To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

Renesas Electronics website: http://www.renesas.com

April 1st, 2010
Renesas Electronics Corporation

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

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

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: “Standard”, “High Quality”, and “Specific”. The recommended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as “Specific” without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as “Specific” or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is “Standard” unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

   “Standard”: Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.

   “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.

   “Specific”: Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

RENESAS

Application Note

V850ES/Jx3-L

Sample Program (16-bit Timer/Event Counter (TMP, TMQ))

Interval Timer Mode

This document provides an operational overview of the sample program and describes how to use it and how to set up and use the interval timer function of the 16-bit timer/event counter P (TMP) and the 16-bit timer/event counter Q (TMQ). In the sample program, LED1 is made to blink at fixed cycles by using the interval timer function of TMP. Furthermore, the blinking cycle of LED1 changes in accordance with the number of switch inputs.

Target devices

V850ES/JF3-L microcontroller

V850ES/JG3-L microcontroller

CONTENTS

Document No. U19674EJ1V0AN00
Date Published April 2009 N

- **The information in this document is current as of February, 2009. The information is subject to change without notice. For actual design-in, refer to the latest publications of NEC Electronics data sheets or data books, etc., for the most up-to-date specifications of NEC Electronics products. Not all products and/or types are available in every country. Please check with an NEC Electronics sales representative for availability and additional information.**
- No part of this document may be copied or reproduced in any form or by any means without the prior written consent of NEC Electronics. NEC Electronics assumes no responsibility for any errors that may appear in this document.
- NEC Electronics does not assume any liability for infringement of patents, copyrights or other intellectual property rights of third parties by or arising from the use of NEC Electronics products listed in this document or any other liability arising from the use of such products. No license, express, implied or otherwise, is granted under any patents, copyrights or other intellectual property rights of NEC Electronics or others.
- Descriptions of circuits, software and other related information in this document are provided for illustrative purposes in semiconductor product operation and application examples. The incorporation of these circuits, software and information in the design of a customer's equipment shall be done under the full responsibility of the customer. NEC Electronics assumes no responsibility for any losses incurred by customers or third parties arising from the use of these circuits, software and information.
- While NEC Electronics endeavors to enhance the quality, reliability and safety of NEC Electronics products, customers agree and acknowledge that the possibility of defects thereof cannot be eliminated entirely. To minimize risks of damage to property or injury (including death) to persons arising from defects in NEC Electronics products, customers must incorporate sufficient safety measures in their design, such as redundancy, fire-containment and anti-failure features.
- NEC Electronics products are classified into the following three quality grades: "Standard", "Special" and "Specific".

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

  "Standard": Computers, office equipment, communications equipment, test and measurement equipment, audio and visual equipment, home electronic appliances, machine tools, personal electronic equipment and industrial robots.

  "Special": Transportation equipment (automobiles, trains, ships, etc.), traffic control systems, anti-disaster systems, anti-crime systems, safety equipment and medical equipment (not specifically designed for life support).

  "Specific": Aircraft, aerospace equipment, submersible repeaters, nuclear reactor control systems, life support systems and medical equipment for life support, etc.

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

(Note)

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

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

M8E 02. 11-1

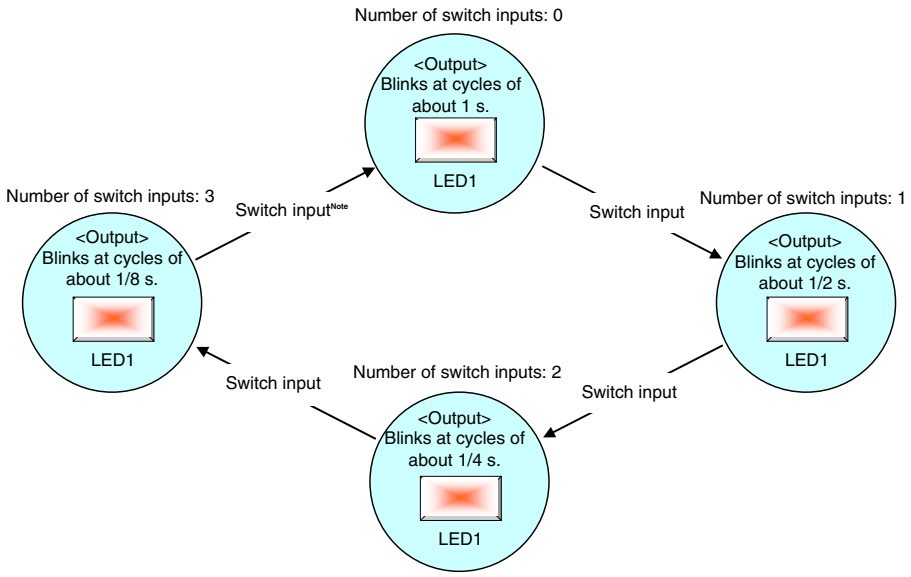
CHAPTER 1 OVERVIEW

An example of using the interval timer function of the 16-bit timer/event counter P (TMP) is presented in the sample program. LED1 is made to blink at fixed cycles and the blinking cycle of LED1 changes in accordance with the number of switch inputs.

The blinking cycle is controlled by using the interval specified for the 16-bit timer/event counter P (TMP) and is changed by changing the interval when a switch input is detected.

Peripherals that stop immediately after a reset release and are not used in the sample program are not set up.

The relationship between the number of switch inputs and the blinking cycle is shown below.



Note The blinking cycle from the zeroth switch input is repeated after the fourth switch input.

1.1 Initial Settings

<Referencing option byte>

- Referencing the oscillation stabilization time after releasing reset

<Settings of on-chip peripheral functions>

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

<Pin settings>

- Setting unused pins
- Setting external interrupt pins (edge specification, priority specification, unmasking)
- Setting LED1 output pins

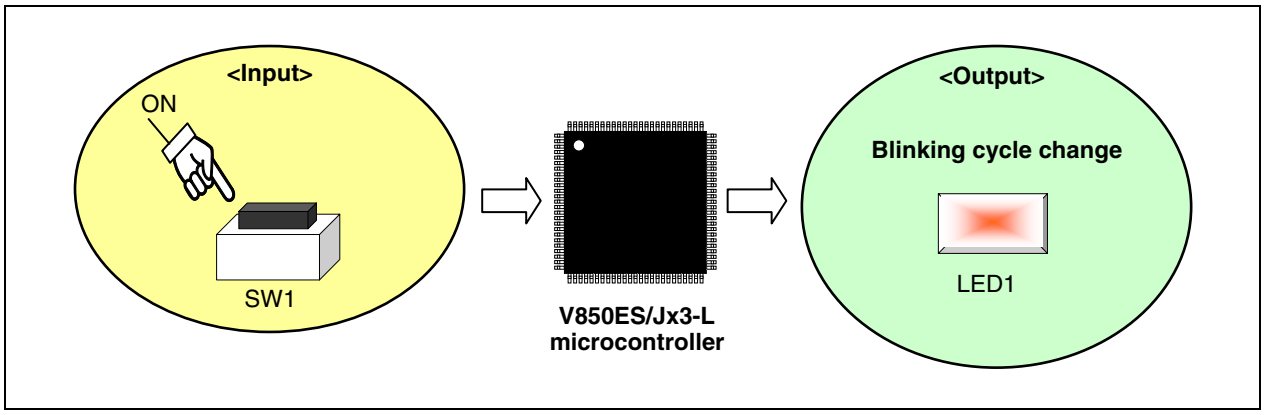
<Timer (TMP) settings>

- Setting the count clock to fxx (20 MHz)
- Setting the interval to 2 ms (0.05 μs $\times$ 40,000)
- Unmasking the INTTP0CC0 interrupt

1.2 Interval Interrupt by 16-bit Timer/Event Counter P (TMP)

After specifying the initial settings, LED1 is made to blink at fixed cycles by using the interrupt (INTTP0CC0) generated by the 16-bit timer/event counter P (TMP).

Blinking cycles of 0.125 to 1 s are calculated by counting the number of interrupts generated at 0.25 to 2 ms intervals, which are specified for TMP, and LED1 blinks at those cycles. The blinking cycle changes each time the switch connected to the INTTP0 pin is turned on, in the manner of 1 s → 0.5 s → 0.25 s → 0.125 s → 1 s (repeat).



1.3 INTP0 Interrupt Servicing Triggered by Switch Input

When the falling edge of the INTP0 pin triggered by switch input is detected, an INTP0 interrupt is serviced. If the INTP0 pin is at high level (the switch is off) 10 ms after a fall of the INTP0 pin was detected, the occurrence of chattering is identified. If the INTP0 pin is at low level (the switch is on) 10 ms after the edge was detected, the LED1 blinking cycle changes in accordance with the number of switch inputs.

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

[Column] Chattering

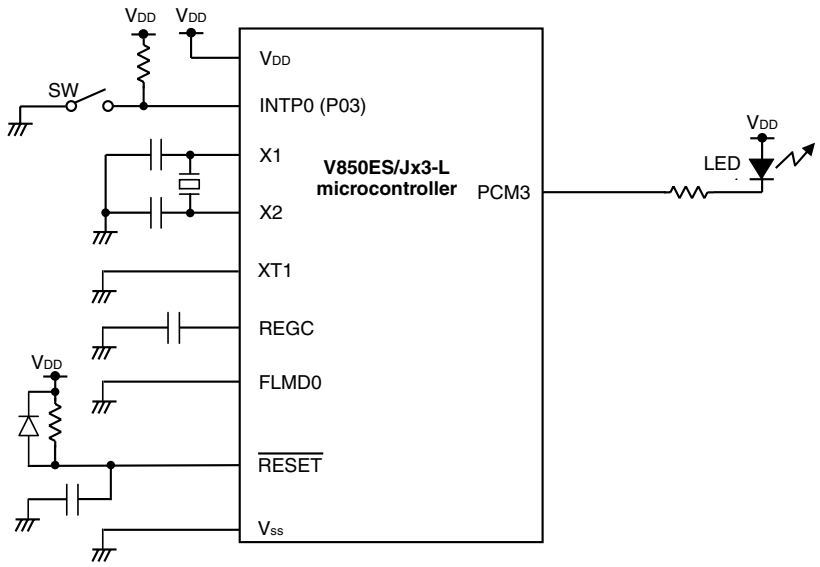
Chattering is a phenomenon in which the electric signal repeats turning on and off due to a mechanical flip-flop of the contacts, immediately after the switch has been pressed.

CHAPTER 2 CIRCUIT DIAGRAM

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

2.1 Circuit Diagram

The circuit diagram is shown below.



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

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

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

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

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

2.2 Peripheral Hardware

The peripheral hardware to be used is shown below.

(1) Switch (SW1)

This switch is used as an input to generate interrupts for controlling the LED1 blinking.

(2) LED (LED1)

LED1 is used as an output for the interval timer function of the 16-bit timer/event counter P (TMP) and switch inputs.

CHAPTER 3 SOFTWARE

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

3.1 File Configuration

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

[C language version]

File Name (Tree Structure)	Description	Compressed (*.zip) Files Included	
			
c — conf — crtE.s	Startup routine file[Note 1]	–	●
conf — AppNote_ITVL_TMP.dir	Link directive file[Note 2]	●	●
conf — AppNote_ITVL_TMP.prj	Project file for integrated development environment PM+	–	●
conf — AppNote_ITVL_TMP.prw	Workspace file for integrated development environment PM+	–	●
c — src — main.c src — minicube2.s src — opt_b.s	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.

3.2 On-Chip Peripheral Functions Used

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

- Interval timer function: 16-bit timer/event counter P (TMP)
- External interrupt input (for switch input): INTP0 (SW1)
- Output ports (for lighting LED1s): PCM3 (LED1)

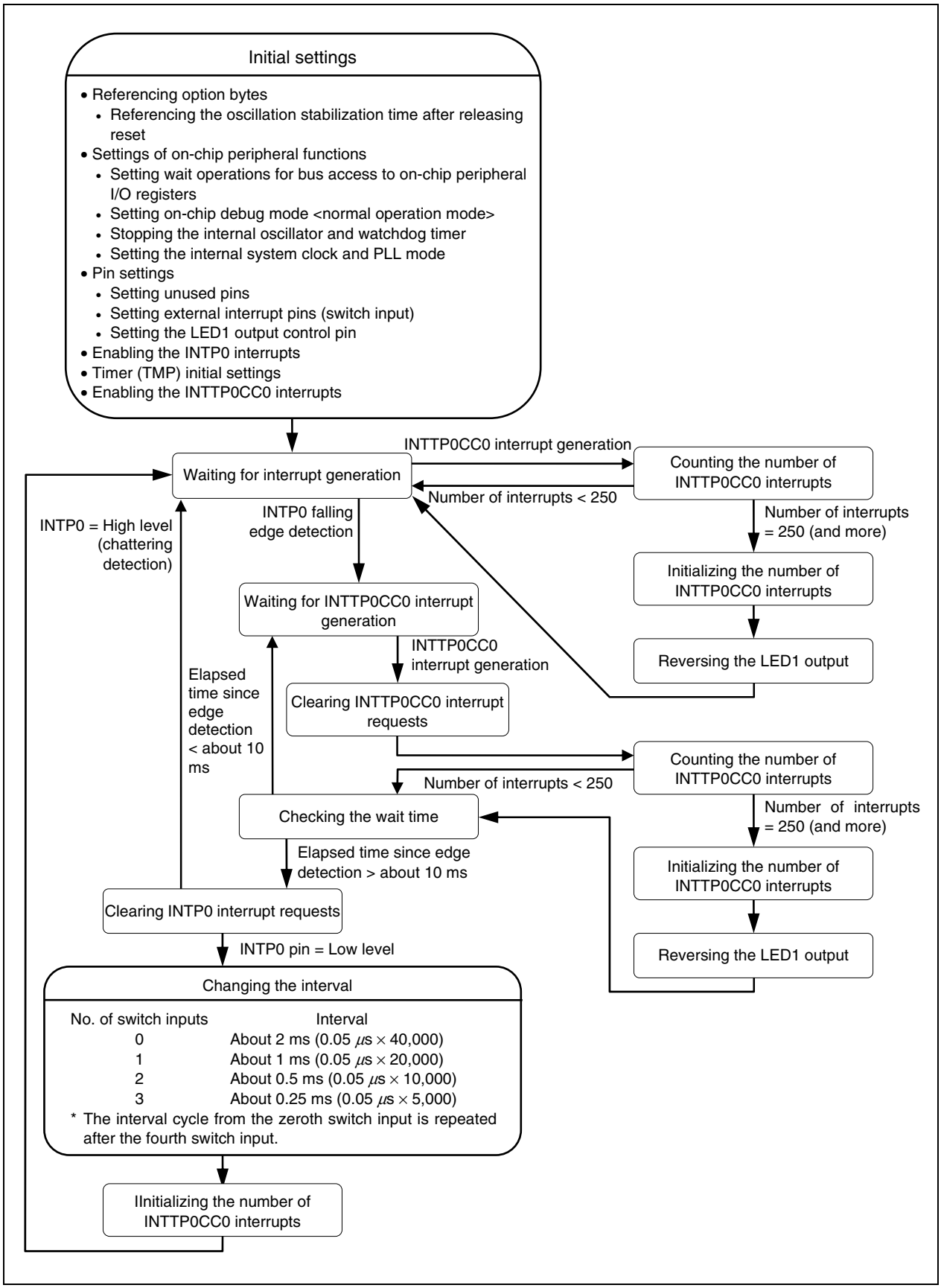
3.3 Initial Settings and Operation Overview

In the initial settings for the sample program, the clock frequency is selected, and settings for stopping the watchdog timer, setting up the I/O ports and external interrupt pins, setting up the interval timer of the 16-bit timer/event counter P (TMP), and setting up interrupts are specified.

The LED1 is made to blink at fixed cycles by using the generation of an interrupt (INTTP0CC0) of the 16-bit timer/event counter P (TMP), after completion of the initial settings.

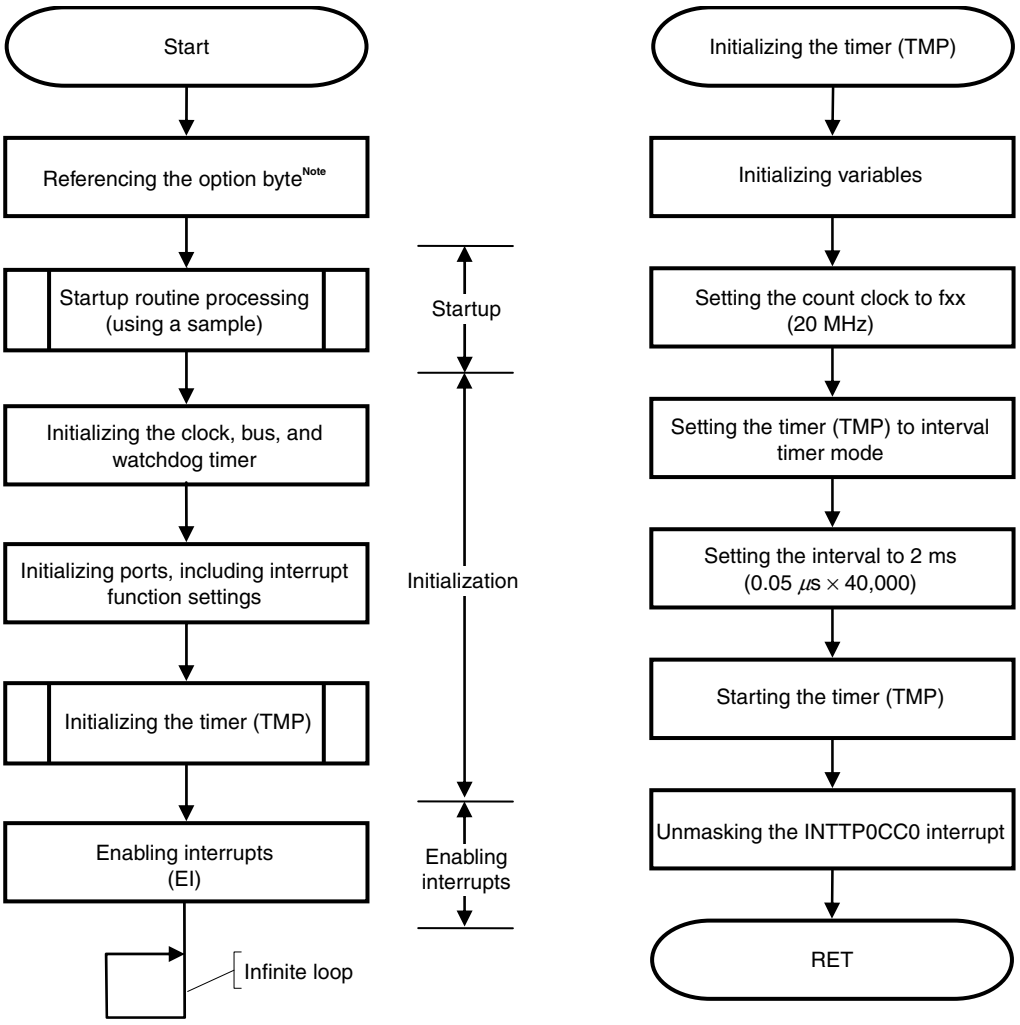
An INTP0 interrupt is serviced when the falling edge of the INTP0 pin, which is generated by switch input, is detected. Chattering is identified when INTP0 is at high level (switch is off), after 10 ms have elapsed since a fall of the INTP0 pin was detected. The blinking cycle of the LED1 is changed in accordance with the number of switch inputs when INTP0 is at low level (switch is on), after 10 ms have elapsed since the edge was detected.

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

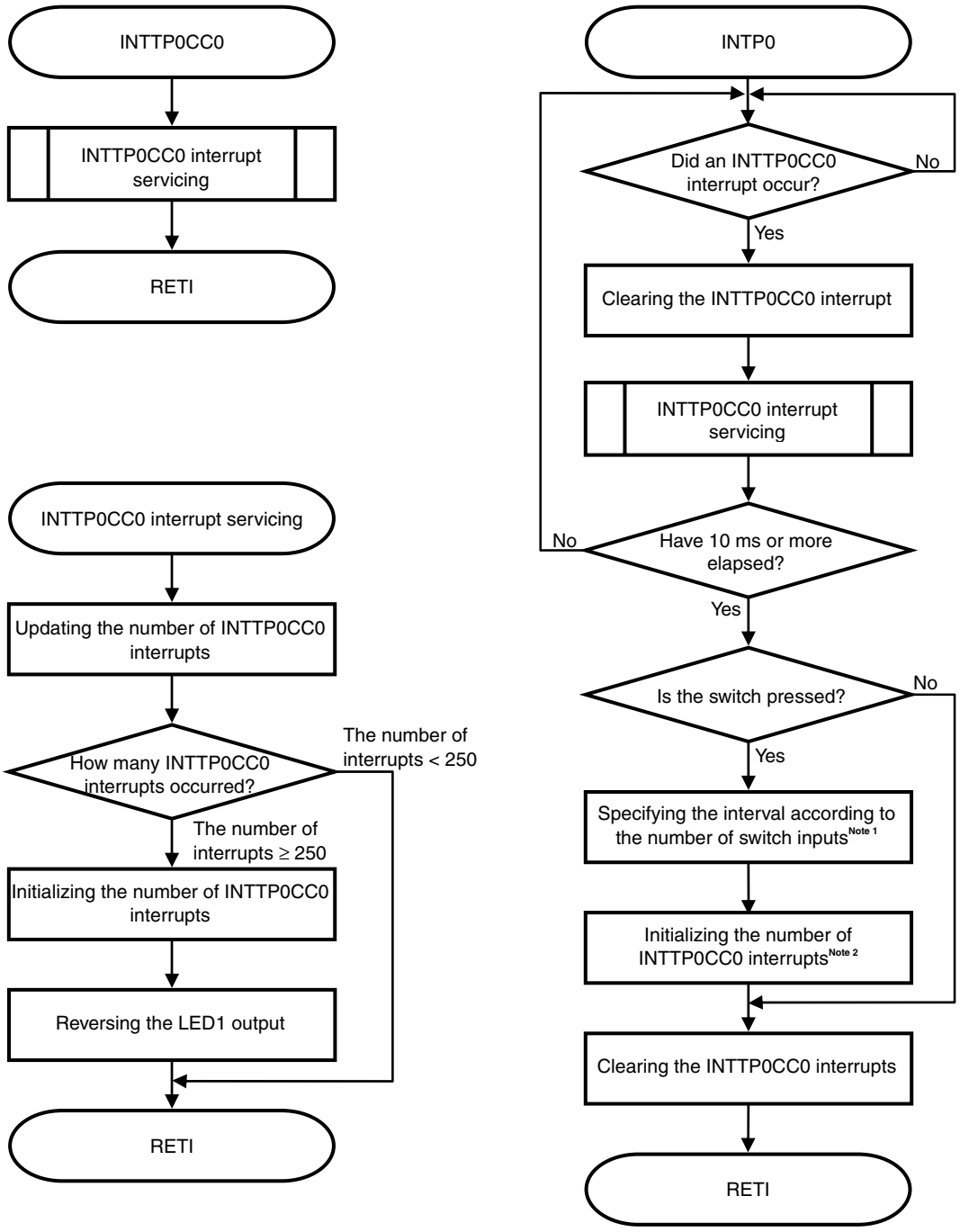


3.4 Flowcharts

Flowcharts for the sample program are shown below.



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



Notes 1. If the TPnCCR0 register value is re-written to a small value without stopping the TMP, the 16-bit counter might overflow. For details, see the user's manual.
In the sample program, the TPnCCR0 register is written in synchronization with the occurrence of an INTTPnCC0 interrupt. The write operation ends before the timer counter value reaches the minimum value specified for the TPnCCR0 register, so the counter does not overflow even if it is not stopped when changing the TPnCCR0 register value.

2. The INTTP0CC0 interrupt counter is initialized after changing the interval. If the switch is turned on and off quickly, therefore, the LED1 does not blink during that period.

[Column] Contents of the startup routine

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

Specifically, the following are performed.

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

3.5 Differences Between V850ES/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 of P1, P3, P7, P9, and PDH in I/O initialization differs.

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

3.6 Difference Between TMP and TMQ

The 16-bit timer/event counter P (TMP) and the 16-bit timer/event counter Q (TMQ) differ in the number of capture trigger pins, timer output pins, and capture compare registers.

In the sample program, the 16-bit timer/event counter P (TMP) is used. When using the 16-bit timer/event counter Q (TMQ), see **CHAPTER 4 SETTING REGISTERS** and **APPENDIX A PROGRAM LIST** for the settings.

3.7 Security ID

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

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

CHAPTER 4 SETTING REGISTERS

This chapter describes the settings of the 16-bit timer/event counter P (TMP) and the 16-bit timer/event counter Q (TMQ).

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

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

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

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

For details about extended descriptions in C, see the **CA850 C Compiler Package C Language User's Manual**.

4.1 Setting Up 16-bit Timer/Event Counter P (TMP)

The following nine registers are used to set up the 16-bit timer/event counter P (TMP):

- TMPn control register 0 (TPnCTL0)
- TMPn control register 1 (TPnCTL1)
- TMPn I/O control register 0 (TPnIOC0)
- TMPn I/O control register 1 (TPnIOC1)
- TMPn I/O control register 2 (TPnIOC2)
- TMPn option register 0 (TPnOPT0)
- TMPn capture/compare register 0 (TPnCCR0)
- TMPn capture/compare register 1 (TPnCCR1)
- TMPn counter read buffer register (TPnCNT)

Remark n = 0 to 5

Caution n = 0 in the sample program

The following eleven registers are used to set up the 16-bit timer/event counter Q (TMQ). The description given on the following pages is of TMP. Therefore, when using TMQ, read the above registers as the following:

- TMQ0 control register 0 (TQ0CTL0)
- TMQ0 control register 1 (TQ0CTL1)
- TMQ0 I/O control register 0 (TQ0IOC0)
- TMQ0 I/O control register 1 (TQ0IOC1)
- TMQ0 I/O control register 2 (TQ0IOC2)
- TMQ0 option register 0 (TQ0OPT0)
- TMQ0 capture/compare register 0 (TQ0CCR0)
- TMQ0 capture/compare register 1 (TQ0CCR1)
- TMQ0 capture/compare register 2 (TQ0CCR2)
- TMQ0 capture/compare register 3 (TQ0CCR3)
- TMQ0 counter read buffer register (TQ0CNT)

4.1.1 Setting up 16-bit timer/event counter P (TMP) operation clock

TMPn control register 0 (TPnCTL0) selects the count clock for the 16-bit timer/event counter P (TMP) and controls the counter.

The TPnCKS2 to TPnCKS0 bits must be set when the TPnCE bit is 0.

In this sample program, fxx (20 MHz) is selected by clearing the TPnCKS2 to TPnCKS0 bits to 0x00 at initialization.

After specifying the settings for the 16-bit timer/event counter P (TMP) registers, set the TPnCE bit to 1.

Figure 4-1. TPnCTL0 Register Format

TMPn control register 0 (TPnCTL0)

Address: TP0CTL0 0xFFFFF590, TP1CTL0 0xFFFFF5A0,
TP2CTL0 0xFFFFF5B0, TP3CTL0 0xFFFFF5C0,
TP4CTL0 0xFFFFF5D0, TP5CTL0 0xFFFFF5E0

7	6	5	4	3	2	1	0
TPnCE	0	0	0	0	TPnCKS2	TPnCKS1	TPnCKS0

TPnCE	TMPn operation control
0	TMPn operation disabled (TMPn reset asynchronously).
1	TMPn operation enabled. TMPn operation started.

TPnCKS2	TPnCKS1	TPnCKS0	Internal count clock selection	
			n = 0, 2, 4	n = 1, 3, 5
0	0	0	fxx	
0	0	1	fxx/2	
0	1	0	fxx/4	
0	1	1	fxx/8	
1	0	0	fxx/16	
1	0	1	fxx/32	
1	1	0	fxx/64	fxx/256
1	1	1	fxx/128	fxx/512

Caution The red values indicate the values set in this sample program.

4.1.2 Setting up 16-bit timer/event counter P (TMP) operation mode

TMPn control register 1 (TPnCTL1) specifies the operation mode of the 16-bit timer/event counter P (TMP).

In this sample program, the interval timer mode is selected by clearing the TPnMD2 to TPnMD0 bits to 000B.

Figure 4-2. TPnCTL1 Register Format

TMPn control register 1(TPnCTL1)

Address: TP0CTL1 0xFFFFF591, TP1CTL1 0xFFFFF5A1,
TP2CTL1 0xFFFFF5B1, TP3CTL1 0xFFFFF5C1,
TP4CTL1 0xFFFFF5D1, TP5CTL1 0xFFFFF5E1

7	6	5	4	3	2	1	0
0	TPnEST	TPnEEE	0	0	TPnMD2	TPnMD1	TPnMD0

TPnEST	Software trigger control
0	–
1	Generate a valid signal for external trigger input.

TPnEEE	Count clock selection
0	Disable operation with external event count input.
1	Enable operation with external event count input.

TPnMD2	TPnMD1	TPnMD0	Timer mode selection
0	0	0	Interval timer mode
0	0	1	External event count mode
0	1	0	External trigger pulse output mode
0	1	1	One-shot pulse output mode
1	0	0	PWM output mode
1	0	1	Free-running timer mode
1	1	0	Pulse width measurement mode
1	1	1	Setting prohibited

Caution The red values indicate the values set in this sample program.

4.1.3 Controlling timer output

TMPn I/O control register 0 (TPnIOC0) controls timer output.

In the interval timer mode, the TPnIOC0 register does not have to be controlled. Therefore, the register is not controlled in this sample program.

Figure 4-3. TPnIOC0 Register Format

TMPn I/O control register 0 (TPnIOC0)

Address: TP0IOC0 0xFFFFF592, TP1IOC0 0xFFFFF5A2,
TP2IOC0 0xFFFFF5B2, TP3IOC0 0xFFFFF5C2,
TP4IOC0 0xFFFFF5D2, TP5IOC0 0xFFFFF5E2

7	6	5	4	3	2	1	0
0	0	0	0	TPnOL1	TPnOE1	TPnOL0	TPnOE0

TPnOL1	TOPn1 pin output level setting
0	TOPn1 pin starts output at high level
1	TOPn1 pin starts output at low level

TPnOE1	TOPn1 pin output setting
0	Timer output disabled
1	Timer output enabled

TPnOL0	TOPn0 pin output level setting
0	TOPn0 pin starts output at high level
1	TOPn0 pin starts output at low level

TPnOE0	TOPn0 pin output setting
0	Timer output disabled
1	Timer output enabled

Caution The TPnIOC0 register is not used in this sample program.

For the 16-bit timer/event counter Q (TMQ), the TQ0OL3, TQ0OE3, TQ0OL2, and TQ0OE2 bits are assigned to bits 7 to 4 of the TQ0IOC0 register.

4.1.4 Controlling valid edge of capture trigger input signal

TMPn I/O control register 1 (TPnIOC1) controls the valid edge of the capture trigger input signal (from the TIPn0 and TIPn1 pins).

In the interval timer mode, the TPnIOC1 register does not have to be controlled. Therefore, the register is not controlled in this sample program.

Figure 4-4. TPnIOC1 Register Format

TMPn I/O control register 1(TPnIOC1)

Address: TP0IOC1 0xFFFFF593, TP1IOC1 0xFFFFF5A3,
TP2IOC1 0xFFFFF5B3, TP3IOC1 0xFFFFF5C3,
TP4IOC1 0xFFFFF5D3, TP5IOC1 0xFFFFF5E3

7	6	5	4	3	2	1	0
0	0	0	0	TPnIS3	TPnIS2	TPnIS1	TPnIS0

TPnIS3	TPnIS2	Capture trigger input signal (TIPn1 pin) valid edge setting
0	0	No edge detection
0	1	Detection of rising edge
1	0	Detection of falling edge
1	1	Detection of both edges

TPnIS1	TPnIS0	Capture trigger input signal (TIPn0 pin) valid edge setting
0	0	No edge detection
0	1	Detection of rising edge
1	0	Detection of falling edge
1	1	Detection of both edges

Caution The TPnIOC1 register is not used in this sample program.

For the 16-bit timer/event counter Q (TMQ), the TQ0IS7, TQ0IS6, TQ0IS5 and TQ0IS4 bits are assigned to bits 7 to 4 of the TQ0IOC1 register.

4.1.5 Controlling external input signals

TMPn I/O control register 2 (TPnIOC2) controls the valid edge of the external event count input signal (from the TIPn0 pin) and external trigger input signal (from the TIPn0 pin).

In the interval timer mode, the TPnIOC2 register does not have to be controlled. Therefore, the register is not controlled in this sample program.

Figure 4-5. TPnIOC2 Register Format

TMPn I/O control register 2 (TPnIOC2)

Address: TP0IOC2 0xFFFFF594, TP1IOC2 0xFFFFF5A4,
TP2IOC2 0xFFFFF5B4, TP3IOC2 0xFFFFF5C4,
TP4IOC2 0xFFFFF5D4, TP5IOC2 0xFFFFF5E4

7	6	5	4	3	2	1	0
0	0	0	0	TPnEES1	TPnEES0	TPnETS1	TPnETS0

TPnEES1	TPnESS0	External event count input signal (TIPn0 pin) valid edge setting
0	0	No edge detection
0	1	Detection of rising edge
1	0	Detection of falling edge
1	1	Detection of both edges

TPnETS1	TPnETS0	External trigger input signal (TIPn0 pin) valid edge setting
0	0	No edge detection
0	1	Detection of rising edge
1	0	Detection of falling edge
1	1	Detection of both edges

Caution The TPnIOC2 register is not used in this sample program.

4.1.6 Controlling capture/compare operation

TMPn option register 0 (TPnOPT0) controls the capture/compare operation setting and overflow detection.

In the interval timer mode, the TPnOPT0 register does not have to be controlled. Therefore, the register is not controlled in this sample program.

Figure 4-6. TPnOPT0 Register Format

TMPn option register 0 (TPnOPT0)

Address: TP0OPT0 0xFFFFF595, TP1OPT0 0xFFFFF5A5,
TP2OPT0 0xFFFFF5B5, TP3OPT0 0xFFFFF5C5,
TP4OPT0 0xFFFFF5D5, TP5OPT0 0xFFFFF5E5

7	6	5	4	3	2	1	0
0	0	TPnCCS1	TPnCCS0	0	0	0	TPnOVF

TPnCCS1	TPnCCR1 register capture/compare selection
0	Compare register selected
1	Capture register selected

TPnCCS0	TPnCCR0 register capture/compare selection
0	Compare register selected
1	Capture register selected

TPnOVF	TMPn overflow detection flag
Set (1)	Overflow occurred
Reset (0)	TPnOVF bit 0 written or TPnCTL0.TPnCE bit = 0

Caution The TPnOPT0 register is not used in this sample program.

For the 16-bit timer/event counter Q (TMQ), the TQ0CCS3, TQ0CCS2, TQ0CCS1 and TQ0CCS0 bits are assigned to bits 7 to 4 of the TQ0OPT0 register.

4.1.7 Controlling intervals

Intervals can be controlled by using TMPn capture/compare register 0 (TPnCCR0).

When the counter value of TMP reaches the value specified for the TPnCCR0 register, an INTTPnCC0 interrupt occurs. When the TMP value reaches the TPnCCR1 register value, an INTTPnCC1 interrupt occurs.

In this sample program, the LED1 blinking cycle must be measured as intervals, and the TPnCCR0 register is used for this. The TPnCCR1 register is not used.

- Interval = (N + 1)/f_{CNT}

Remark N: The value specified for the TPnCCR0 or TPnCCR1 register
f_{CNT}: The count clock frequency of the 16-bit timer/event counter

Figure 4-7. TPnCCR0 Register Format

TMPn capture/compare register 0 (TPnCCR0)
Address: TP0CCR0 0xFFFFF596, TP1CCR0 0xFFFFF5A6,
TP2CCR0 0xFFFFF5B6, TP3CCR0 0xFFFFF5C6,
TP4CCR0 0xFFFFF5D6, TP5CCR0 0xFFFFF5E6

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Figure 4-8. TPnCCR1 Register Format

TMPn capture/compare register 1 (TPnCCR1)
Address: TP0CCR1 0xFFFFF598, TP1CCR1 0xFFFFF5A8,
TP2CCR1 0xFFFFF5B8, TP3CCR1 0xFFFFF5C8,
TP4CCR1 0xFFFFF5D8, TP5CCR1 0xFFFFF5E8

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Caution The TPnCCR1 register is not used in this sample program.

For the 16-bit timer/event counter Q (TMQ), four capture/compare registers (TQ0CCR0 to TQ0CCR3) are provided. These registers are used in the same way as the TPnCCR0 and TPnCCR1 registers.

4.1.8 Referencing timer count value

The TPnCNT register is a read buffer register from which a 16-bit counter value can be read.

The current counter value can be read by reading this register while the timer is operating (TPnCTL0.TPnCE bit = 1).

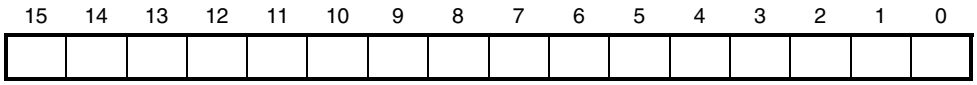
If this register is read while the timer is stopped (TPnCTL0.TPnCE bit = 0), 0x0000 is returned.

In the sample program, this register is not used because counter values do not have to be referenced.

Figure 4-9. TPnCNT Register Format

TMPn counter read buffer register (TPnCNT)

Address: TP0CNT 0xFFFFF59A, TP1CNT 0xFFFFF5AA,
TP2CNT 0xFFFFF5BA, TP3CNT 0xFFFFF5CA,
TP4CNT 0xFFFFF5DA, TP5CNT 0xFFFFF5EA



[Example 1] When starting the timer with the following conditions (same as the sample program):

- 16-bit timer/event counter P (TMP) in interval timer mode
- Frequency: fxx (20 MHz)
- Interval: 2 ms

- Setup procedure

<1> Specifies the count clock fxx.
<2> Specifies the interval timer mode.
<3> Specifies a value for the compare register.
<4> Starts the timer (TMP).
<5> Unmasks an interrupt.

- Program example (same as the sample program)

```
#pragma interrupt INTTP0CC0 f_int_inttmp0    /* Specifies the timer interrupt (INTTP0CC0) handler.*/

const unsigned short COMP_TBL [] ={(40000-1),/* 2 ms interval compare value     */
                                   (20000-1), /* 1 ms interval compare value    */
                                   (10000-1), /* 0.5 ms interval compare value  */
                                   (5000-1)}; /* 0.25 ms interval compare value */
static void f_init_timer( void )
{
    /* Variable initialization */
    g_blk_ptn_cnt = 0;        /* Initializes the compare register setting pattern storage variable. */
    g_led_int_cnt = 0;        /* Initializes the timer interrupt (INTTP0CC0) count variable */

    /* Timer settings */
    TP0CTL0 = 0x00;                  /* Count clock = fxx (20 MHz)                    */ <1>
    TP0CTL1 = 0x00;                  /* Specifies the interval timer mode.            */ <2>
    TP0CCR0 = COMP_TBL[g_blk_ptn_cnt]; /* Specifies a value for the compare register. */ <3>
    TP0CE = 1;                       /* Starts the timer (TMP).                       */ <4>

    /* Interrupt settings */
    TP0CCIC0 = 0x07;          /* Sets the priority of INTTP0CC0 to level 7 and unmasks INTTP0CC0 */ <5>

    return;
}

__interrupt
void f_int_inttmp0( void )
{
    f_int_subtmp0();
    return;                   /* reti by using the __interrupt modifier */
}
```

Registers the f_int_inttmp0 function as the interrupt handler.

Specifies a 2 ms interval by assigning the value of CMP_TBL[0] (40000 – 1) to TP0CCR0.

Starts interrupt servicing by generating the INTTP0CC0 interrupt.

[Example 2] When starting timer operation with the following conditions:

- 16-bit timer/event counter Q (TMQ): Interval timer mode
- Frequency: fxx (20 MHz)
- Interval: 2 ms

- Setup procedure

<1> Specifies the count clock fxx.
<2> Specifies the interval timer mode.
<3> Specifies a value for the compare register.
<4> Starts the timer (TMQ).
<5> Unmasks an interrupt.

- Program example

```
#pragma interrupt INTTQ0CC0 f_int_inttmq0    /* Specifies the timer interrupt (INTTQ0CC0) handler. */
const unsigned short COMP_TBL[] ={(40000-1),/* 2 ms interval compare value */
                                  (20000-1),/* 1 ms interval compare value */
                                  (10000-1),/* 0.5 ms interval compare value      */
                                  (5000-1)};/* 0.25 ms interval compare value     */
static void f_init_timer( void )
{
    /* Variable initialization */
    g_blk_ptn_cnt = 0;        /* Initializes the compare register setting pattern storage variable. */
    g_led_int_cnt = 0;        /* Initializes the timer interrupt (INTTQ0CC0) count variable */

    /* Timer settings */
    TQ0CTL0 = 0x00;           /* Count clock = fxx (20 MHz)                          */ } <1>
    TQ0CTL1 = 0x00;           /* Specifies the interval timer mode.                  */ } <2>
    TQ0CCR0 = COMP_TBL[g_blk_ptn_cnt]; /* Specifies a value for the compare register. */ } <3>
    TQ0CE = 1;                /* Starts the timer (TMP).                             */ } <4>
    /* Interrupt settings */
    TQ0CCIC0 = 0x07;          /* Sets the priority of INTTQ0CC0 to level 7 and unmasks INTTQ0CC0 */ } <5>

    return;
}

__interrupt
void f_int_inttmq0( void )
{
f_int_subtmp0();

return;                       /* reti by using the __interrupt modifier */
}
```

Specifies INTTQ0CC0 as an interrupt source.

Registers the f_int_inttmp0 function as the interrupt handler.

Specifies a 2 ms interval by assigning the value of CMP_TBL[0] (40000 – 1) to TQ0CCR0.

Starts interrupt servicing by generating the INTTQ0CC0 interrupt.

4.2 Setting LED1 Blinking Cycle and Chattering Detection Time

The LED1 blinking cycle and chattering detection time are set as follows in this sample program.

(1) Setting the LED1 blinking cycle

The LED1 output is reversed every 250 generations of 16-bit timer/event counter P (TMP) interrupts (INTTP0CC0) in this sample program.

- Interrupt cycle (interval time) = (N + 1)/f_{CNT}
- LED1 output reversal cycle = Interrupt cycle × Number of interrupts
- LED1 blinking cycle = LED1 output reversal cycle × 2

Remark N: TP0CCR0 register setting value
f_{CNT}: Count clock frequency of 16-bit timer/event counter P (TMP)

Calculation example: The following values result when the TP0CCR0 register setting value is 39999 (during operation at count clock frequency = 20 kHz).

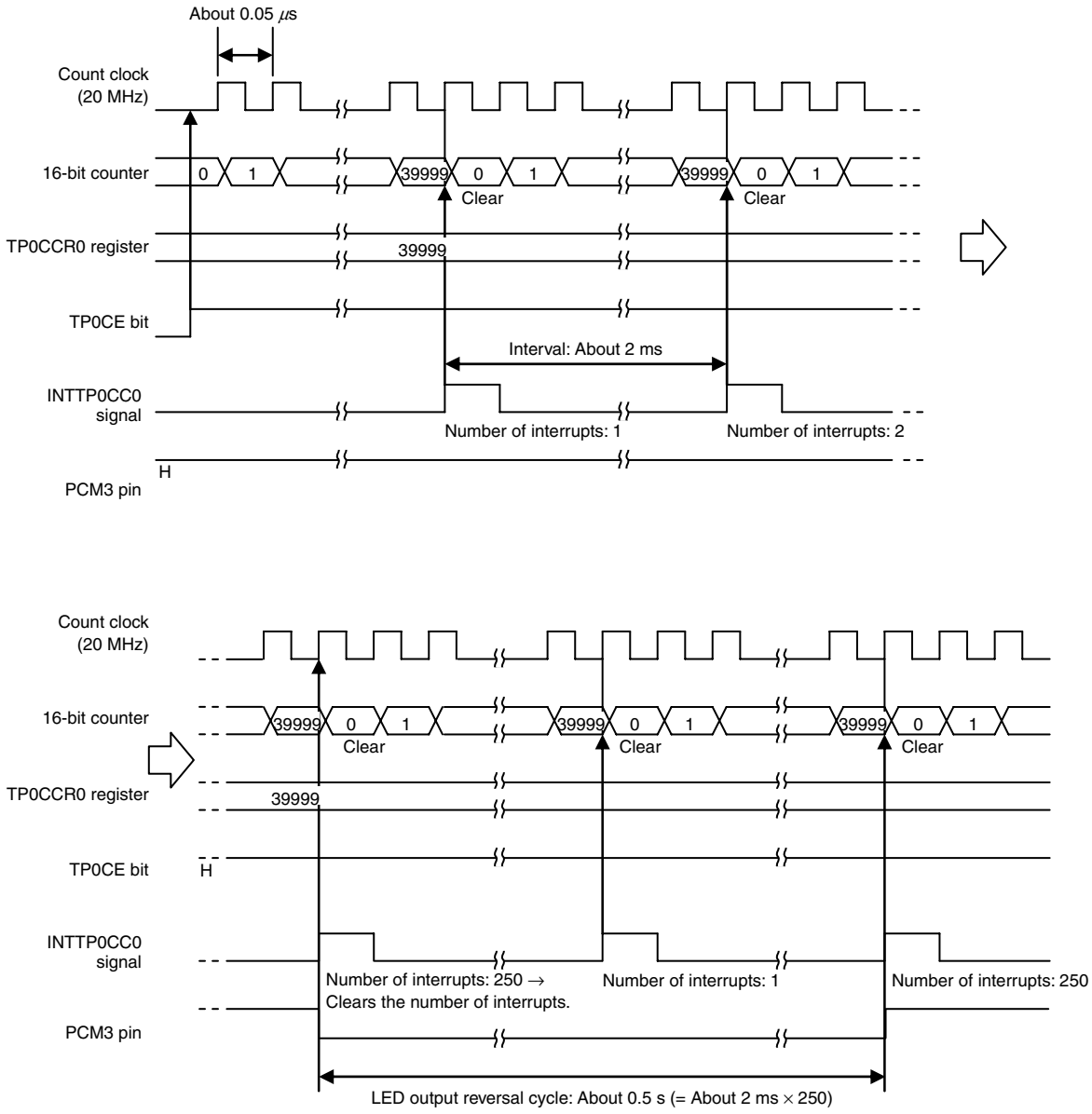
- Interrupt cycle (interval time) = (N + 1)/f_{CNT} = (39999 + 1)/20 MHz = 2 ms
- LED1 output reversal cycle = Interrupt cycle × Number of interrupts = 2 ms × 250 = 500 ms
- LED1 blinking cycle = LED1 output reversal cycle × 2 = 500 ms × 2 = 1 s

Furthermore, the TP0CCR0 register setting value is changed in accordance with the number of switch inputs, and the LED1 blinking cycle is changed.

Number of Switch Inputs[Note]	TP0CCR0 Register Setting Value	Interrupt Cycle	LED1 Blinking Cycle
0	39999	About 2 ms ((39999 + 1)/20 MHz)	About 1 s (About 2 ms × 250 × 2)
1	19999	About 1 ms ((19999 + 1)/20 MHz)	About 0.5 s (About 1 ms × 250 × 2)
2	9999	About 0.5 ms ((9999 + 1)/20 MHz)	About 0.25 s (About 0.5 ms × 250 × 2)
3	4999	About 0.25 ms ((4999 + 1)/20 MHz)	About 0.125 s (About 0.25 ms × 250 × 2)

Note The blinking cycle from the zeroth switch input is repeated after the fourth switch input.

Figure 4-10. Timing Chart Example of LED1 Blinking Cycle (When LED1 Blinks at Cycles of About 1 s)



Remark The TP0CCR0 register setting value is 19999, 9999, and 4999 when the LED1 blinks at respective cycles of about 1/2 s, 1/4 s, and 1/8 s.

(2) Setting the chattering detection time

The number of 16-bit timer/event counter P (TMP) interrupts (INTTP0CC0) generated is counted to eliminate chattering of 10 ms or less in order to reduce chattering during switch input (INTP0 interrupt generation) in this sample program.

By using INTTP0CC0 interrupts for chattering elimination, INTTP0CC0 interrupts can be continuously counted even during chattering detection. Consequently, offsets of the LED1 blinking cycle, which are caused by chattering, can be suppressed.

• Chattering detection time (T_C) = $T' + T \times (M - 1)$

Remark T: INTTP0CC0 interrupt cycle

T': Time from the start of INTP0 edge detection until the first INTTP0CC0 is generated after INTP0 edge detection ($0 < T' \leq T$)

M: Number of INTTP0CC0 interrupts after INTP0 edge detection

When set such that $T \times (M - 1) = 10$ ms,

$T_C = T' + 10$ ms

$0 < T' \leq T$, therefore,

$10 \text{ ms} < T_C \leq T + 10 \text{ ms}$

↓

Chattering detection time (T_C) > 10 ms

Calculation example: When the interrupt cycle (T) is 2 ms (see the calculation example in 4.2 (1) Setting the LED1 blinking cycle), and the number of INTTP0CC0 interrupts after INTP0 edge detection (M) is 6

$T_C = T' + T \times (M - 1)$

$= T' + 2 \text{ ms} \times (6 - 1)$

$= T' + 10$ ms

$0 < T' \leq 2$ ms, therefore,

$10 \text{ ms} < T_C \leq 12$ ms

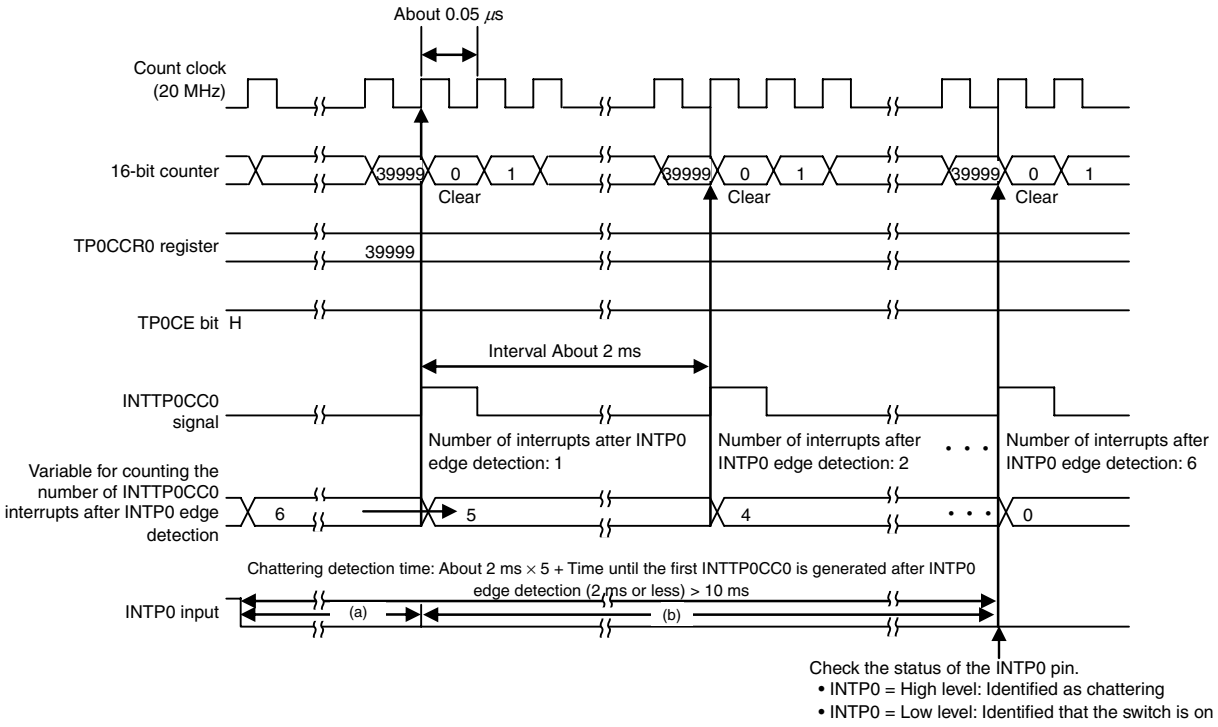
↓

Chattering detection time (T_C) > 10 ms

The following table shows the correspondence between the interrupt cycles during switch input and the number of INTTP0CC0 interrupts after INTP0 edge detection in this sample program.

LED1 Blinking Cycle	Interrupt Cycle	Number of INTTP0CC0 Interrupts After INTP0 Edge Detection	Chattering Detection Time
About 1 s	About 2 ms	6	$10 \text{ ms} < T_C \leq 12$ ms
About 0.5 s	About 1 ms	11	$10 \text{ ms} < T_C \leq 11$ ms
About 0.25 s	About 0.50 ms	21	$10 \text{ ms} < T_C \leq 10.50$ ms
About 0.125 s	About 0.25 ms	41	$10 \text{ ms} < T_C \leq 10.25$ ms

Figure 4-11. Timing Chart Example of Chattering Detection (When LED1 Blinks at Cycles of About 1 s During Switch Input)



(a) Time until the first INTTP0CC0 is generated after INTP0 edge detection (2 ms or less)

(b) About 2 ms × 5

Remark The variable for counting the number of INTTP0CC0 interrupts after INTP0 edge detection depends on the LED1 blinking cycle during switch input. The variable is 11, 21, and 41, when the LED1 blinks at respective cycles of about 1/2 s, 1/4 s, and 1/8 s.

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 User's Manual	PDF
CA850 Ver.3.20 C Compiler Package C Language User's Manual	PDF
CA850 Ver.3.20 C Compiler Package Assembly Language User's Manual	PDF
CA850 Ver.3.20 C Compiler Package Link Directive User's Manual	PDF
V850ES Architecture User's Manual	PDF
QB-MINI2 On-Chip Debug Emulator with Programming Function User's Manual	PDF
ID850QB Ver. 3.40 Integrated Debugger Operation User's Manual	PDF

APPENDIX A PROGRAM LIST

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

● opt_b.s

```
#-------------------------------------------------------------------------------
#
#    NEC Electronics     V850ES/Jx3-L microcontroller
#
#-------------------------------------------------------------------------------
#    V850ES/JG3-L JF3-L  sample program
#-------------------------------------------------------------------------------
#    Interval Timer Mode
#-------------------------------------------------------------------------------
#[History]
#    2009.1.--   Released
#-------------------------------------------------------------------------------
#[Overview]
#    This sample program sets the option byte.
#-------------------------------------------------------------------------------

    .section "OPTION_BYTES"
    .byte 0b00000101 -- 0x7a (5MHz: Sets the oscillation stabilization time to 6.554ms)
    .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

```
#-------------------------------------------------------------------------------
#
#    NEC Electronics     V850ES/Jx3-L microcontroller
#
#-------------------------------------------------------------------------------
#    V850ES/JG3-L JF3-L  sample program
#-------------------------------------------------------------------------------
#    Interval Timer Mode
#-------------------------------------------------------------------------------
#[History]
#    2009.1.--   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_LVI.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         : uPD70F3738 (C:\Program Files\NEC Electronics Tools\DEV\DF3738.800)
#   Internal RAM   : 0x3ffb000 - 0x3ffefff
#
#   NOTICE:
#        Allocation of SCONST, CONST and TEXT depends on the user program.
#
#        If interrupt handler(s) are specified in the user program then
#        the interrupt handler(s) are allocated from address 0 and
#        SCONST, CONST and TEXT are allocated after the interrupt handler(s).

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

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

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

### For MINICUBE2 ###
MROMSEG : !LOAD ?R V0x03F800{
      MonitorROM = $PROGBITS ?A MonitorROM;
};
```

0x01F800 for products with 128 KB internal ROM

Difference from the default link directive file (additional code)

A reserved area for MINICUBE2 is secured.

```
SIDATA  : !LOAD ?RW V0x3ffb000 {
        .tidata.byte    = $PROGBITS  ?AW .tidata.byte;
        .tibss.byte     = $NOBITS    ?AW .tibss.byte;
        .tidata.word    = $PROGBITS  ?AW .tidata.word;
        .tibss.word     = $NOBITS    ?AW .tibss.word;
        .tidata         = $PROGBITS  ?AW .tidata;
        .tibss          = $NOBITS    ?AW .tibss;
        .sidata         = $PROGBITS  ?AW .sidata;
        .sibss          = $NOBITS    ?AW .sibss;
};

DATA    : !LOAD ?RW V0x3ffb100 {
        .data           = $PROGBITS  ?AW  .data;
        .sdata          = $PROGBITS  ?AWG .sdata;
        .sbss           = $NOBITS    ?AWG .sbss;
        .bss            = $NOBITS    ?AW  .bss;
};

### For MINICUBE2 ###
MRAMSEG : !LOAD ?RW V0x03FFEFF0{
MonitorRAM = $NOBITS ?AW MonitorRAM;
};

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

Difference from the default link directive file (additional code)

A reserved area for MINICUBE2 is secured.

● main.c

```
/*-------------------------------------------------------------------------*/
/*
/*    NEC Electronics     V850ES/Jx3-L microcontroller
/*
/*-------------------------------------------------------------------------*/
/*    V850ES/JG3-L sample program
/*-------------------------------------------------------------------------*/
/*    Interval Timer Mode
/*-------------------------------------------------------------------------*/
/*[History]
/*    2009.1.--   Released
/*-------------------------------------------------------------------------*/
/*[Overview]
/*    This sample program shows an example of using the interval timer mode
/*    of the 16-bit timer/event counter (TMP).
/*    By using 16-bit timer/event counter (TMP) interrupts,
/*    the PCM3 pin output is inverted to make the LED1 blink.
/*    The blinking cycle of the LED1 is changed by re-writing the compare register
/*    of the timer when a switch input interrupt occurs.
/*
/*
/*    <Main settings>
/*    • Using pragma directives to enable setting up the interrupt handler and specifying
       peripheral I/O register names
/*    • Declaring prototypes
/*    • Defining the compare register setting table and the 10 ms wait count value table
/*    • Declaring the compare register setting pattern storage variable and the timer
       interrupt (INTTP0CC0) counter variable
/*    • Setting up a bus wait for on-chip peripheral I/O registers, stopping the watchdog timer,
       and setting up the clock
/*    • Initializing unused ports
/*    • Initializing switch interrupt (INTP0) ports (falling edge) and LED1 output ports
/*    • Initializing the timer (TMP)
/*    <Switch interrupt (INTP0) servicing>
/*    • Updating the compare register setting pattern
/*      (chattering elimination time at switch input = 10 ms)
/*    <Timer interrupt (INTTP0CC0) servicing>
/*    • Making the LED1 blink
```

```
/*
/* <Switch input count and LED1 blinking pattern>
/*
/* +-----------------------------------------+
/* | Switch input count | LED1 blinking cycle |
/* |     (P03/INTP0)    |        (PCM3)       |
/* |--------------------|---------------------|
/* |          0         |           1 s       |
/* |          1         |         1/2 s       |
/* |          2         |         1/4 s       |
/* |          3         |         1/8 s       |
/* +-----------------------------------------+
/*   *The interval cycle from the zeroth switch input is repeated after the fourth switch input.
/*
/*
/*[I/O port settings]
/*
/* Input port: P03(INTP0)
/* Output port: PCM3
/* Unused ports: P02, P04 to P06, P10 to P11, P30 to P39, P50 to P55, P70 to P711, P90 to P915,
/*               PCM0 to PCM2, PCT0, PCT1, PCT4, PCT6, PDH0 to PDH5, PDL0 to PDL15
/*               *Preset all unused ports as output ports (low-level output).
/*
/*--------------------------------------------------------------------------*/

/*--------------------------*/
/*    pragma directives     */
/*--------------------------*/
#pragma ioreg                               /* Enables writing to peripheral I/O registers. */
#pragma interrupt INTP0     f_int_intp0     /* Specifies the switch interrupt (INTP0) handler.*/
#pragma interrupt INTTP0CC0 f_int_inttmp0   /* Specifies the timer interrupt (INTTP0CC0) handler.*/

/*--------------------------*/
/*   Prototype definitions  */
/*--------------------------*/
void main( void );                          /* Main function                                */
static    void f_init( void );              /* Initialization function                      */
static    void f_init_clk_bus_wdt2( void );/* Clock bus WDT initialization function        */
static    void f_init_port_func( void );    /* Port/alternate-function initialization function  */
static    void f_init_timer( void );        /* Timer initialization function                */
static    void f_int_subtmp0( void );       /* Timer interrupt (INTTP0CC0) subroutine       */
```

When using the TMQ, specify the INTTQ0CC0.

```
/*---------------------------------------*/
/* Tables declarations and initialization */
/*---------------------------------------*/
const unsigned short COMP_TBL[] ={(40000-1), /* 2 ms interval compare value            */
                                  (20000-1), /* 1 ms interval compare value            */
                                  (10000-1), /* 0.5 ms interval compare value          */
                                  (5000-1)}; /* 0.25 ms interval compare value         */

const unsigned char WAIT_TBL[] = {(5+1),   /* 10 ms wait adjustment constant definition
                                              (for a 2 ms interval)*/
                                  (10+1),  /* 10 ms wait adjustment constant definition
                                              (for a 1 ms interval)*/
                                  (20+1),  /* 10 ms wait adjustment constant definition
                                              (for a 0.5 ms interval)*/
                                  (40+1)};  /* 10 ms wait adjustment constant definition
                                              (for a 0.25 ms interval)*/


/*-------------------------------*/
/* Global variable declarations */
/*-------------------------------*/
static unsigned char g_blk_ptn_cnt;      /* Saves the compare register setting pattern. */
static unsigned char g_led_int_cnt;      /* Counts the number of timer interrupts (INTTP0CC0).*/


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

    __EI();                                  /* Enables interrupts.             */

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

    return;
}
```

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

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

    f_init_timer();                     /* Specifies initial settings for the timer (TMP).*/

    return;
}

/*-----------------------------*/
/* Initializing clock bus WDT2 */
/*-----------------------------*/
static void f_init_clk_bus_wdt2( void )
{
    VSWC   = 0x01;                  /* Sets a bus wait for on-chip peripheral I/O registers. */
                                    /* Specifies normal operation mode for OCDM. */
#pragma asm
    st.b    r0, PRCMD
    st.b    r0, OCDM
#pragma endasm

    RSTOP  = 1;                     /* Stops the internal oscillator. */
    WDTM2  = 0x00;                  /* Stops the watchdog timer. */

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

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

    return;
}
```

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

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

    P1    = 0x00;                        /* Sets P10 and P11 to output low level signal. */
    PM1   = 0xFC;
    P3    = 0x0000;                      /* Sets P30 to P39 to output low level signal.  */
    PM3   = 0xFC00;
    PMC3  = 0x0000;

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

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

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

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

    PCM   = 0x08;                        /* Sets PCM0 to PCM2 to output low level signals and
    PMCM  = 0xF0;                           specifies the turn-off pattern for PCM3.*/
    PMCCM = 0x00;

    PCT   = 0x00;                        /* Sets PCT0, PCT1, PCT4, and PCT6 to output low
    PMCT  = 0xAC;                           level signal. */
    PMCCT = 0x00;
```

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

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

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

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

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

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

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

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

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

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

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

    return;
}

/*---------------------------*/
/*   Timer initial settings   */
/*---------------------------*/
static void f_init_timer( void )
{
    /* Timer settings */
g_blk_ptn_cnt = 0;          /* Initializes the compare register setting pattern storage variable.*/
g_led_int_cnt = 0;          /* Initializes the timer interrupt (INTTP0CC0) counter variable.*/

    /* Timer settings */
    TP0CTL0 = 0x00;                     /* Count clock = fxx (20 MHz)                  */
    TP0CTL1 = 0x00;                     /* Specifies the interval timer mode.          */
    TP0CCR0 = COMP_TBL[g_blk_ptn_cnt];  /* Specifies a value for the compare register. */
    TP0CE   = 1;                        /* Starts the timer (TMP).                     */
/* Caution: The following registers are not used in the interval timer mode: */
/*           <Unused registers>     */
/*             - TP0ICC0 register   */
/*             - TP0ICC1 register   */
/*             - TP0ICC2 register   */
/*             - TP0OPT0 register   */
/*             - TP0CNT register    */
    /* Interrupt settings */
    TP0CCIC0 = 0x07;            /* Sets the priority of INTTP0CC0 to level 7 and unmasks INTTP0CC0. */
    return;
}
```

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

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

When using the TMQ, specify values for the TQ0xxx registers.

When using the TMQ, set a value to the TQ0CCIC0 register.

```
/*****************************/
/*   Switch (INTP0) module   */
/*****************************/
__interrupt
void f_int_intp0( void )
{
    unsigned int loop_wait;

    /* 10 ms wait for chattering elimination */
    for( loop_wait = WAIT_TBL[g_blk_ptn_cnt]; loop_wait > 0; loop_wait-- )
    {
        while( !TP0CCIF0 )          /* Checks the timer interrupt (INTTP0CC0) request flag. */
        {
            __nop();
        }
        TP0CCIF0 = 0;               /* Clears the timer interrupt (INTTP0CC0) request flag. */
        f_int_subtmp0();            /* Services the timer interrupt (INTTP0CC0). */
    }
    if( ( P0 & 0x08 ) == 0x00 )     /* Confirms whether SW1 is pressed after waiting. */
    {
        g_blk_ptn_cnt++;            /* Changes the compare register setting pattern (four patterns). */
        g_blk_ptn_cnt &= 3;

        TP0CCR0 = COMP_TBL[g_blk_ptn_cnt]; /* Changes the compare register value.*/


        g_led_int_cnt = 0;          /* Clears the number of timer interrupts (INTTP0CC0).*/
    }

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

    return;                         /* reti by using the __interrupt modifier */
}
```

When using the TMQ, the TQ0CCIF0 bit value is monitored.

When using the TMQ, sets a value to the TQ0CCIF0 bit.

When using the TMQ, a value is specified for the TQ0CCR0 register.

```
/*************************************/
/*  Timer interrupt (INTTP0CC0) module  */
/*************************************/
__interrupt
void f_int_inttmp0( void )
{
    f_int_subtmp0();

    return;                         /* reti by using the __interrupt modifier   */
}


/*****************************************/
/* Timer interrupt (INTTP0CC0) subroutine */
/*****************************************/
static void f_int_subtmp0( void )
{
    g_led_int_cnt++;                /* Updates the number of timer interrupts (INTTP0CC0). */

    if( g_led_int_cnt >= 250 )      /* Checks the number of timer interrupts (INTTP0CC0). */
    {
        g_led_int_cnt = 0;          /* Initializes the timer interrupt (INTTP0CC0) counter variable. */
        PCM.3 ^= 1;                 /* Reverses LED1. */
    }

    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