2)

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.

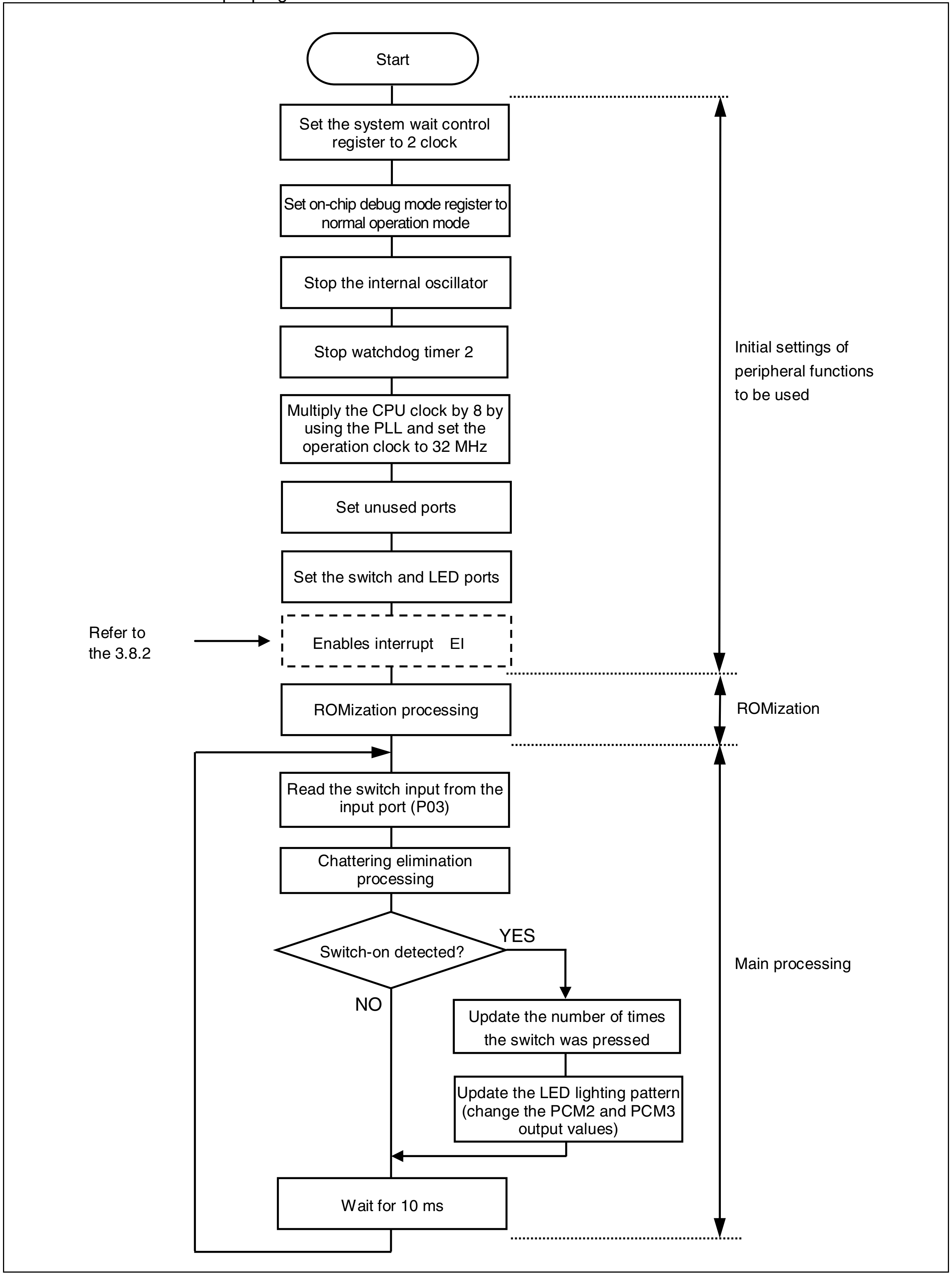
After completing the initial settings, the lighting of two LEDs (LED1 and LED2) is controlled according to the number of switch (SW1) inputs.

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


Initial settings
<Main contents of initial settings>
•Setting the system wait control register (VSWC) to two clock
•Setting on-chip debug mode register to normal operation mode
•Stopping the internal oscillator
•Stopping watchdog timer 2 operation
•Setting the system clock to 32 MHz by multiplying the input clock by 8 using the PLL
•Setting unused ports
•Setting switch, LED, and port modes
<ROMization>
•ROMization processing
Display update and output
Switch-on detection
All LEDs turned off
Only LED2 lights
Switch-on detection
Switch-on detection
Only LED1 lights
LED1 & LED2 lights
Switch-on detection

3.4 Flowchart

A flowchart for the sample program is shown below.


Start
Set the system wait control register to 2 clock
Set on-chip debug mode register to normal operation mode
Stop the internal oscillator
Stop watchdog timer 2
Multiply the CPU clock by 8 by using the PLL and set the operation clock to 32 MHz
Set unused ports
Set the switch and LED ports
Refer to the 3.8.2
Enables interrupt EI
ROMization processing
Read the switch input from the input port (P03)
Chattering elimination processing
Switch-on detected?
YES
NO
Update the number of times the switch was pressed
Update the LED lighting pattern (change the PCM2 and PCM3 output values)
Wait for 10 ms
Initial settings of peripheral functions to be used
ROMization
Main processing

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 in I/O initialization differs.

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

3.6 ROMization

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

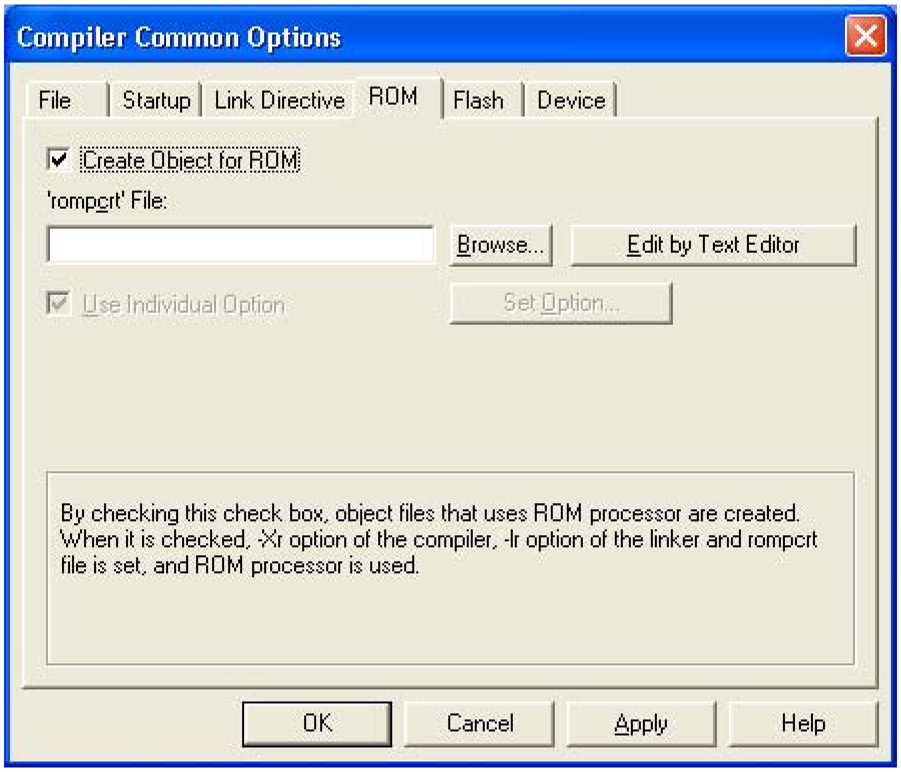
ROMization information is the information of the initial values of variables that have initial values (the section to which variables that have initial values are placed). Variables that have initial values (the section to which variables that have initial values are placed) will hold their software-based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



[✓] Create Object for ROM

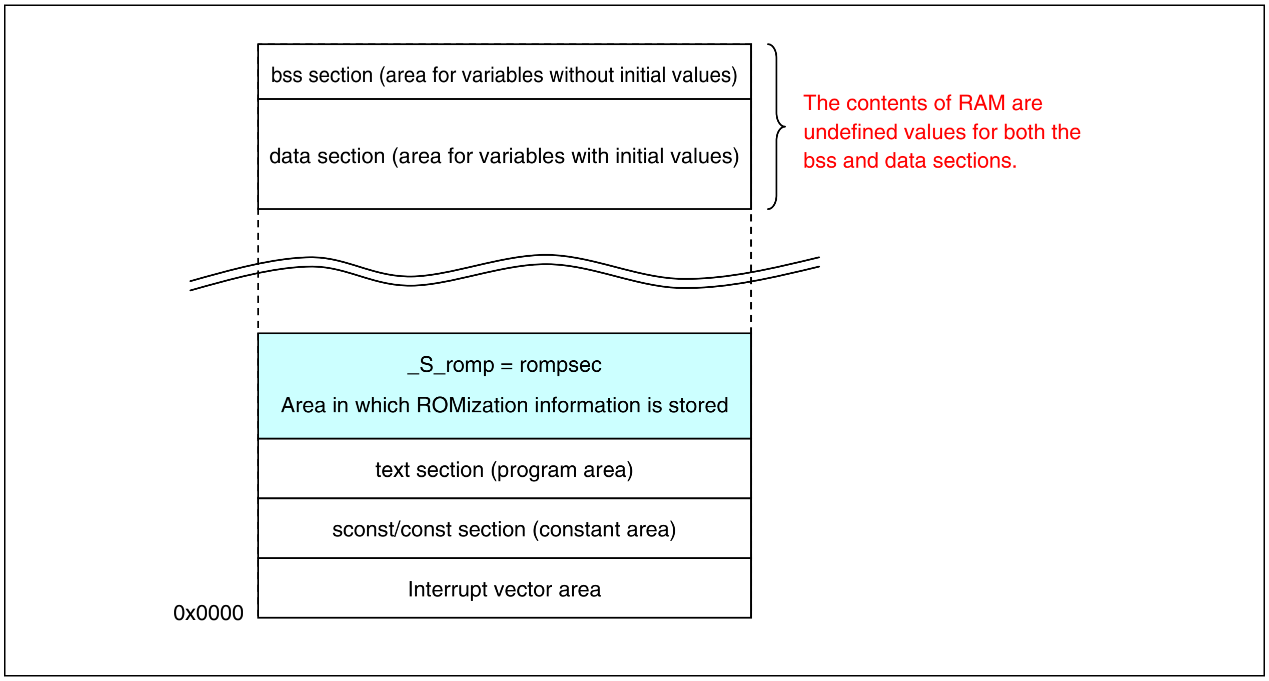
'rompcrt' File:

[✓] Use Individual Option

By checking this check box, object files that uses ROM processor are created. When it is checked, -Xr option of the compiler, -lr option of the linker and rompcrt file is set, and ROM processor is used.

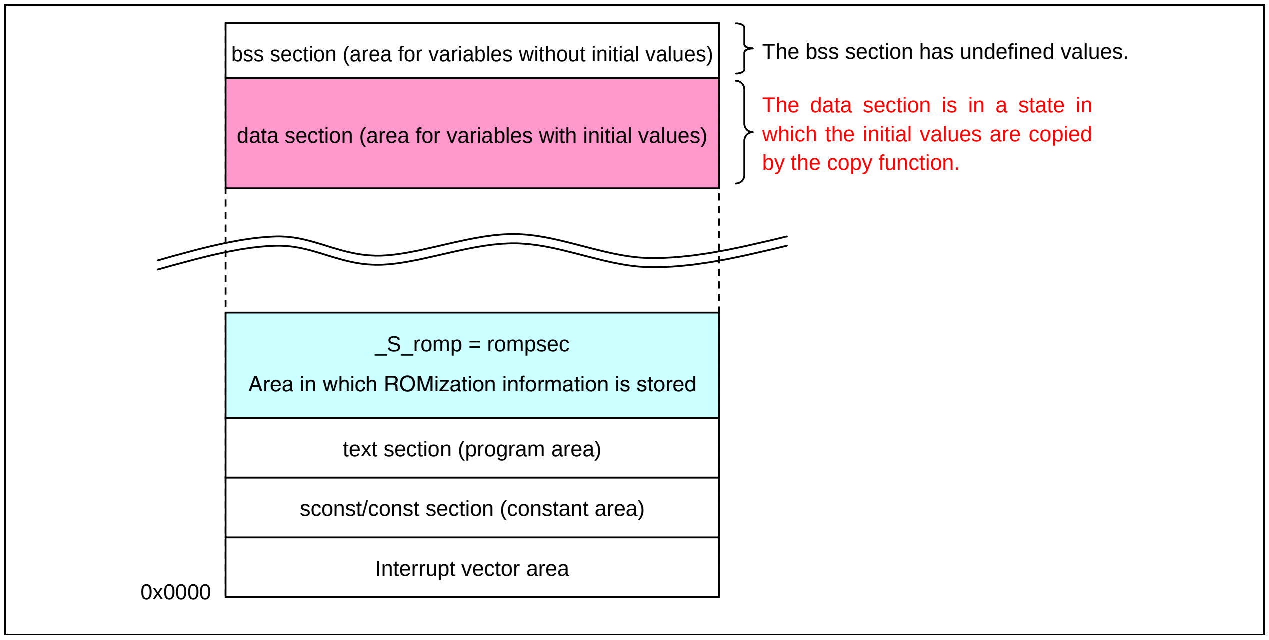
The section into which the ROMization information is to be stored (rompsec) will be automatically added immediately after the program area (.text) section. However, by checking “Create Object for ROM”, a code that indicates the same address as that of rompsec will be generated for the default label _S_romp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


bss section (area for variables without initial values)
data section (area for variables with initial values)
The contents of RAM are undefined values for both the bss and data sections.
_S_romp = rompsec
Area in which ROMization information is stored
text section (program area)
sconst/const section (constant area)
Interrupt vector area
0x0000

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.


bss section (area for variables without initial values)
The bss section has undefined values.
data section (area for variables with initial values)
The data section is in a state in which the initial values are copied by the copy function.
_S_romp = rompsec
Area in which ROMization information is stored
text section (program area)
sconst/const section (constant area)
Interrupt vector area
0x0000

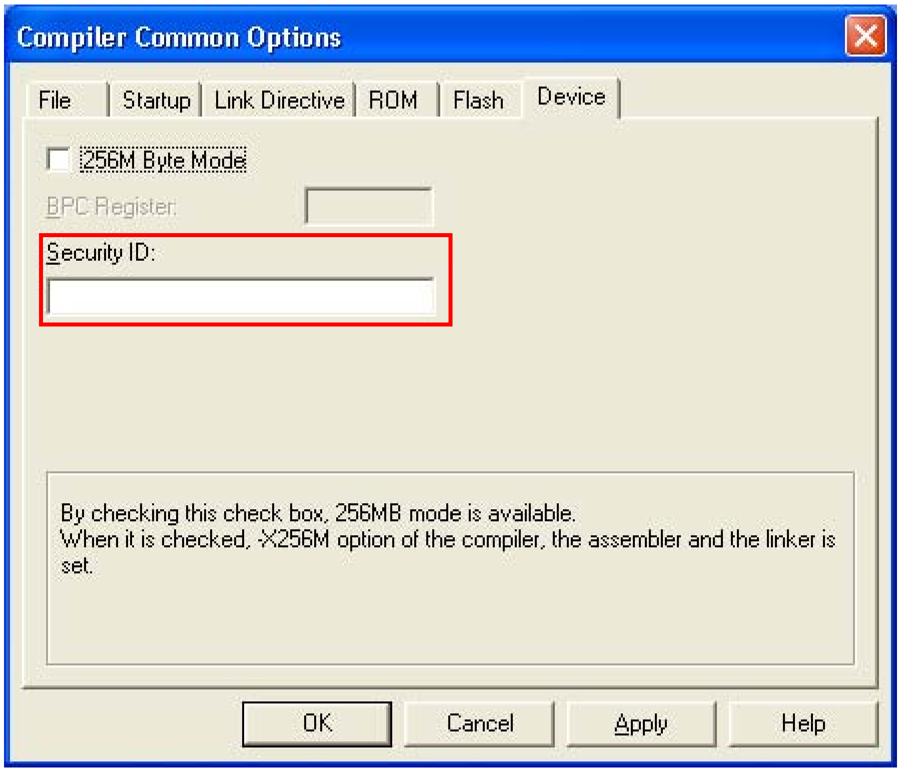
3.7 Security ID

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.

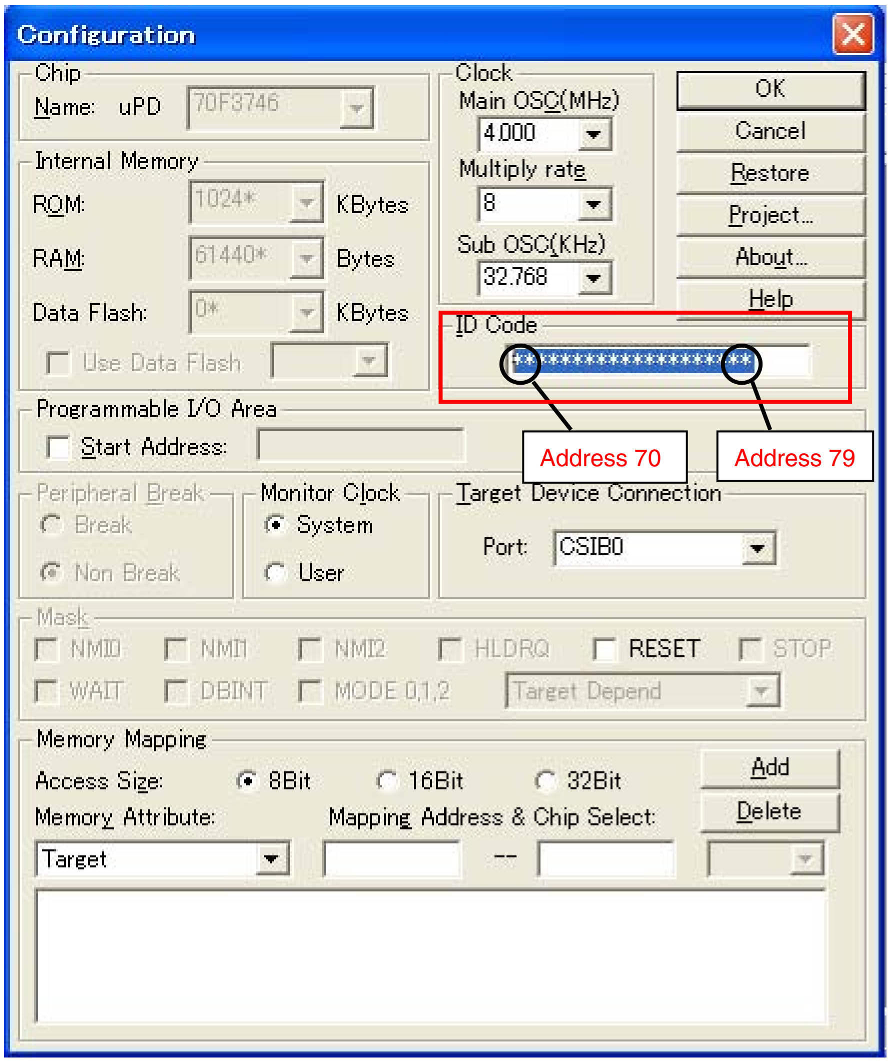
In this sample program (complete-environment version), the security ID is not set and the default security ID value 0xFFFF FFFF FFFF FFFF FFFF is applied.



By checking this check box, 256MB mode is available.
When it is checked, -X256M option of the compiler, the assembler and the linker is set.

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.



Configuration

Chip
Name: uPD 70F3746

Internal Memory
ROM: 1024* KBytes
RAM: 61440* Bytes
Data Flash: 0* KBytes
Use Data Flash

Clock
Main OSC(MHz) 4.000
Multiply rate 8
Sub OSC(KHz) 32.768

OK
Cancel
Restore
Project...
About...
Help

ID Code
****

Programmable I/O Area
Start Address:

Peripheral Break
Break
[✓ Non Break]

Monitor Clock
[✓ System]
User

Target Device Connection
Port: CSIB0

Mask
NMI0 NMI1 NMI2 HLDRQ RESET STOP
WAIT DBINT MODE 0,1,2 Target Depend

Memory Mapping
Access Size: [✓ 8Bit] 16Bit 32Bit
Memory Attribute: Target
Mapping Address & Chip Select: --
Add
Delete

- 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”.

3.8 On-Chip debug with MINICUBE2

The following describes how to set an on-chip debug using MINICUBE2 with pins for CSIB0(SIB0, SOB0, $\overline{\text{SCKB0}}$, and HS(PCM0)) as debug interfaces. These items need to be set in the user program or using the compiler options.

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

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

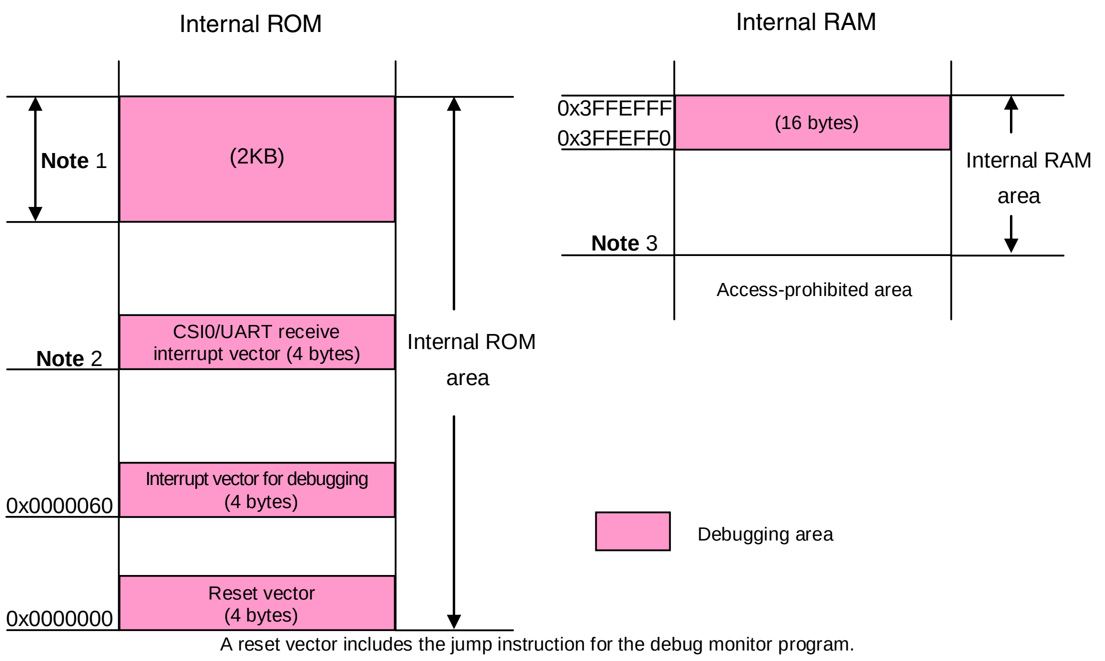
- QB-MINI2 On-Chip Debug Emulator with Programming Function
  User's Manual

3.8.1 Securement of debug monitor program area

The shaded portions in Figure 3-1 are the areas where the debug monitor program is allocated. The monitor program performs initialization processing for debug communication interface and RUN or break processing for the CPU. The internal ROM area must be filled with 0xFF. In using the NEC Electronics compiler CA850, it is necessary adding the assemble source file and link directive code for securing the area.

Remark It is not necessarily required to secure this area if the user program does not use this area. To avoid problems that may occur during the debugger startup, however, it is recommended to secure this area in advance, using the compiler.

Figure 3-1. Memory Spaces Where Debug Monitor Programs Are Allocated



A reset vector includes the jump instruction for the debug monitor program.

Notes 1. Address values vary depending on the product.

	Internal ROM Size	Address Value
µPD70F3739 µPD70F3743	384 KB	0x005F800 - 0x005FFFF
µPD70F3740 µPD70F3744	512 KB	0x007F800 - 0x007FFFF
µPD70F3741 µPD70F3745	768 KB	0x00BF800 - 0x00BFFFF
µPD70F3742 µPD70F3746	1024 KB	0x00FF800 - 0x00FFFFF

2. The Address values depending on the debug interfaces. It starts at 0x0000290 when CSIB0 is used, and at 0x00002F0 when CSIB3 is used, and at 0x0000310 when UARTA0 is used.

3. Address values vary depending on the product.

	Internal RAM Size	Address Value
µPD70F3739 µPD70F3743	32 KB	0x3FF7000
µPD70F3740 µPD70F3744	40 KB	0x3FF5000
µPD70F3741 µPD70F3745	60 KB	0x3FF0000
µPD70F3742 µPD70F3746		

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

• How to secure areas

In this sample program, the following shows examples for securing the area, using the CA850. Add the assemble source file and link directive code, as shown below.

(a) Assemble source (Add the following code as an assemble source file.)

```
 -- Secures 2 KB space for monitor ROM section
 .section "MonitorROM", const
 .space 0x800, 0xff

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

  -- Secures interrupt vector for serial communication
-- Change the section name according to the serial communication mode used
-- In this sample program, the CSIB0 is used
 .section "INTCB0R"
 .space 4, 0xff

 -- Secures 16-byte space for monitor RAM section
 .section "MonitorRAM", bss
 .lcomm monitorramsym, 16, 4
```

(b) Link directive (Add the following code to the link directive file.)

The following shows an example when the internal ROM has 1024 KB (end address is 0x00FFFFF) and internal RAM has 60 KB (end address is 0x3FFEFFF).

```
MROMSEG : !LOAD ?R V0x0ff800{
             MonitorROM     = $PROGBITS ?A MonitorROM;
};
MRAMSEG : !LOAD ?RW V0x03ffeff0{
         MonitorRAM     = $NOBITS ?AW MonitorRAM;
};
```

3.8.2 Enable interrupts (EI) of Serial interfaces

As serial interfaces(for example CSIB0) are used for communication between MINICUBE2 and the target device, forced breaks cannot be executed when interrupts issued for the serial interface, which is used for communication between MINICUBE2 and the target device, are masked. Thus the enable interrupt (EI) is executed as following in sample program.

• Program example

```
/*--------------------------------------------------*/
/* MINICUBE2 enable interrupts for on-chip debug     */
/*--------------------------------------------------*/
__EI();                                   /* Enable interrupt */
```

3.9 #pragma directives

The NEC Electronics compiler CA850 can specify the following #pragma directives.

• Description with assembler instruction

```
    #pragma asm
              assembler instruction
    #pragma endasm
```

Assembler directives can be described in a C language source program.

• Peripheral I/O register name validation specification

```
    #pragma ioreg
```

The peripheral I/O registers of a device are accessed by using peripheral function register names. This specification can use peripheral function register name as variable in C language source.

• Program example

```
    #pragma ioreg       /* Peripheral I/O register name validation specification*/


    P0 =  0b00000000;   /* register name P0 is able to use as variable */
```

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

•CA850 Ver.3.20 C Compiler Package for C Language

CHAPTER 4 SETTING REGISTERS

This chapter describes details of the system wait control register, on-chip debug mode register, watchdog timer 2, operation prohibition of DMA, internal system clock, PLL mode, pin function setting, and main processing.

Set up for peripheral function that are not used in this sample program and stopped after reset release is not done.

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

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

4.1 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.

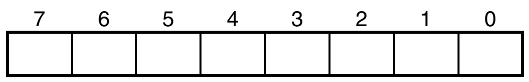
An on-chip peripheral I/O register can be accessed in three clocks (no wait cycles), but the V850ES/Jx3 requires wait cycles according to the operating frequency. Set the following values to the VSWC register in accordance with the operating frequency used.

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

Reset sets this register to 0x77.

Figure 4-1. VSWC Register Format

System wait control register (VSWC)
Address: 0xFFFFF06E



Operating frequency (f_{CLK})	VSWC setting value	Number of wait cycles
32 kHz ≤ f_{CLK} < 16.6 MHz	0x00	0 (no wait cycles)
16.6 MHz ≤ f_{CLK} < 25 MHz	0x01	1
25 MHz ≤ f_{CLK} ≤ 32 MHz	**0x11**	**2**

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

The value set to VSWC is 0x11.

• Program example

```
VSWC = 0b00010001;   /* Inserts two wait cycle when an on-chip peripheral I/O register is accessed.*/
```

4.2 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.

4.2.1 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 is provided with the following eight 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)
- Internal RAM data status register (RAMS)
- 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 system status register (SYS).

4.2.2 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.

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> and <10>.

4.2.3 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.

Figure 4-2. 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).

4.3 Setting Normal Operation Mode for On-Chip Debug mode register

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 **4.2.2 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 ODCM 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 4-3. 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	**[When using MINICUBE2 (QB-MINI2) in normal operation mode]** 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.
1	**[When using MINICUBE (QB-V850MINI)]** 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) 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)

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

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

- Program example

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

4.4 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 4-4. RCM Register Format

Internal oscillation mode register (RCM)
Address: 0xFFFFF80C

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

RSTOP	Oscillating or stopping internal oscillator
0	Oscillate the internal oscillator.
1	Stop the internal oscillator.

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

The value set to RCM is 0x01.

• Program example

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

4.5 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 units.

Reset sets this register to 0x67.

Figure 4-5. WDTM2 Register Format

Watchdog timer mode register 2 (WDTM2)
Address: 0xFFFFF6D0

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

WDM21	WDM20	Watchdog timer 2 operation mode selection
0	**0**	Stop operation.
0	1	Non-maskable interrupt request mode (INTWDT2 signal generated)
1	–	Reset mode (WDT2RES signal generated)

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

The value set to WDTM2 is 0x00.

- Program example

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

4.6 Clock Setting

In this sample program, an example in which a 4 MHz ceramic or crystal resonator is connected to the X1 and X2 pins and the clock of the resonator is multiplied by 8 in PLL mode and used as the internal system clock (32 MHz) is shown. The subclock is not used.

4.6.1 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 4-6. PCC Register Format

Processor clock control register (PCC)

Address: 0xFFFFF828

7	6	5	4[Note]	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.

Note The CLS bit is a read-only bit.

The value set to PCC is 0x80.

- Program example

```
                    /* 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
```

4.6.2 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 4-7. PLLCTL Register Format

PLL control register (PLLCTL)

Address: 0xFFFFF82C

7	6	5	4	3	2	1	0
0	0	0	0	0	0	SELPLL	PLLON

SELPLL	Operation mode
0	Clock-through mode
1	PLL mode
The SELPLL bit can be set to 1 only when the PLL clock frequency has stabilized. If the SELPLL bit is written while the PLL clock frequency is not stable (unlocked), 0 is written.	

PLLON	PLL operation stop control
0	Stop PLL.
1	Operate PLL. (A lockup time is required until the frequency stabilizes after the PLL is started.)

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

4.6.3 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: 0xFFFFF824

7	6	5	4	3	2	1	0
0	0	0	0	1	0	1	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 = 0x00). When shifting to PLL mode without stopping the PLL, such as during the initial settings, checking the lock register can be omitted.

4.6.4 Clock control register (CKC)

The CKC register controls the internal system clock in the PLL mode.

The CKC register is a special register. Data can be written to this register only in a combination of specific sequence.

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

Reset sets this register to 0x0A.

Figure 4-9. CKC Register Format

Clock control register (CKC)
Address: 0xFFFFF822

7	6	5	4	3	2	1	0
0	0	0	0	1	0	1	CKDIV0

CKDIV0	Internal system clock (f_{XX}) in PLL mode
0	f_{XX} = 4 x f_X (f_X = 2.5 to 5.0 MHz)
1	f_{XX} = 8 x f_X (f_X = 2.5 to 4.0 MHz)
The PLL mode cannot be used at f_X = 5.0 to 10.0 MHz. And before changing the multiplication factor between 4 and 8 by using the CKC register, set the clock-through mode and stop the PLL.	

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

4.6.5 Usage

To set to eight multiplication in PLL mode, Crock control register (CKC) is set to 0x0B (The CKC register is a special register. Data can be written to this register only in a combination of specific sequence.).

First, stop the PLL, and then changing the multiplication factor for 4 to 8 by using the CKC register.

- To disable PLL operation

  <1> Set the SELLPLL bit to 0 for changing the clock-through mode.

  <2> Wait for eight clocks or more, then stop the PLL by set PLLON bit to 0.

  Remark After the reset signal has been released, the PLL is operated. Because the default mode of CPU operation clock selection is the clock-through mode, it is possible to stop PLL (set PLLON bit to 0) without setting SELPLL bit to 0 and eight clocks waiting.

Second, set the PLL to operated for translated PLL mode.

- To enable PLL operation

  <1> To enable PLL operation from PLL is stopped, set the PLLON bit to 1, and then set the SELPLL bit to 1 after the LOCKR.LOCK bit = 0.

- Program example

```
  /* Select the PLL mode (CPU operating clock fxx is 20 to 32 MHz (fx = 2.5 ~ 4.0 MHz)*/
  /* Default setting is clock-through mode */

  PLLON = 0;          /* PLL is stopped */

                      /* 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 */

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

4.7 Setting Ports

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

	V850ES/JG3	V850ES/JJ3
Port 0	P02 to P06	P00 to P06
Port 1	P10, P11	P10 to P11
Port 3	P30 to P39	P30 to P39
Port 4	P40 to P42	P40 to P42
Port 5	P50 to P55	P50 to P55
Port 6	None	P60 to P615
Port 7	P70 to P711	P70 to P715
Port 8	None	P80 to P81
Port 9	P90 to P915	P90 to P915
Port CD	None	PCD0 to PCD3
Port CM	PCM0 to PCM3	PCM0 to PCM5
Port CS	None	PCS0 to PCS7
Port CT	PCT0, PCT1, PCT4, PCT6	PCT0 to PCT7
Port DH	PDH0 to PDH5	PDH0 to PDH7
Port DL	PDL0 to PDL15	PDL0 to PDL15

4.7.1 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

Port n register (Pn)

7	6	5	4	3	2	1	0
Pn7	Pn6	Pn5	Pn4	Pn3	Pn2	Pn1	Pn0

Pnm	Output data control (in output mode)
0	Output 0.
1	Output 1.

Remark The red value indicates the value set to unused ports.

[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.

4.7.2 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

Port n mode register (PMn)

7	6	5	4	3	2	1	0
PMn7	PMn6	PMn5	PMn4	PMn3	PMn2	PMn1	PMn0

PMnm	I/O mode control
0	Output mode
1	Input mode

Remark The red value indicates the value set to unused ports.

4.7.3 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

Port n mode control register (PMCn)

7	6	5	4	3	2	1	0
PMCn7	PMCn6	PMCn5	PMCn4	PMCn3	PMCn2	PMCn1	PMCn0

PMCnm	Operation mode specification
0	Port mode
1	Alternate-function mode

Remark The red value indicates the value set to unused ports.

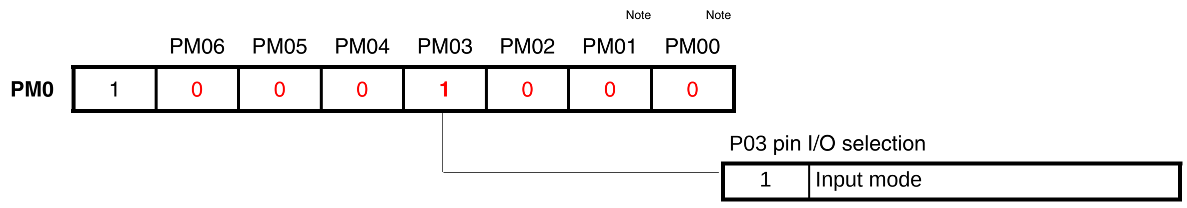
[Column] Writing to and reading from the Pn register

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 (V850ES/JJ3)



		PM06	PM05	PM04	PM03	PM02	PM01	PM00
PM0	1	0	0	0	1	0	0	0

P03 pin I/O selection

1	Input mode

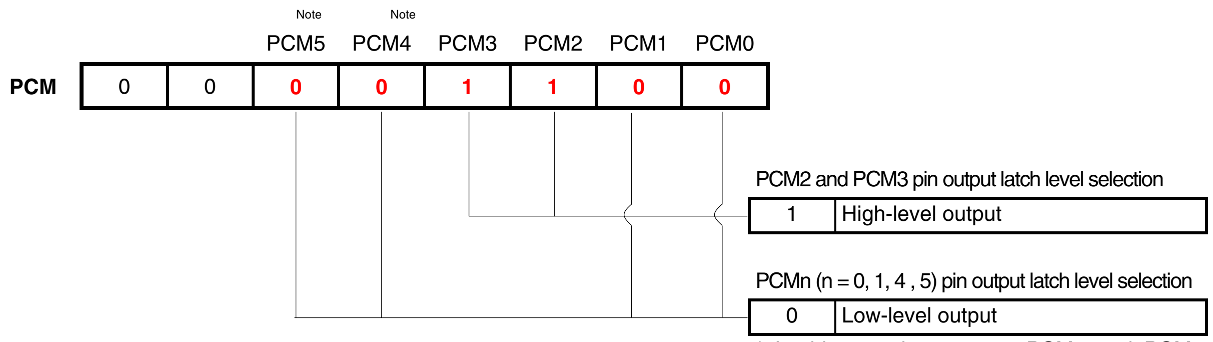
Remark The value of nameless bit (7 bit) is fixed by each products.
Note V850ES/JG3 doesn't have P00 and P01, then PM01 and PM00 are fixed to 1.

The value set to PM0 is 0x88 (In case of V850ES/JG3, the value set to PM0 is 0x8B.).

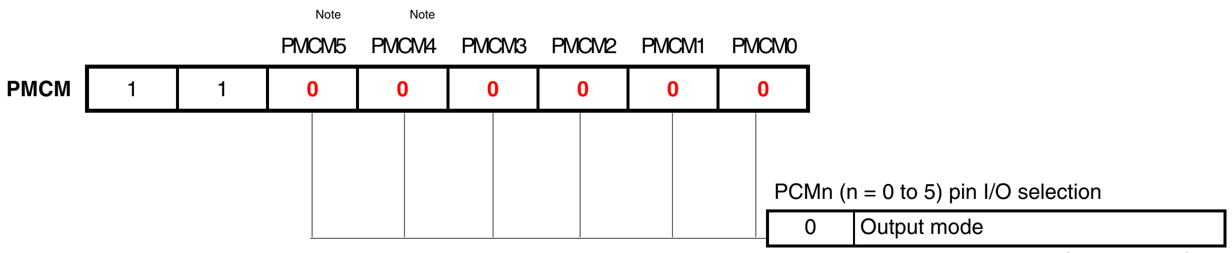
• Program example (V850ES/JJ3)

```
PM0 = 0b10001000;      /* Sets P00 to P06 to low-level output (except P03). */
```

[Example 2] • Setting the output latches of PCM2 and PCM3 to high-level output (V850ES/JJ3)
• Setting PCM2 and PCM3 as output ports (V850ES/JJ3)


Note
Note
PCM5
PCM4
PCM3
PCM2
PCM1
PCM0
PCM
0
0
0
0
1
1
0
0
PCM2 and PCM3 pin output latch level selection
1 High-level output
PCMn (n = 0, 1, 4 , 5) pin output latch level selection
0 Low-level output

* 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.

Remark The value of nameless bit (6, 7 bit) is fixed by each products.

Note V850ES/JG3 doesn't have PCM4 and PCM5, then PCM4 and PCM5 are fixed to 0, and PMCM4 and PMCM5 are fixed to 1.

The value set to PCM is 0x0C and the value set to PMCM is 0xC0

(In case of V850ES/JG3, the value set to PCM is 0x0C, and the value set to PMCM is 0xF0).

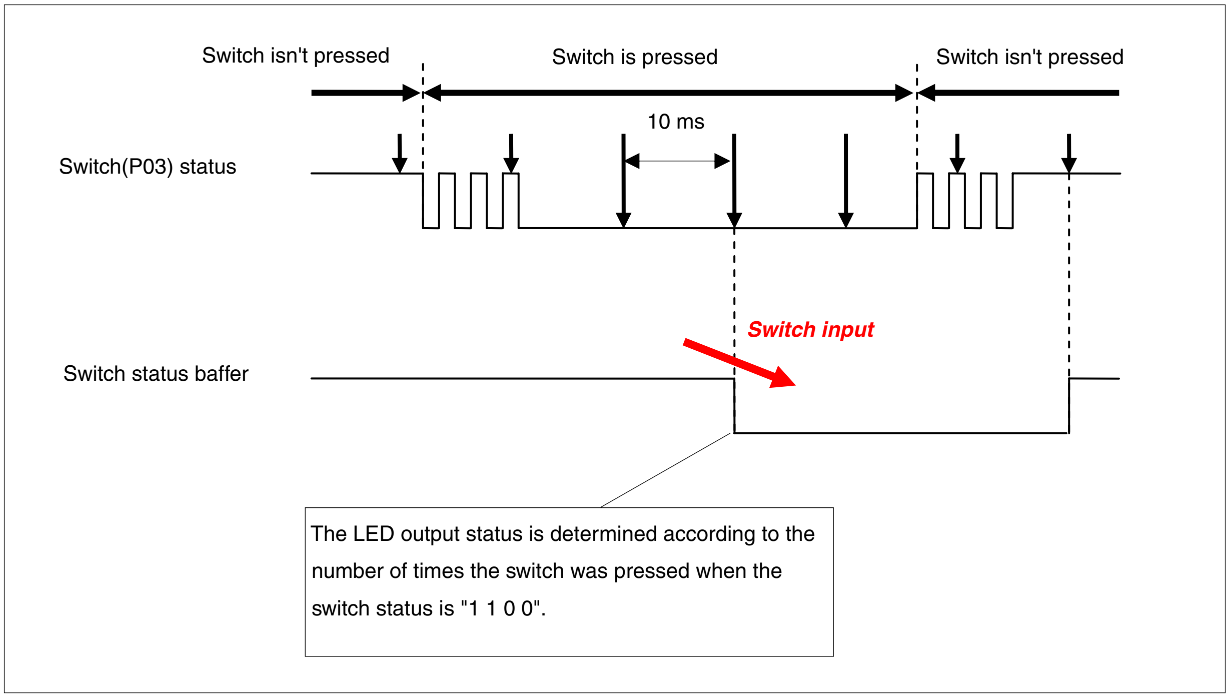
• Program example (V850ES/JJ3)

```
PCM =  0b00001100;          /* Sets the output latches of PCM2 and PCM3 to high-level output. */
PMCM = 0b11000000;          /* Sets PCM0 to PCM5 as output ports. */
```

4.8 Main Processing

4.8.1 Chattering countermeasure

To eliminate chattering, a change in the switch (P03) status is determined by reading inputs every 10 ms and detecting the same level for the switch status two times in succession.


Switch isn't pressed
Switch is pressed
Switch isn't pressed
10 ms
Switch(P03) status
Switch input
Switch status baffer
The LED output status is determined according to the number of times the switch was pressed when the switch status is "1 1 0 0".

The following operation is performed as 10 ms wait processing.

• Program example

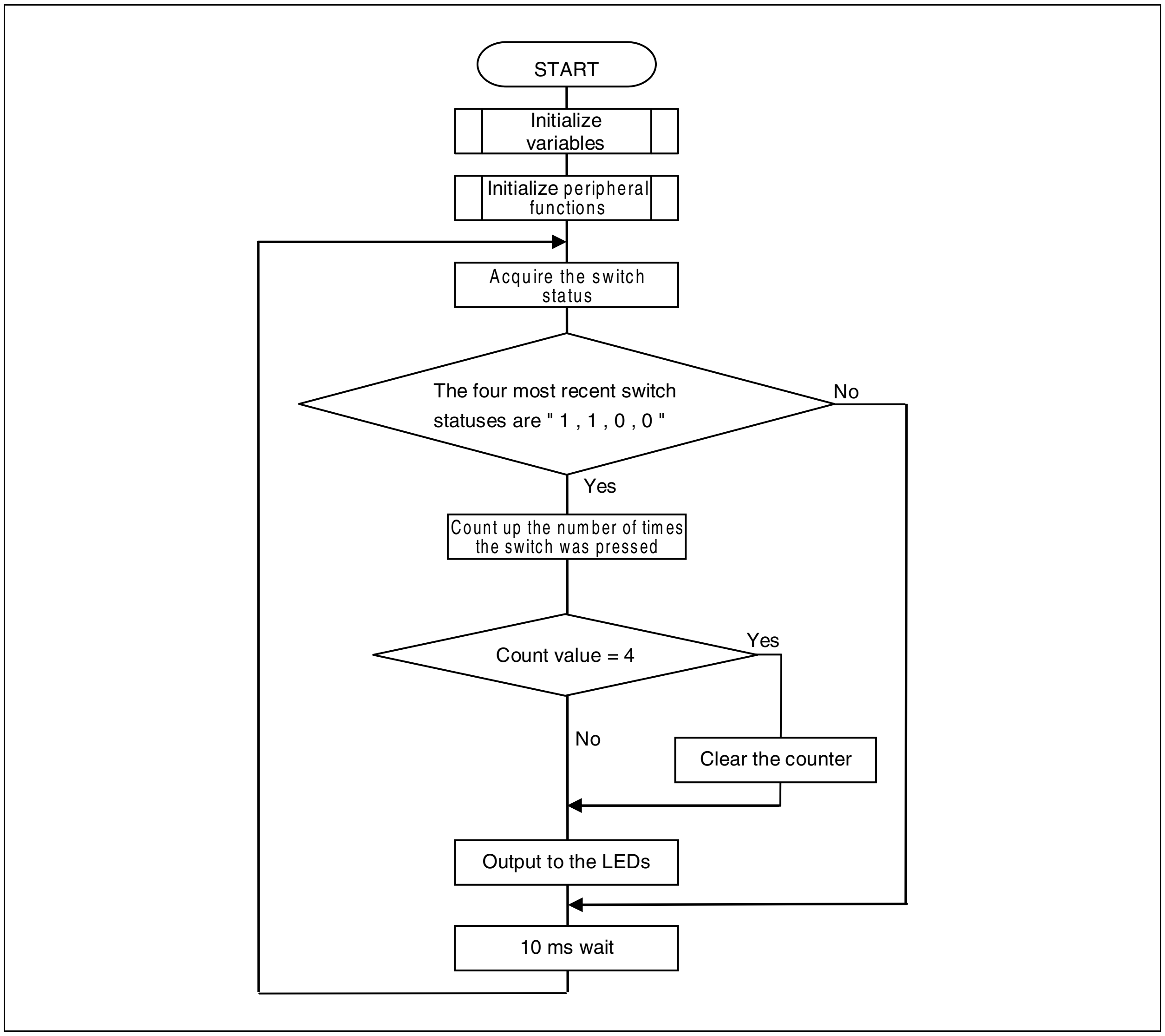
```
unsigned long loop_wait;          /* Counter for loop */

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

4.8.2 Main processing

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

In this sample program, the correspondence between the input and output data is set in an array.


START
Initialize variables
Initialize peripheral functions
Acquire the switch status
The four most recent switch statuses are " 1 , 1 , 0 , 0 "
No
Yes
Count up the number of times the switch was pressed
Count value = 4
Yes
No
Clear the counter
Output to the LEDs
10 ms wait

```
/************************************************************************
       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 LED display pattern data    */
        0x0c,                       /* Turns off all LEDs.          */
        0x04,                       /* Lights LED1.
        0x00,                       /* Lights LED1 and LED2
        0x08                        /* Lights LED2.                 */
    };
    unsigned char indata = 0b00000001;    /* To memorize the Switch status
                                          (initialized if the previous value is "1") */
    unsigned char count;                  /* Number of times the switch input      */
    unsigned long loop_wait;              /* Counter for loop                  */

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

    /*----------------------------------------*/
    /* Peripheral-function initialization      */
    /*----------------------------------------*/
    f_init_vswc();                        /* Sets the VSWC register                   */
    f_init_ocdm();                        /* Sets on-chip debug mode register 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          */
    /*------------------------------------*/
    _rcopy( &_S_romp, -1 );               /* Executes ROMization                  */

     /*------------------------------------*/
    /* MINICUBE2 : Enable interrupt for OCD*/
    /*------------------------------------*/
    __EI();                               /* Enable interrupt                  */
```

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

```
    /*------------------------------------*/
    /* 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 input */
            count &= 0b00000011
            PCM = outdata[count];    /* Displays the LED 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.

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/Jx3 microcontroller source program is shown below as a program list example.

● minicube2.s

```
#-----------------------------------------------------------------------------
#
#    NEC Electronics     V850ES/Jx3 series
#
#-----------------------------------------------------------------------------
#    V850ES/JJ3 JG3 sample program
#-----------------------------------------------------------------------------
#    LED lighting switch control
#-----------------------------------------------------------------------------
#[History]
#    2009.6.--    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
```

Set the section name according to use serial interface.
  when interface is CSIB3 : INTCB3R
  when interface is UARTA0 : INTUA0R

● 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;
};


SIDATA  : !LOAD ?RW V0x3ff0000 {
```

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.

```
        .tidata.byte    = $PROGBITS     ?AW .tidata.byte;
        .tibss.byte     = $NOBITS       ?AW .tibss.byte;
        .tidata.word    = $PROGBITS     ?AW .tidata.word;
        .tibss.word     = $NOBITS       ?AW .tibss.word;
        .tidata         = $PROGBITS     ?AW .tidata;
        .tibss          = $NOBITS       ?AW .tibss;
        .sidata         = $PROGBITS     ?AW .sidata;
        .sibss          = $NOBITS       ?AW .sibss;
};

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


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


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

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

A reserved area for MINICUBE2 is secured.

● main.c

```
/*------------------------------------------------------------------------------*/
/*
/*  NEC Electronics     V850ES/Jx3 series
/*
/*------------------------------------------------------------------------------*/
/*  V850ES/JJ3 sample program
/*------------------------------------------------------------------------------*/
/*  LED lighting switch control
/*------------------------------------------------------------------------------*/
/*[History]
/*  2009.06.--  Released
/*------------------------------------------------------------------------------*/
/*[Overview]
/*  This sample program selects the clock frequency, sets the port I/Os, and performs
/*  the basic initial settings of the V850ES/JJ3 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 two clock
/*  • Setting on-chip debug mode register to normal operation mode
/*  • Stopping the internal oscillator
/*  • Stopping watchdog timer 2 operation
/*  • Setting the system clock to 32 MHz by multiplying the input clock by 8 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   : P00-P02, P04-P06, P10-P11, P30-P39, P40-P42, P50-P55, P60-P615,
/*                    P70-P715, P80-P81, P90-P915, PCD0-PCD3, PCM0-PCM1, PCM4-PCM5,
/*                    PCS0-PCS7, PCT0-PCT7, PDH0-PDH7, PDL0-PDL15
/*  *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
                                        input      */
#define VAL_TIMER_WAIT      (40193)  /* 10 ms wait        */
```

```
/*------------------------------*/
/* Prototype declarations      */
/*------------------------------*/
static  void f_init_vswc( void );          /* VSWC register setting processing       */
static  void f_init_ocdm( void );        /* On-chip debug mode register 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 = 0b00010001;                   /* Inserts two wait cycle when the on-chip
                                            peripheral I/O register is accessed.  */

    return;
}



/*-----------------------------------------------------*/
/* Setting on-chip debug mode register 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 )
{
/* Select the PLL mode (CPU operating clock fXX is 20 to 32 MHz (fX = 2.5 ~ 4.0 MHz)*/
/* Default setting is clock-through mode */

PLLON = 0;          /* PLL is stopped */

                    /* 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 */

   SELPLL = 1;       /* Set PLL mode */


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

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

       return;
}

/*---------------------------*/
/* Setting unused ports */
/*---------------------------*/
static void f_init_blank_port( void )
{
    P0 =  0b00000000;          /* Sets P00 to P06 to low-level output (except P03)   */
    PM0 = 0b10001000;
```

In the V850ES/JG3, sets only P02 to P06.

```
    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;

    P3H =  0b00000000;       /* Sets P38 and P39 to low-level output.             */
    PM3H = 0b11111100;
    P3L =  0b00000000;       /* Sets P30 to P37 to low-level output.              */
    PM3L = 0b00000000;

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

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

P6H   = 0b00000000;      /* Sets P68 to P615 to low-level output. */
PM6H  = 0b00000000;
P6L   = 0b00000000;      /* Sets P60 to P67 to low-level output. */
PM6L  = 0b00000000;
```

In the V850ES/ JG3, P6 settings are unnecessary.

This settings are unnecessary

```
P7H =  0b00000000;        /* Sets P78 to P715 to low-level output.             */
PM7H = 0b00000000;
P7L =  0b00000000;        /* Sets P70 to P77 to low-level output.             */
PM7L = 0b00000000;
```

In the V850ES/ JG3 sets only P70 to P711.

```
P7H =  0b00000000;        /* Sets P78 to P711 to low-level output.             */
PM7H = 0b11110000;
P7L =  0b00000000;        /* Sets P70 to P77 to low-level output.             */
PM7L = 0b00000000;
```

```
P8 = 0b00000000;       /* Sets P80 to P81 to low-level output.                  */
PM8 = 0b11111100;
```

In the V850ES/ JG3, P8 settings are unnecessary.

This settings are unnecessary

```
P9H =  0b00000000;      /* Sets P98 to P915 to low-level output.               */
PM9H = 0b00000000;
P9L =  0b00000000;      /* Sets P90 to P97 to low-level output.              */
PM9L = 0b00000000;

PCD = 0b00000000;       /* Sets PCD0 to PCD3 to low-level output.               */
PMCD = 0b11110000;
```

In the V850ES/ JG3, PCD settings are unnecessary.

This settings are unnecessary

```
    PCM =  0b00000000;     /* Sets PCM0 and PCM5 to low-level output.               */
    PMCM = 0b11000000;
```

In the V850ES/ JG3, sets only PCM0 to PCM3.

```
    PCM =  0b00000000;     /* Sets PCM0 and PCM3 to low-level output.               */
    PMCM = 0b11110000;
```

```
    PCS = 0b00000000;      /* Sets PCS0 to PCS7 to low-level output.                */
    PMCS = 0b00000000;
```

In the V850ES/ JG3, PCS settings are unnecessary.

This settings are unnecessary

```
    PCT =  0b00000000;     /* Sets PCT0 to PCT7 to low-level output.           */
    PMCT = 0b00000000;
```

In the V850ES/ JG3, sets only PCT0, PCT1, PCT4 and PCT6.

```
    PCT =  0b00000000;     /* Sets PCT0, PCT1, PCT4 and PCT6 to low-level output.  */
    PMCT = 0b10101100;
```

```
    PDH =  0b00000000;     /* Sets PDH0 to PDH7 to low-level output.                */
    PMDH = 0b00000000;
```

In the V850ES/ JG3, sets only PDH0 to PDH5.

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

    PDLH =  0b00000000;    /* Sets PDL8 to PDL15 to low-level output.              */
    PMDLH = 0b00000000;
    PDLL =  0b00000000;    /* Sets PDL0 to PDL7 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 = 0b10001000;

    /* Setting the LED output port */
    PCM =  0b00001100;            /* Sets PCM2 and PCM3 to high-level output.        */
    PMCM = 0b11000000;

    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 LED display pattern data     */
        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 “1”) */
    unsigned char count;                /* Number of times the switch input */
    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 debug mode register 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                              */

/*-----------------------------------*/
/* MINICUBE2 : Enable interrupt for OCD*/
/*-----------------------------------*/
__EI();                          /* Enable interrupt                 */

/*----------------------------------------*/
/* 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 input      */
        count &= 0b00000011;
        PCM = outdata[count]; /* Displays the LED display data read from the table*/
    }

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

```
    return;
}
```

For further information, please contact:

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

[America]

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

[Europe]

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

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

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

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

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

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

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

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

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

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

[Asia & Oceania]

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

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

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

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

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

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

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

G0706