

To our customers,

---

## Old Company Name in Catalogs and Other Documents

---

On April 1<sup>st</sup>, 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 1<sup>st</sup>, 2010  
Renesas Electronics Corporation

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

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

## Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: “Standard”, “High Quality”, and “Specific”. The recommended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as “Specific” without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as “Specific” or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is “Standard” unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.
  - “Standard”: Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.
  - “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.
  - “Specific”: Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

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

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

# Application Note

## 78K0S/Kx1+

### Sample Program (16-bit Timer/Event Counter 00)

#### PPG Output

This document describes an operation overview of the sample program and how to use it, as well as how to set and use the PPG output function of 16-bit timer/event counter 00. In the sample program, a rectangular wave with an arbitrary cycle and pulse width is output by using the PPG output function of 16-bit timer/event counter 00. Furthermore, the LED brightness is changed by changing the PPG output duty in accordance with the number of switch inputs.

#### Target devices

- 78K0S/KA1+ microcontroller
- 78K0S/KB1+ microcontroller
- 78K0S/KU1+ microcontroller
- 78K0S/KY1+ microcontroller

#### CONTENTS

<b>CHAPTER 1 OVERVIEW</b> .....	<b>3</b>
1.1 Main Contents of the Initial Settings.....	3
1.2 Contents Following the Main Loop.....	3
<b>CHAPTER 2 CIRCUIT DIAGRAM</b> .....	<b>5</b>
2.1 Circuit Diagram.....	5
2.2 Peripheral Hardware.....	5
<b>CHAPTER 3 SOFTWARE</b> .....	<b>6</b>
3.1 File Configuration.....	6
3.2 Internal Peripheral Functions to Be Used.....	7
3.3 Initial Settings and Operation Overview.....	7
3.4 Flow Charts.....	9
<b>CHAPTER 4 SETTING METHODS</b> .....	<b>10</b>
4.1 Setting the PPG Output Function of 16-bit Timer/Event Counter 00.....	10
4.2 Setting and Changing the PPG Output Duty.....	22
4.2.1 Setting the PPG output duty.....	22
4.2.2 Changing the CR010 setting value (PPG output duty).....	23
4.3 Setting the Chattering Detection Time.....	27
<b>CHAPTER 5 OPERATION CHECK USING SYSTEM SIMULATOR SM+ ....</b>	<b>29</b>
5.1 Building the Sample Program.....	29
5.2 Operation with SM+.....	31
<b>CHAPTER 6 RELATED DOCUMENTS</b> .....	<b>36</b>
<b>APPENDIX A PROGRAM LIST</b> .....	<b>37</b>
<b>APPENDIX B REVISION HISTORY</b> .....	<b>53</b>

• **The information in this document is current as of July, 2008. The information is subject to change without notice. For actual design-in, refer to the latest publications of NEC Electronics data sheets or data books, etc., for the most up-to-date specifications of NEC Electronics products. Not all products and/or types are available in every country. Please check with an NEC Electronics sales representative for availability and additional information.**

• No part of this document may be copied or reproduced in any form or by any means without the prior written consent of NEC Electronics. NEC Electronics assumes no responsibility for any errors that may appear in this document.

• NEC Electronics does not assume any liability for infringement of patents, copyrights or other intellectual property rights of third parties by or arising from the use of NEC Electronics products listed in this document or any other liability arising from the use of such products. No license, express, implied or otherwise, is granted under any patents, copyrights or other intellectual property rights of NEC Electronics or others.

• Descriptions of circuits, software and other related information in this document are provided for illustrative purposes in semiconductor product operation and application examples. The incorporation of these circuits, software and information in the design of a customer's equipment shall be done under the full responsibility of the customer. NEC Electronics assumes no responsibility for any losses incurred by customers or third parties arising from the use of these circuits, software and information.

• While NEC Electronics endeavors to enhance the quality, reliability and safety of NEC Electronics products, customers agree and acknowledge that the possibility of defects thereof cannot be eliminated entirely. To minimize risks of damage to property or injury (including death) to persons arising from defects in NEC Electronics products, customers must incorporate sufficient safety measures in their design, such as redundancy, fire-containment and anti-failure features.

• NEC Electronics products are classified into the following three quality grades: "Standard", "Special" and "Specific".

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

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

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

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

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

(Note)

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

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

## CHAPTER 1 OVERVIEW

An example of using the PPG output function of 16-bit timer/event counter 00 is presented in this sample program. The PPG output duty is controlled to change the LED brightness in accordance with the number of switch inputs.

### 1.1 Main Contents of the Initial Settings

The main contents of the initial settings are as follows.

- Selecting the high-speed internal oscillator as the system clock source<sup>Note</sup>
- Stopping watchdog timer operation
- Setting  $V_{LVI}$  (low-voltage detection voltage) to 4.3 V  $\pm$ 0.2 V
- Generating an internal reset (LVI reset) signal when it is detected that  $V_{DD}$  is less than  $V_{LVI}$ , after  $V_{DD}$  (power supply voltage) becomes greater than or equal to  $V_{LVI}$
- Setting the CPU clock frequency to 8 MHz
- Setting the I/O ports
- Setting 16-bit timer/event counter 00
  - Setting CR000 and CR010 as compare registers
  - Setting one PPG output cycle to 200  $\mu$ s (0.5  $\mu$ s  $\times$  400)
  - Setting the PPG output pulse width to 20  $\mu$ s (0.5  $\mu$ s  $\times$  40)
  - Setting the count clock to  $f_{XP}/2^2$  (2 MHz)
  - Enabling timer output reversal caused upon a match between CR000 and TM00, or CR010 and TM00
  - Setting the initial timer output value to 1 (setting (1) the timer output F/F)
  - Enabling timer output (TO00 pin output)
  - Setting the operation mode to clear & start upon a match between TM00 and CR000
- Setting the valid edge of INTP1 (external interrupt) to the falling edge
- Enabling INTP1 interrupts

**Note** This is set by using the option byte.

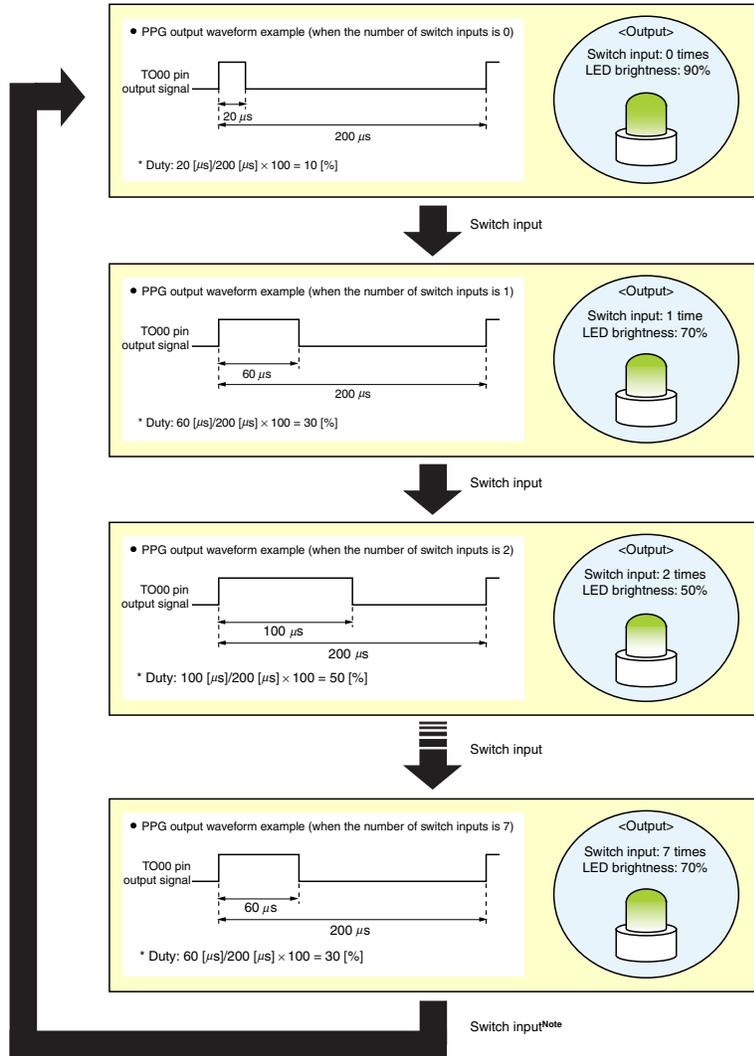
### 1.2 Contents Following the Main Loop

The LED is turned on at a fixed brightness by the PPG output of 16-bit timer/event counter 00, after completion of the initial settings.

An INTP1 interrupt is serviced when the falling edge of the INTP1 pin, which is generated by switch input, is detected. Chattering is identified when INTP1 is at high level (switch is off), after 10 ms have elapsed since a fall of the INTP1 pin was detected. The LED brightness is changed by changing the PPG output duty in accordance with the number of switch inputs when INTP1 is at low level (switch is on), after 10 ms have elapsed since an edge was detected.

Number of switch inputs <sup>Note</sup>	0 times	1 time	2 times	3 times	4 times	5 times	6 times	7 times
PPG output duty	10%	30%	50%	70%	90%	70%	50%	30%
LED brightness	90%	70%	50%	30%	10%	30%	50%	70%

In this sample program, the active level of the PPG output is set to high level, and “LED brightness = 100 – PPG output duty” results because the LED is turned on when it is at low level.



**Note** The PPG output duty from the zeroth switch input is repeated after the eighth switch input.

**Caution** For cautions when using the device, refer to the user’s manual of each product ([78K0S/KU1+](#), [78K0S/KY1+](#), [78K0S/KA1+](#), [78K0S/KB1+](#)).

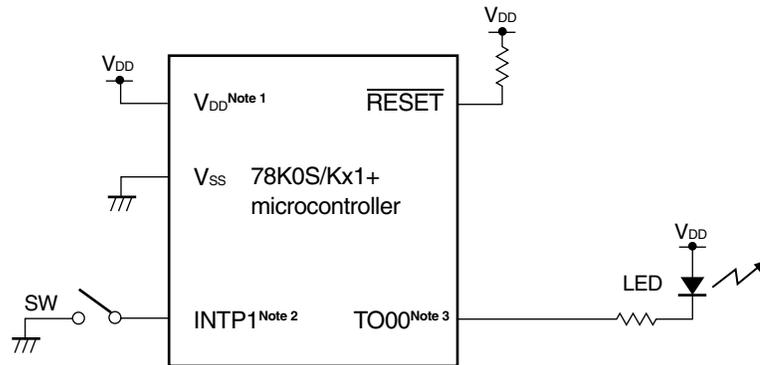
 [Column] Chattering  
Chattering is a phenomenon in which the electric signal repeats turning on and off due to a mechanical flip-flop of the contacts, immediately after the switch has been pressed.

## CHAPTER 2 CIRCUIT DIAGRAM

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

### 2.1 Circuit Diagram

A circuit diagram is shown below.



**Notes** 1. Use this in a voltage range of  $4.5\text{ V} \leq V_{DD} \leq 5.5\text{ V}$ .

2. INTP1/TxD6/P43: 78K0S/KA1+ and 78K0S/KB1+ microcontrollers  
INTP1/P32: 78K0S/KY1+ and 78K0S/KU1+ microcontrollers
3. TO00/TI010/INTP2/P31: 78K0S/KA1+ and 78K0S/KB1+ microcontrollers  
TO00/TI010/INTP0/ANI1/P21: 78K0S/KY1+ and 78K0S/KU1+ microcontrollers

- Cautions**
1. Connect the AV<sub>REF</sub> pin directly to V<sub>DD</sub> (only for the 78K0S/KA1+ and 78K0S/KB1+ microcontrollers).
  2. Connect the AV<sub>SS</sub> pin directly to GND (only for the 78K0S/KB1+ microcontroller).
  3. Leave all unused pins open (unconnected), except for the pins shown in the circuit diagram and the AV<sub>REF</sub> and AV<sub>SS</sub> pins.

### 2.2 Peripheral Hardware

The peripheral hardware to be used is shown below.

#### (1) Switch (SW)

A switch is used as an input to control the LED brightness.

#### (2) LED

An LED is used as an output corresponding to the PPG output function of 16-bit timer/event counter 00 and switch inputs.

## CHAPTER 3 SOFTWARE

This chapter describes the file configuration of the compressed file to be downloaded, internal peripheral functions of the microcontroller to be used, and initial settings and operation overview of the sample program, and shows a flow chart.

### 3.1 File Configuration

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

File Name	Description	Compressed (*.zip) File Included		
				
main.asm (Assembly language version) ----- main.c (C language version)	Source file for hardware initialization processing and main processing of microcontroller	● Note 1	● Note 1	
op.asm	Assembler source file for setting the option byte (sets the system clock source)	●	●	
tm00ppg.prw	Work space file for integrated development environment PM+		●	
tm00ppg.prj	Project file for integrated development environment PM+		●	
tm00ppg.pri tm00ppg.prs tm00ppg.prm	Project files for system simulator SM+ for 78K0S/Kx1+		● Note 2	
tm00ppg0.pnl	I/O panel file for system simulator SM+ for 78K0S/Kx1+ (used for checking peripheral hardware operations)		● Note 2	●
tm00ppg0.wvo	Timing chart file for system simulator SM+ for 78K0S/Kx1+ (used for checking waveforms)			●

**Notes 1.** “main.asm” is included with the assembly language version, and “main.c” with the C language version.

**2.** These files are not included among the files for the 78K0S/KU1+ microcontroller.

**Remark**



: Only the source file is included.



: The files to be used with integrated development environment PM+ and 78K0S/Kx1+ system simulator SM+ are included.



: The microcontroller operation simulation file to be used with system simulator SM+ for 78K0S/Kx1+ is included.

### 3.2 Internal Peripheral Functions to Be Used

The following internal peripheral functions of the microcontroller are used in this sample program.

- PPG output function: 16-bit timer/event counter 00
- $V_{DD} < V_{LVI}$  detection: Low-voltage detector (LVI)
- Switch input: INTP1<sup>Note 1</sup> (external interrupt)
- PPG output (LED output): TO00<sup>Note 2</sup> (timer output)

- Notes 1.** INTP1/TxD6/P43: 78K0S/KA1+ and 78K0S/KB1+ microcontrollers  
 INTP1/P32: 78K0S/KY1+ and 78K0S/KU1+ microcontrollers
- 2.** TO00/TI010/INTP2/P31: 78K0S/KA1+ and 78K0S/KB1+ microcontrollers  
 TO00/TI010/INTP0/ANI1/P21: 78K0S/KY1+ and 78K0S/KU1+ microcontrollers

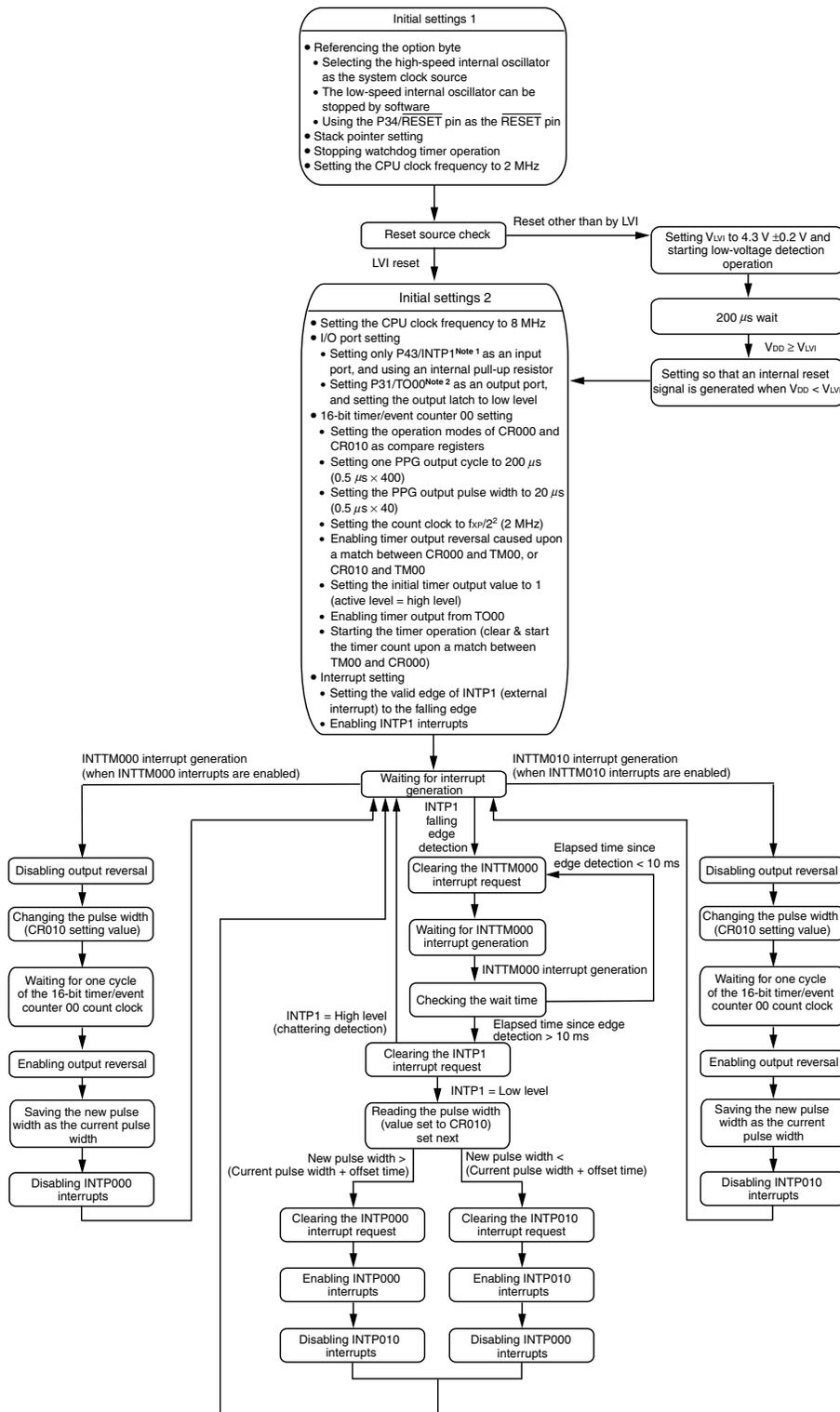
### 3.3 Initial Settings and Operation Overview

In this sample program, initial settings including the setting of the low-voltage detection function, selection of the clock frequency, setting of the I/O ports, setting of 16-bit timer/event counter 00 (PPG output function), and setting of interrupts are performed.

The LED is turned on at a fixed brightness by the PPG output of 16-bit timer/event counter 00, after completion of the initial settings.

An INTP1 interrupt is serviced when the falling edge of the INTP1 pin, which is generated by switch input, is detected. Chattering is identified when INTP1 is at high level (switch is off), after 10 ms have elapsed since a fall of the INTP1 pin was detected. The LED brightness is changed by changing the PPG output duty in accordance with the number of switch inputs when INTP1 is at low level (switch is on), after 10 ms have elapsed since an edge was detected.

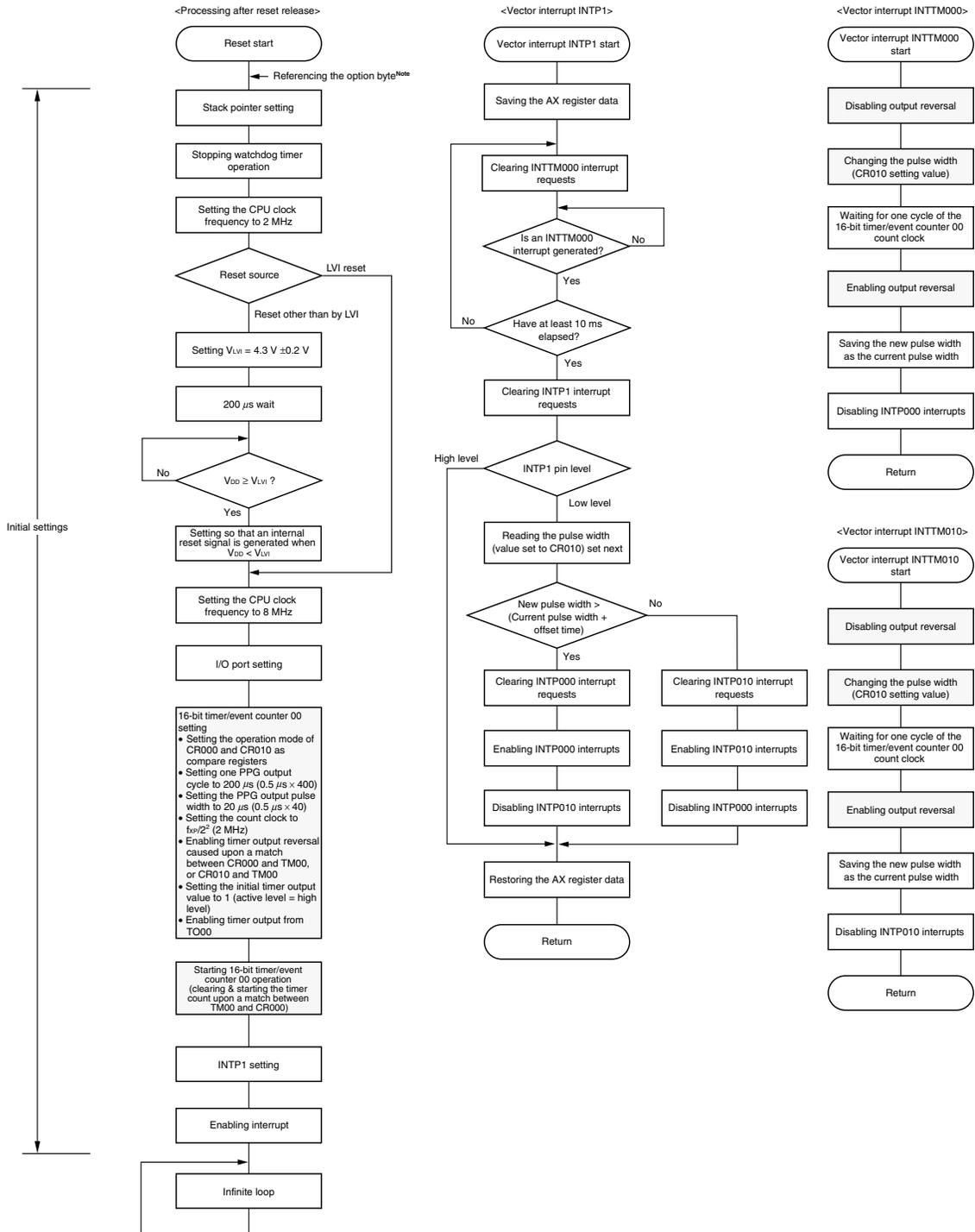
The details are described in the status transition diagram shown below.



- Notes 1.** INTP1/P43: 78K0S/KA1+ and 78K0S/KB1+ microcontrollers  
 INTP1/P32: 78K0S/KY1+ and 78K0S/KU1+ microcontrollers
- 2.** TO00/P31: 78K0S/KA1+ and 78K0S/KB1+ microcontrollers  
 TO00/P21: 78K0S/KY1+ and 78K0S/KU1+ microcontrollers

3.4 Flow Charts

The flow charts for the sample program are shown below.



**Note** Referencing the option byte is automatically performed by the microcontroller after reset release. In this sample program, the following contents are set by referencing the option byte.

- Using the high-speed internal oscillation clock (8 MHz (TYP.)) as the system clock source
- The low-speed internal oscillator can be stopped by using software
- Using the P34/RESET pin as the RESET pin

## CHAPTER 4 SETTING METHODS

This chapter describes the PPG output function of 16-bit timer/event counter 00.

For other initial settings, refer to the [78K0S/Kx1+ Sample Program \(Initial Settings\) LED Lighting Switch Control Application Note](#). For interrupt, refer to the [78K0S/Kx1+ Sample Program \(Interrupt\) External Interrupt Generated by Switch Input Application Note](#). For low-voltage detection (LVI), refer to the [78K0S/Kx1+ Sample Program \(Low-Voltage Detection\) Reset Generation During Detection at Less than 2.7 V Application Note](#).

For how to set registers, refer to the user's manual of each product ([78K0S/KU1+](#), [78K0S/KY1+](#), [78K0S/KA1+](#), [78K0S/KB1+](#)).

For assembler instructions, refer to the [78K/0S Series Instructions User's Manual](#).

### 4.1 Setting the PPG Output Function of 16-bit Timer/Event Counter 00

The following nine types of registers are used when using the PPG output function of 16-bit timer/event counter 00.

- Capture/compare control register 00 (CRC00)
- 16-bit timer capture/compare register 000 (CR000)
- 16-bit timer capture/compare register 010 (CR010)
- Prescaler mode register 00 (PRM00)
- 16-bit timer output control register 00 (TOC00)
- 16-bit timer mode control register 00 (TMC00)
- Port register x (Px)<sup>Note</sup>
- Port mode register x (PMx)<sup>Note</sup>
- Port mode control register x (PMCx)<sup>Note</sup>

**Note** Set the Px, PMx, and PMCx registers as follows, because the PPG output function uses the TO00 pin for timer output.

	Px Register	PMx Register	PMCx Register
78K0S/KA1+ and 78K0S/KB1+ microcontrollers	P31 = 0	PM31 = 0	Setting not required
78K0S/KY1+ and 78K0S/KU1+ microcontrollers	P21 = 0	PM21 = 0	PMC21 = 0

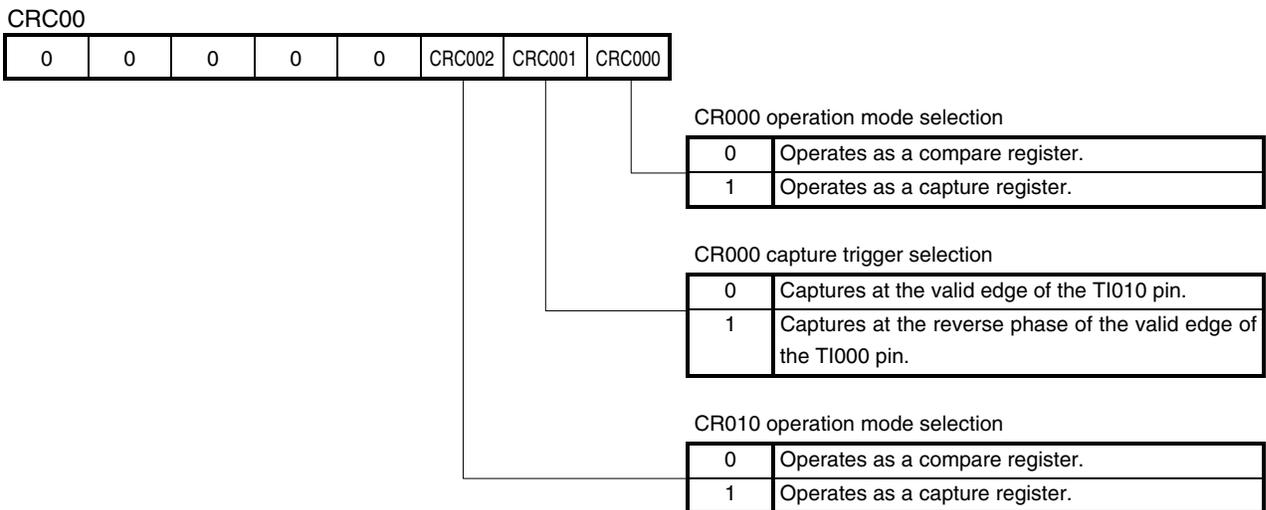
- <Example of the basic operation setting procedure when using 16-bit timer/event counter 00 as a PPG output>
- <1> Setting the CRC00 register
  - <2> Setting arbitrary values (0000H < CR010 < CR000 ≤ FFFFH) to the CR000 and CR010 registers
  - <3> Setting the count clock using the PRM00 register
  - <4> Setting the TOC00 register
  - <5> Setting the TMC00 register: starting operation

**Caution** Steps <1> to <4> may be performed randomly.

**(1) Setting the CRC00 register**

This register controls the operation of the CR000 and CR010 registers.

**Figure 4-1. Format of Capture/Compare Control Register 00 (CRC00)**



- Cautions**
1. The timer operation must be stopped before setting the CRC00 register.
  2. Do not specify the CR000 register as a capture register when the clear & start mode has been selected upon a match between TM00 and CR000 by using the TMC00 register.

**(2) Setting the CR000 register**

This register has the functions of both a capture register and a compare register.

**Figure 4-2. Format of 16-bit Timer Capture/Compare Register 000 (CR000)**



When using CR000 as a compare register

The value set to CR000 is constantly compared with the 16-bit timer counter 00 (TM00) count value, and an interrupt request (INTTM000) is generated if they match.

- Cautions**
1. Set a value other than 0000H to the CR000 register in the clear & start mode entered on a match between TM00 and CR000. When 0000H is set to the CR000 register in the free-running mode or the clear & start mode entered by the valid edge of the TI000 pin, an interrupt request (INTTM000) is generated when 0000H turns to 0001H, after an overflow (FFFFH) occurs.
  2. If the new value of the CR000 register is less than the value of 16-bit timer counter 00 (TM00), the TM00 register continues counting, overflows, and then starts counting from 0 again. If the new value of the CR000 register is less than the old value, therefore, the timer must be reset and restarted after the CR000 register value is changed.
  3. The value of the CR000 register after the TM00 counter has been stopped is not guaranteed.
  4. Capture operation may not be performed for the CR000 register set to the compare mode, even if a capture trigger is input.
  5. Changing the CR000 register setting during TM00 counter operation may cause a malfunction.

### (3) Setting the CR010 register

This register has the functions of both a capture register and a compare register.

**Figure 4-3. Format of 16-bit Timer Capture/Compare Register 010 (CR010)**



When using CR010 as a compare register

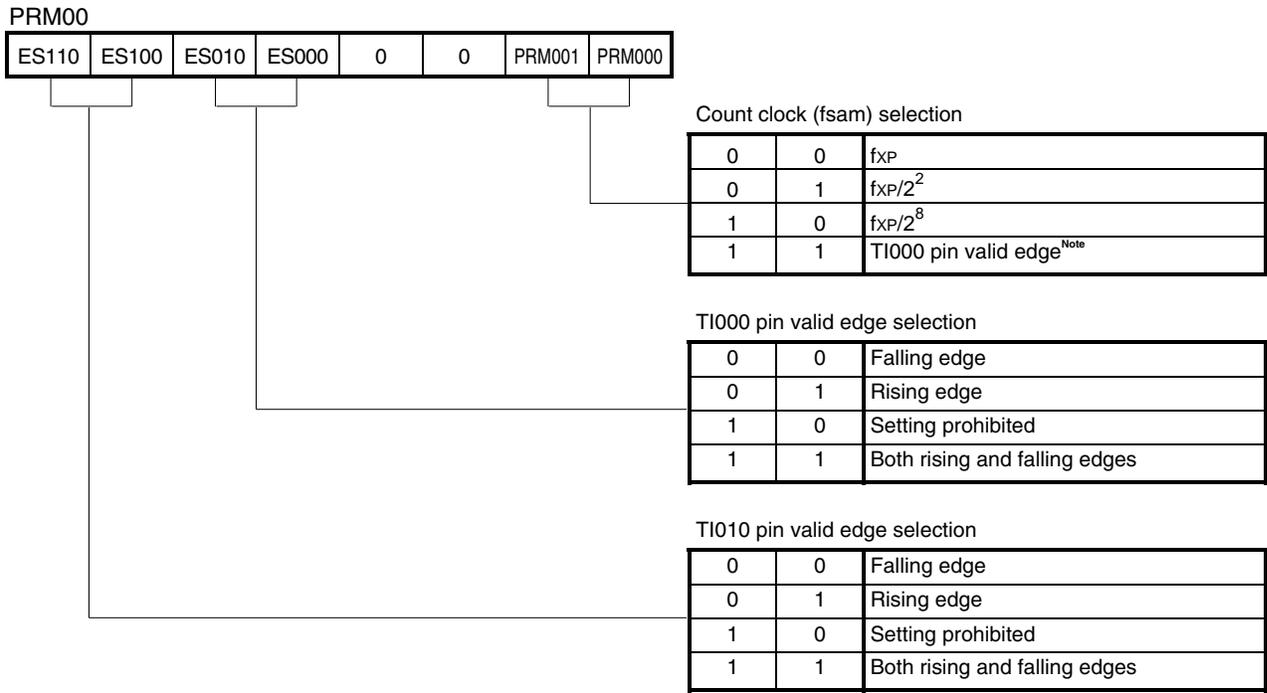
The value set to CR010 is constantly compared with the 16-bit timer counter 00 (TM00) count value, and an interrupt request (INTTM010) is generated if they match.

- Cautions**
1. When 0000H is set to CR010 in the free-running mode or the clear & start mode entered by the valid edge of the TI000 pin, an interrupt request (INTTM010) is generated when 0000H turns to 0001H, after an overflow (FFFFH) occurs.
  2. If the new value of the CR010 register is less than the TM00 counter value, the TM00 counter continues counting, overflows, and then starts counting from 0 again. If the new value of the CR010 register is less than the old value, therefore, the timer must be reset and restarted after the CR010 register value is changed.
  3. The value of the CR010 register after the TM00 counter has been stopped is not guaranteed.
  4. Capture operation may not be performed for the CR010 register set to the compare mode, even if a capture trigger is input.
  5. Changing the CR010 register setting during TM00 counter operation may cause a malfunction.

**(4) Setting the PRM00 register**

This register is used to set the count clock of the TM00 counter and the valid edges of the TI000 and TI010 pin inputs.

**Figure 4-4. Format of Prescaler Mode Register 00 (PRM00)**



**Note** The external clock requires a pulse longer than two cycles of the internal clock ( $f_{XP}$ ).

**Remark**  $f_{XP}$ : Oscillation frequency of the clock supplied to peripheral hardware

**Cautions** 1. Always set data to the PRM00 register after stopping timer operation.

2. When setting the valid edge of the TI000 pin as the count clock, do not set the clear & start mode with the valid edge of the TI000 pin and the TI000 pin as the capture trigger.
3. In the following cases, note with caution that the valid edge of the TI0n0 pin ( $n = 0, 1$ ) is detected.

- <1> A high level is input to the TI0n0 pin and the TM00 operation is enabled immediately after a system reset.
  - If the rising edge or both the rising and falling edges are specified as the valid edge of the TI0n0 pin, a rising edge is detected immediately after the TM00 operation is enabled.
- <2> The TM00 operation is stopped while the TI0n0 pin is at high level and it is then enabled after a low level is input to the TI0n0 pin.
  - If the falling edge or both the rising and falling edges are specified as the valid edge of the TI0n0 pin, a falling edge is detected immediately after the TM00 operation is enabled.
- <3> The TM00 operation is stopped while the TI0n0 pin is at low level and it is then enabled after a high level is input to the TI0n0 pin.
  - If the rising edge or both the rising and falling edges are specified as the valid edge of the TI0n0 pin, a rising edge is detected immediately after the TM00 operation is enabled.

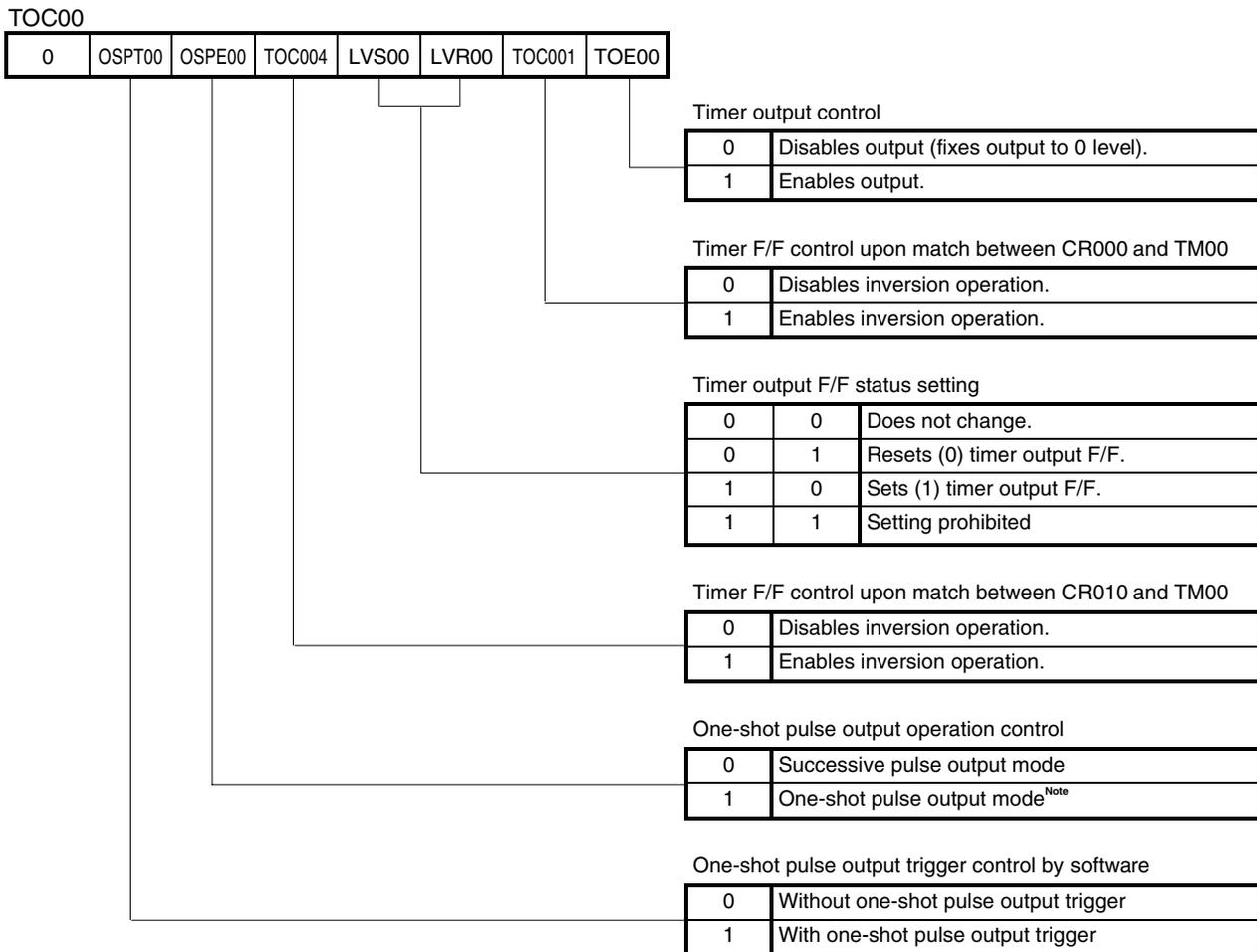
**Cautions 4.** To use the valid edge of TI000 with the count clock, it is sampled with  $f_{XP}$  to eliminate noise. The capture operation is not performed until the valid edge is sampled and the valid level is detected twice, thus eliminating noise with a short pulse width.

**5.** When the TI010/TO00/Pxx pin is used as the input pin (TI010) of the valid edge, it cannot be used as a timer output pin (TO00). When it is used as a timer output pin (TO00), it cannot be used as the input pin (TI010) of the valid edge.

**(5) Setting the TOC00 register**

This register controls the operation of the 16-bit timer/event counter 00 output controller. It is used to set/reset the timer output F/F, enable or disable output inversion, timer output (TO00 pin output), and one-shot pulse output operation, and set the one-shot pulse output trigger by software.

**Figure 4-5. Format of 16-bit Timer Output Control Register 00 (TOC00)**



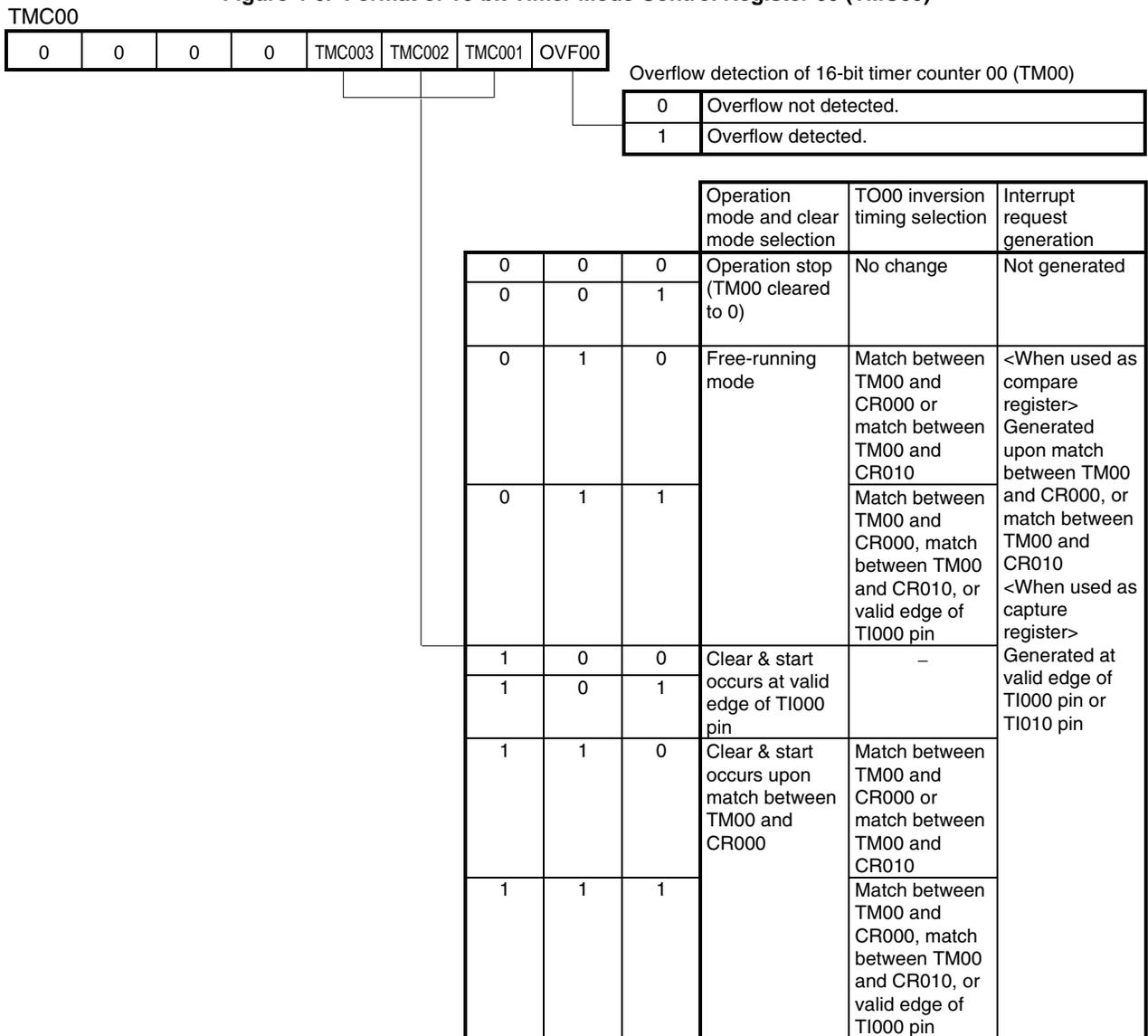
**Note** The one-shot pulse output mode operates normally only in the free-running mode and the clear & start mode set with the valid edge of the TI000 pin. In the clear & start mode set upon a match between TM00 and CR000, one-shot pulse output is not possible, because an overflow does not occur.

1. The timer operation must be stopped before setting bits other than OSPT00.
2. If LVS00 and LVR00 are read, 0 is read.
3. OSPT00 is automatically cleared after data is set, so 0 is read.
4. Do not set OSPT00 to 1 other than in one-shot pulse output mode.
5. A write interval of at least two cycles of the count clock that has been selected by using the PRM00 register is required to set (1) successively to OSPT00.
6. When TOE00 is 0, set TOE00, LVS00, and LVR00 at the same time with the 8-bit memory manipulation instruction. When TOE00 is 1, LVS00 and LVR00 can be set with the 1-bit memory manipulation instruction.
7. When the TI010/TO00/Pxx pin is used as the input pin (TI010) of the valid edge, it cannot be used as a timer output pin (TO00). When it is used as a timer output pin (TO00), it cannot be used as the input pin (TI010) of the valid edge.

**(6) Setting the TMC00 register**

This register sets the 16-bit timer/event counter 00 operation mode, TM00 counter clear mode, and output timing, and detects overflows.

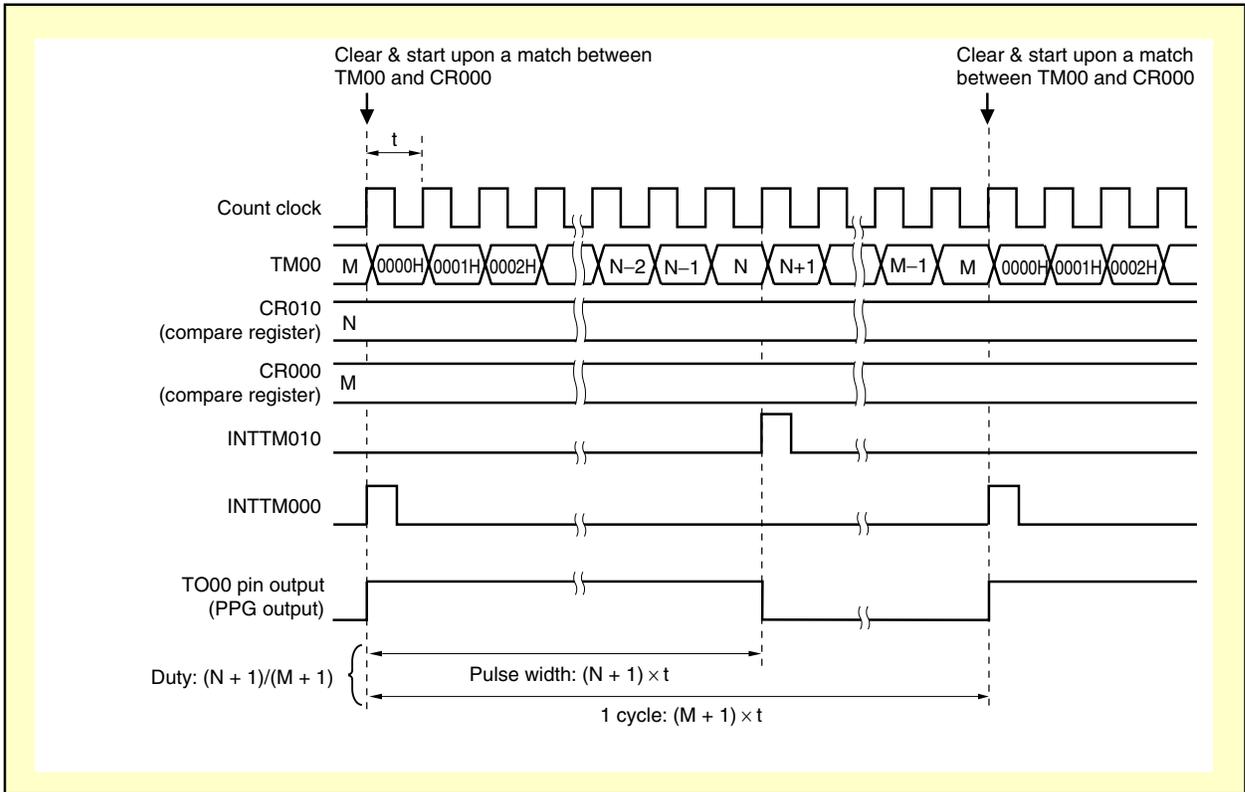
**Figure 4-6. Format of 16-bit Timer Mode Control Register 00 (TMC00)**



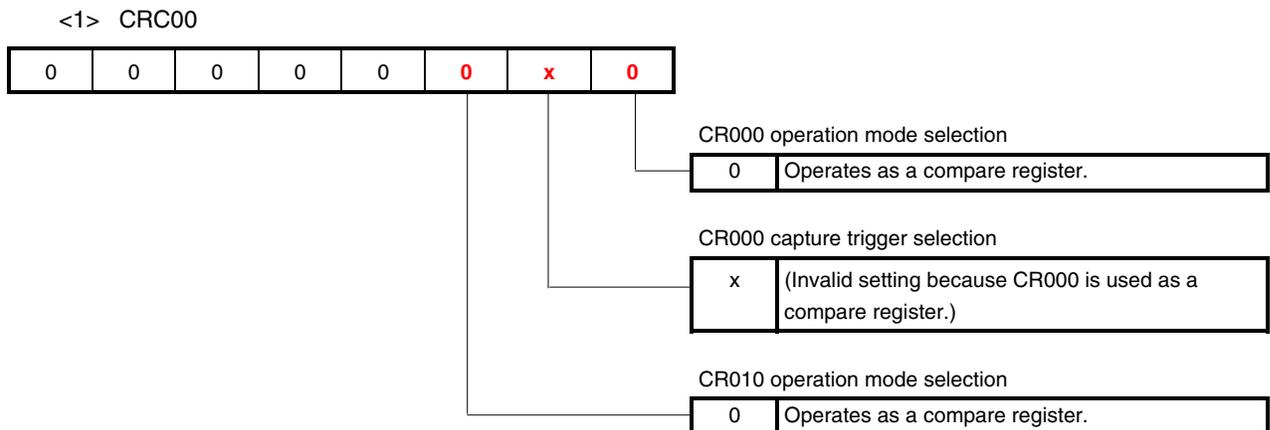
- Cautions**
1. The operation of the TM00 counter starts when values other than 0 and 0 (operation stop mode) are set to TMC002 and TMC003, respectively. To stop the operation, set TMC002 and TMC003 to 0 and 0, respectively.
  2. Write to the bits other than the OVF00 flag after stopping the timer operation.
  3. When the timer is stopped, timer counts and timer interrupts do not occur, even if a signal is input to the TI000/TI010 pin.
  4. Except when the valid edge of the TI000 pin is selected as the count clock, stop the timer operation before setting to the STOP mode or system clock stop mode; otherwise the timer may malfunction when the system clock starts.
  5. Set the valid edge of the TI000 pin with bits 4 and 5 of the PRM00 register after stopping the timer operation.
  6. If the clear & start mode is set upon a match between TM00 and CR000 or at the valid edge of the TI000 pin, or the free-running mode is selected, when the set value of the CR000 register is FFFFH and the TM00 counter value changes from FFFFH to 0000H, the OVF00 flag is set to 1.
  7. Even if the OVF00 flag is cleared before the next count clock is counted (before the TM00 counter becomes 0001H) after the TM00 counter overflows, it is re-set and clearing is disabled.
  8. Capture operation is performed at the fall of the count clock. An interrupt request (INTTM0n0: n = 0, 1), however, occurs at the rise of the next count clock.

**[Example]** When setting one PPG output cycle to 200  $\mu\text{s}$  and the pulse width to 20  $\mu\text{s}$ , and performing PPG output  
 (Count clock:  $f_{XP}/2^2$  ( $f_{XP} = 8 \text{ MHz}$ )) (Same contents as in this sample program source)

**Figure 4-7. PPG Output Timing Example**



**(1) Register settings**



<2> CR010

Setting value (N): 39

- Count clock fsam =  $8 \text{ [MHz]} / 2^2 = 2 \text{ [MHz]}$
- Pulse width  $20 \text{ [\mu s]} = (N + 1) / 2 \text{ [MHz]}$
- $N = 20 \text{ [\mu s]} \times 2 \text{ [MHz]} - 1 = 39$

<3> CR000

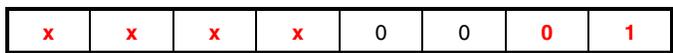
Setting value (M): 399

- Count clock fsam =  $8 \text{ [MHz]} / 2^2 = 2 \text{ [MHz]}$
- 1 cycle  $200 \text{ [\mu s]} = (M + 1) / 2 \text{ [MHz]}$
- $M = 200 \text{ [\mu s]} \times 2 \text{ [MHz]} - 1 = 399$

**Caution** Set values within the following range to CR000 and CR010.

$$0000\text{H} < \text{CR010} < \text{CR000} \leq \text{FFFFH}$$

<4> PRM00



Count clock (fsam) selection

0	1	$f_{XP} / 2^2$
---	---	----------------

TI000 pin valid edge selection

x	x	(Invalid setting because the TI000 pin is not used. Setting "1, 0", however, is prohibited.)
---	---	--

TI010 pin valid edge selection

x	x	(Invalid setting because the TI010 pin is not used. Setting "1, 0", however, is prohibited.)
---	---	--

<5> TOC00



Timer output control

1	Enables output.
---	-----------------

Timer F/F control upon match between CR000 and TM00

1	Enables inversion operation.
---	------------------------------

Timer output F/F status setting

1	0	Sets (1) timer output F/F.
---	---	----------------------------

Timer F/F control upon match between CR010 and TM00

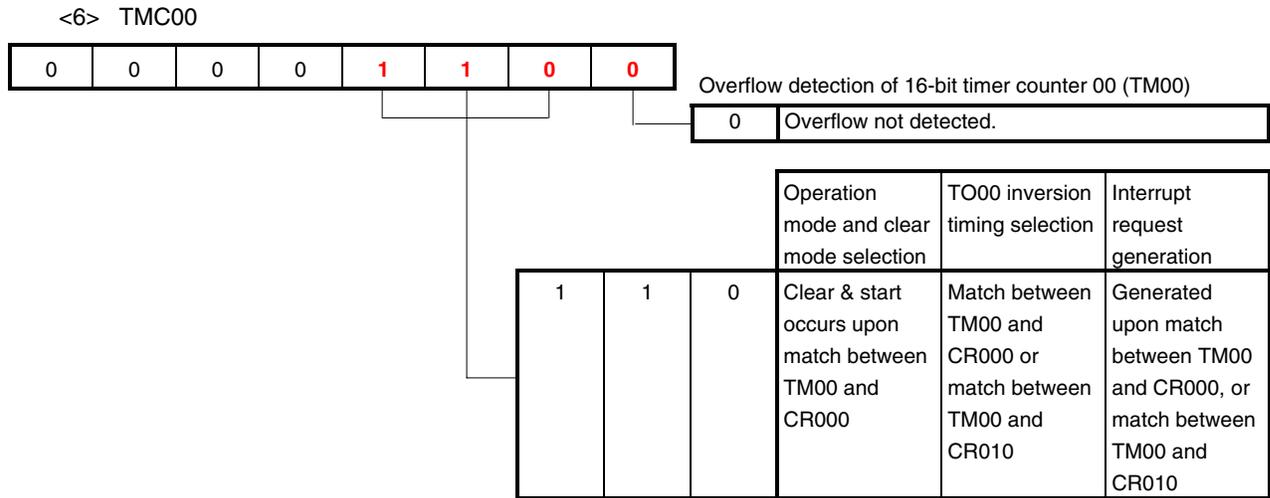
1	Enables inversion operation.
---	------------------------------

One-shot pulse output operation control

0	Successive pulse output mode
---	------------------------------

One-shot pulse output trigger control by software

0	Without one-shot pulse output trigger
---	---------------------------------------



<7> Px, PMx, PMCx

	Px Register	PMx Register	PMCx Register
78K0S/KA1+ and 78K0S/KB1+ microcontrollers	P31 = 0	PM31 = 0	Setting not required
78K0S/KY1+ and 78K0S/KU1+ microcontrollers	P21 = 0	PM21 = 0	PMC21 = 0

**(2) Sample program**

In the example below, "x" in **(1) Register settings** is set to "0".

<1> Assembly language (when using the 78K0S/KA1+ and 78K0S/KB1+ microcontrollers)

```

CLR1   P3.1
CLR1   PM3.1
MOV    CRC00, #00000000B
MOVW   CR000, #399
MOVW   CR010, #39
MOV    PRM00, #00000001B
MOV    TOC00, #00011011B
MOV    TMC00, #00001100B
    
```

<2> C language (when using the 78K0S/KA1+ and 78K0S/KB1+ microcontrollers)

```

P3.1 = 0;
PM3.1 = 0;
CRC00 = 0b00000000;
CR000 = 399;
CR010 = 39;
PRM00 = 0b00000001;
TOC00 = 0b00011011;
TMC00 = 0b00001100;
    
```





## 4.2 Setting and Changing the PPG Output Duty

### 4.2.1 Setting the PPG output duty

The PPG output function of 16-bit timer/event counter 00 is used to output a pulse from the TO00 pin in this sample program.

- Pulse width =  $(N + 1)/fsam$
- 1 cycle =  $(M + 1)/fsam$
- PPG output duty =  $(N + 1)/(M + 1) \times 100$

**Remark** N: CR010 register setting value  
M: CR000 register setting value  
fsam: Count clock frequency of 16-bit timer/event counter 00  
 $0000H < N < M \leq FFFFH$

Calculation example: When the CR010 register setting value is 39 and the CR000 register setting value is 399 (during fsam = 2 MHz operation)

- Pulse width =  $(N + 1)/fsam = (39 + 1)/2 \text{ [MHz]} = 20 \text{ [}\mu\text{s]}$
- 1 cycle =  $(M + 1)/fsam = (399 + 1)/2 \text{ [MHz]} = 200 \text{ [}\mu\text{s]}$
- PPG output duty =  $(N + 1)/(M + 1) \times 100 = (39 + 1)/(399 + 1) \times 100 = 10 \text{ [%]}$

Furthermore, the CR010 register setting value and PPG output duty are changed according to the number of switch inputs.

No. of Switch Inputs <sup>Note</sup>	CR010 Register Setting Value	CR000 Register Setting Value	PPG Output Duty
0	39	399	10% $((39 + 1)/(399 + 1) \times 100)$
1	119		30% $((119 + 1)/(399 + 1) \times 100)$
2	199		50% $((199 + 1)/(399 + 1) \times 100)$
3	279		70% $((279 + 1)/(399 + 1) \times 100)$
4	359		90% $((359 + 1)/(399 + 1) \times 100)$
5	279		70% $((279 + 1)/(399 + 1) \times 100)$
6	199		50% $((199 + 1)/(399 + 1) \times 100)$
7	119		30% $((119 + 1)/(399 + 1) \times 100)$

**Note** The PPG output duty from the zeroth switch input is repeated after the eighth switch input.

#### 4.2.2 Changing the CR010 setting value (PPG output duty)

To increase and decrease the PPG output duty in this sample program, basically, the CR010 register value is changed during an INTTM000 interrupt and during an INTTM010 interrupt, respectively. The following is performed, however, in consideration of the offset time “from INTTM010 interrupt generation, via changing the CR010 register value, to enabling timer output reversal”.

- When “new pulse width > current pulse width + offset time”: Changed during INTTM000 interrupt servicing
- When “new pulse width < current pulse width + offset time”: Changed during INTTM010 interrupt servicing

If the CR010 register value is not changed by using the above-mentioned method, a match between CR010 and TM00 may not occur within the period of one PPG output cycle, or multiple matches between CR010 and TM00 may occur and the TO00 pin output level may be reversed for every match.

Use the following procedure to change the CR010 register value during INTTM0x0 (x = 0, 1) interrupt servicing.

- <1> Disable timer output reversal operation upon a match between TM00 and CR010 (TOC004 = 0).
- <2> Rewrite CR010.
- <3> Wait for one cycle of the TM00 count clock.
- <4> Enable timer output reversal operation upon a match between TM00 and CR010 (TOC004 = 1).

Next, a timing chart example is shown for when the CR010 register value is changed.

**Figure 4-8. Timing Chart Example When “New Pulse Width > Current Pulse Width + Offset Time”**

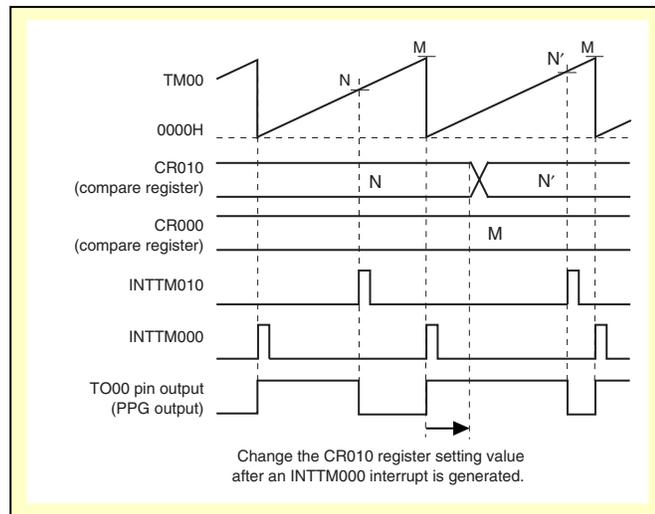
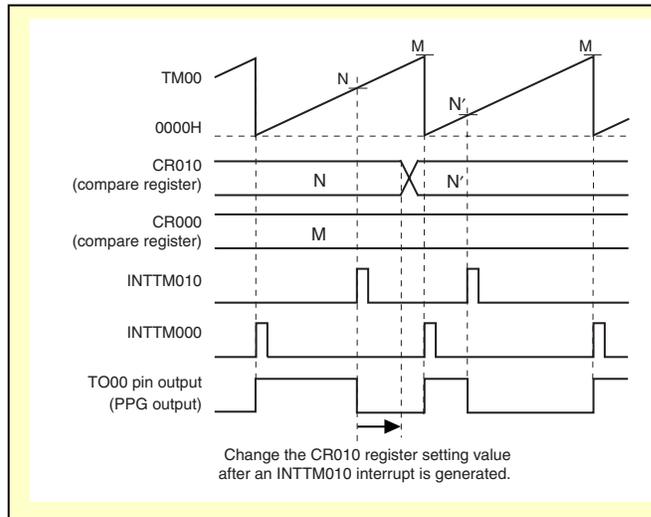


Figure 4-9. Timing Chart Example When “New Pulse Width &lt; Current Pulse Width + Offset Time”



For details of how to change compare registers during timer operation and cautions, refer to **6.5 Cautions Related to 16-bit Timer/Event Counter 00** in the user's manual of each product ([78K0S/KU1+](#), [78K0S/KY1+](#), [78K0S/KA1+](#), [78K0S/KB1+](#)).



## (2) C language

```

#define offsetT 56/4          /* For the correction of servicing time corresponding
to that of INTTM010 */
    .
    .
    .
__interrupt void fn_intp1(){
    .
    .
    .
    if (g_unNextD > g_unCurrentD + offsetT){/* Use an INTTM000 interrupt to
increase the pulse width */
        TMIF000 = 0;          /* Clear invalid interrupt requests */
        TMMK000 = 0;          /* Unmask INTTM000 interrupts */
        TMMK010 = 1;          /* Mask INTTM010 interrupts */
    }
    else {                    /* Use an INTTM010 interrupt to decrease the pulse
width */
        TMIF010 = 0;          /* Clear invalid interrupt requests */
        TMMK010 = 0;          /* Unmask INTTM010 interrupts */
        TMMK000 = 1;          /* Mask INTTM000 interrupts */
    }
    .
    .
    .
__interrupt void fn_inttm000(){
    TOC00.4 = 0;              /* Disable output reversal */
    CR010 = g_unNextD;        /* Change the pulse width (duty) */
    NOP();                    /* Wait for one timer 00 count clock cycle */
    NOP();
    TOC00.4 = 1;              /* Enable output reversal */
    g_unCurrentD = g_unNextD; /* Save the new pulse width as the current pulse
width */
    TMMK000 = 1;              /* Mask INTTM000 interrupts */

    return;
}

__interrupt void fn_inttm010(){
    TOC00.4 = 0;              /* Disable output reversal */
    CR010 = g_unNextD;        /* Change the pulse width (duty) */
    NOP();                    /* Wait for one timer 00 count clock cycle */
    NOP();
    TOC00.4 = 1;              /* Enable output reversal */
    g_unCurrentD = g_unNextD; /* Save the new pulse width as the current pulse
width */
    TMMK010 = 1;              /* Mask INTTM010 interrupts */

    return;
}

```

Comparing the total of the current pulse value and offset time, and the new pulse width

Using an INTTM000 interrupt, if "new pulse width > current pulse value + offset time"

Using an INTTM010 interrupt, if "new pulse width < current pulse value + offset time"

Changing the duty (CR010) at the INTTM000 interrupt generation timing<sup>Note</sup>

Changing the duty (CR010) at the INTTM010 interrupt generation timing<sup>Note</sup>

**Note** In this sample program, the interrupt servicing to be used is not required to be disable or enabled when changing the setting value, because the CR010 register setting value is changed during interrupt servicing.

### 4.3 Setting the Chattering Detection Time

The generation of 16-bit timer/event counter 00 interrupts (INTTM000) is counted to remove chattering of 10 ms or less, in order to handle chattering during switch input (INTP1 interrupt generation) in this sample program.

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

**Remark** T: INTTM000 interrupt cycle

T': Time from the start of INTP1 edge detection until the first INTTM000 is generated after INTP1 edge detection ( $0 < T' \leq T$ )

M: Number of INTTM000 interrupts after INTP1 edge detection

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

$$T_c = T' + 10 \text{ ms}$$

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

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

↓

Chattering detection time ( $T_c$ ) > 10 ms

Calculation example: When the INTTM000 interrupt cycle (T) is 200  $\mu\text{s}$  (refer to [4.2 Setting and Changing the PPG Output Duty](#)), and the number of INTTM000 interrupts after INTP1 edge detection (M) is 51

$$\begin{aligned} T_c &= T' + T \times (M - 1) \\ &= T' + 200 [\mu\text{s}] \times (51 - 1) \\ &= T' + 10000 [\mu\text{s}] \\ &= T' + 10 [\text{ms}] \end{aligned}$$

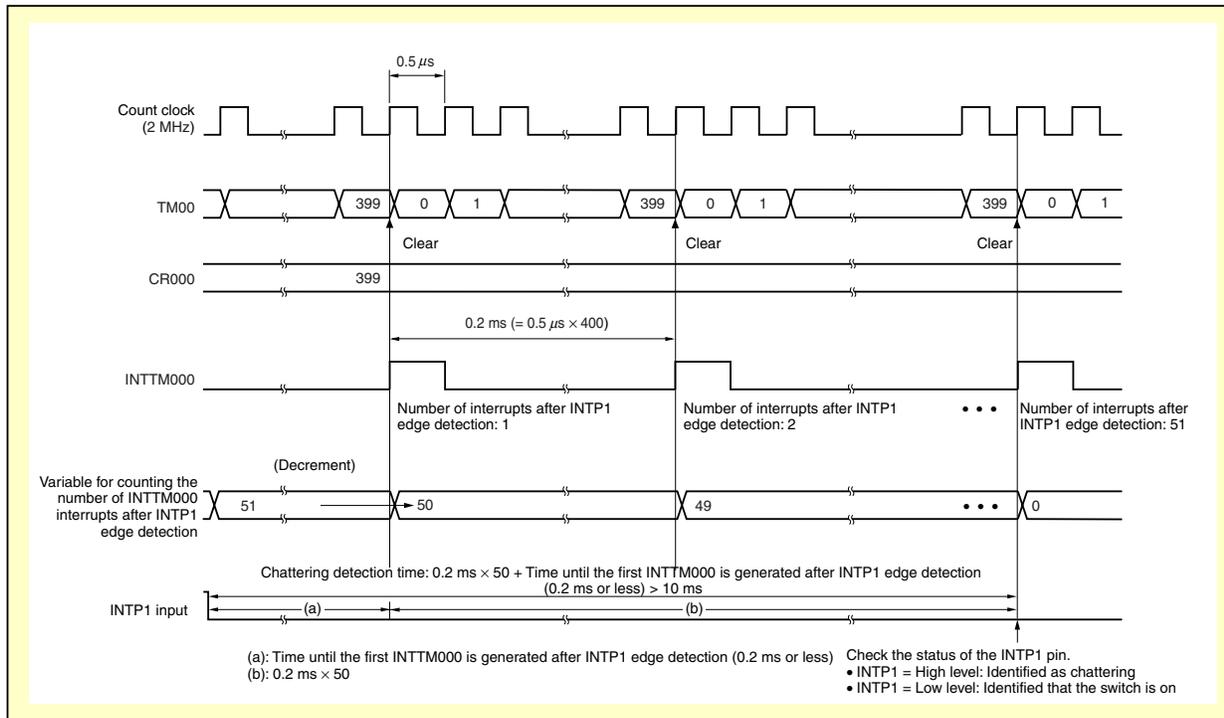
$0 < T' \leq 200 \mu\text{s}$ , therefore,

$$10 \text{ ms} < T_c \leq 10.2 [\text{ms}]$$

↓

Chattering detection time ( $T_c$ ) > 10 ms

Figure 4-10. Timing Chart Example of Chattering Detection



## CHAPTER 5 OPERATION CHECK USING SYSTEM SIMULATOR SM+

This chapter describes how the sample program operates with system simulator SM+ for 78K0S/Kx1+, by using the assembly language file (source files + project file) that has been downloaded by selecting the  icon.

<R> **Caution** System simulator SM+ for 78K0S/Kx1+ is not supported with the 78K0S/KU1+ microcontroller (as of July 2008). The operation of the 78K0S/KU1+ microcontroller, therefore, cannot be checked by using system simulator SM+ for 78K0S/Kx1+.

### <R> 5.1 Building the Sample Program

To check the operation of the sample program by using system simulator SM+ for 78K0S/Kx1+ (hereinafter referred to as “SM+”), SM+ must be started after building the sample program. This section describes how to build a sample program by using the assembly language sample program (source program + project file) downloaded by clicking the  icon. See the [78K0S/Kx1+ Sample Program Startup Guide Application Note](#) for how to build other downloaded programs.

For the details of how to operate PM+, refer to the [PM+ Project Manager User's Manual](#).



#### [Column] Build errors

Change the compiler option setting according to the following procedure when the error message “A006 File not found ‘C:\NECTOOLS32\LIB78K0S\s0sl.rel’” or “\*\*\* ERROR F206 Segment ‘@@DATA’ can’t allocate to memory - ignored.” is displayed, when building with PM+.

<1> Select [Compiler Options] from the [Tool] menu.

<2> The [Compiler Options] dialog box will be displayed. Select the [Startup Routine] tab.

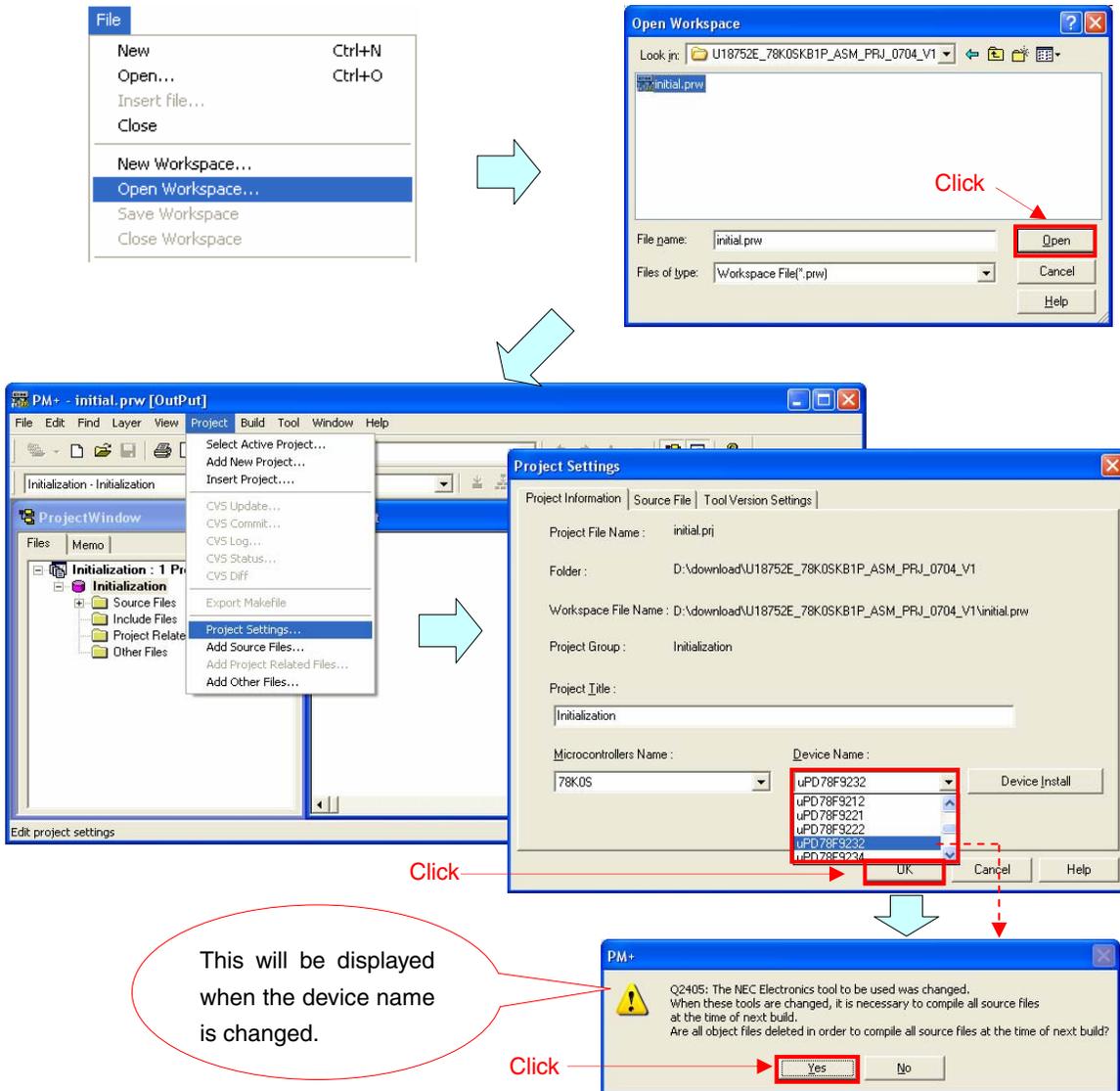
<3> Uncheck the [Using Fixed Area of Standard Library] check box. (Leave the other check boxes as they are.)

A RAM area of 118 bytes that has been secured as a fixed standard library area will be enabled for use when the [Using Fixed Area of Standard Library] check box is unchecked; however, the standard libraries (such as the getchar function and malloc function) will be disabled for use.

The [Using Fixed Area of Standard Library] check box is unchecked by default when the file that has been downloaded by clicking the  icon is used in this sample program.

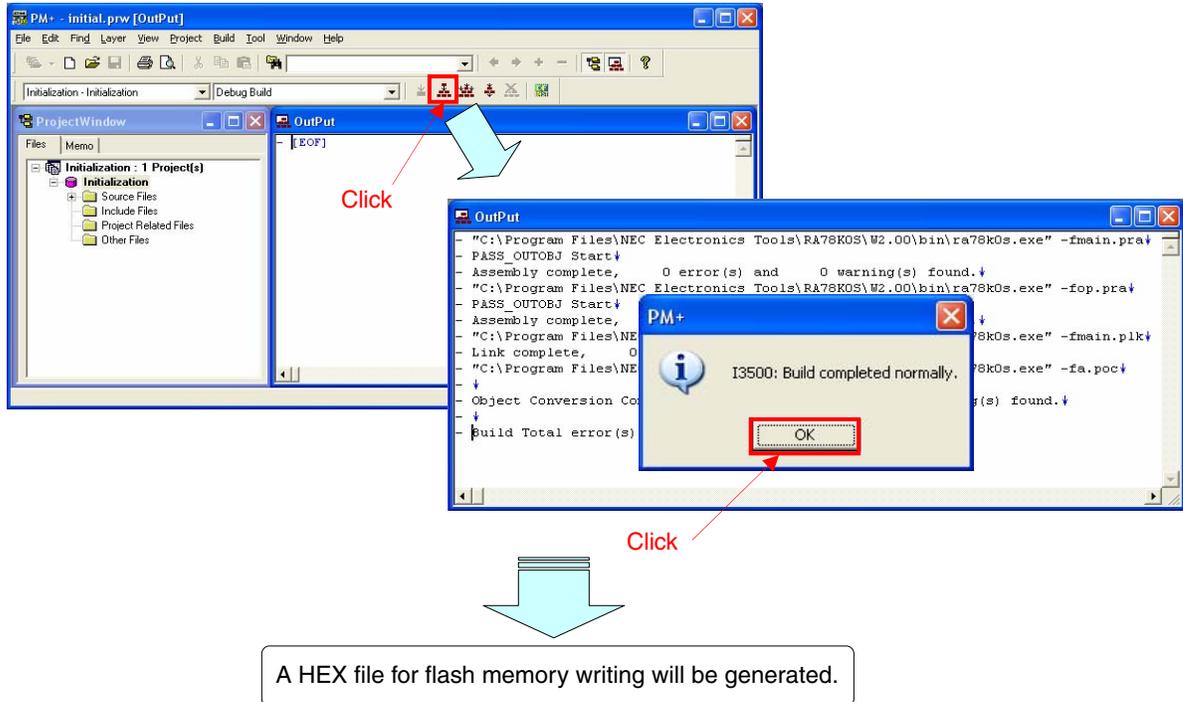
- (1) Start PM+.
- (2) Select "tm00ppg.prw" by clicking [Open Workspace] from the [File] menu and click [Open]. A workspace into which the source file will be automatically read will be created.
- (3) Select [Project Settings] from the [Project] menu. When the [Project Settings] window opens, select the name of the device to be used (the device with the largest ROM or RAM size will be selected by default), and click [OK].

**Remark** Screenshots of the Sample Program (Initial Settings) LED Lighting Switch Control are shown below.



- (4) Click  ([Build] button). When the source files are built normally, the message "I3500: Build completed normally." will be displayed.
- (5) Click the [OK] button in the message dialog box. A HEX file for flash memory writing will be created.

**Remark** Screenshots of the Sample Program (Initial Settings) LED Lighting Switch Control are shown below.

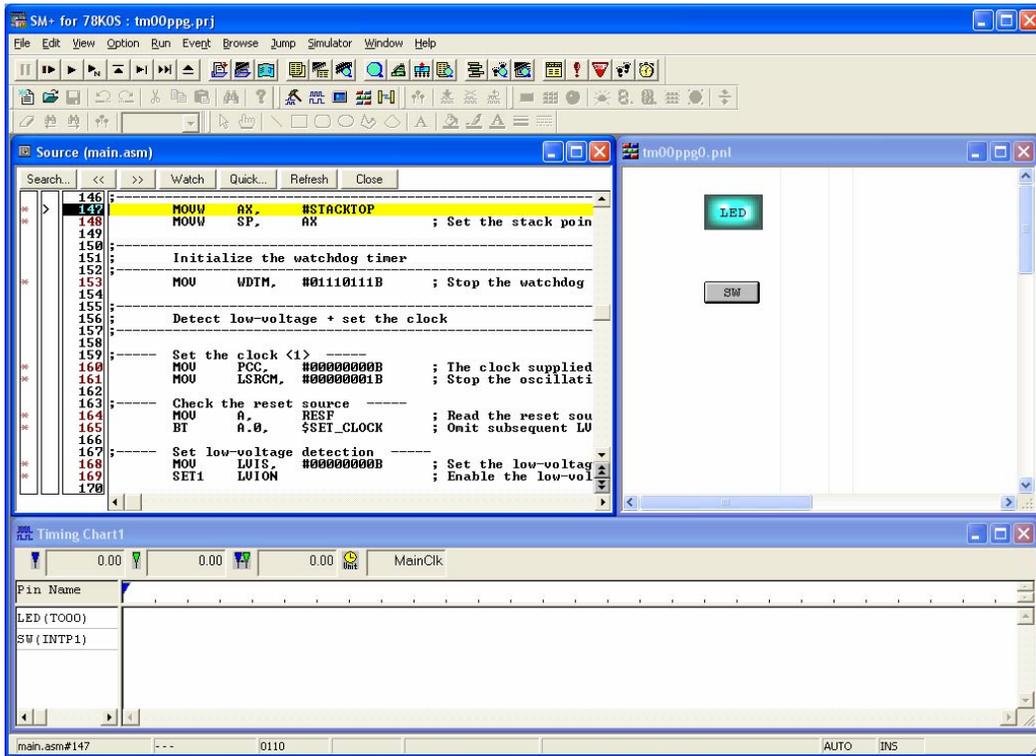


## 5.2 Operation with SM+

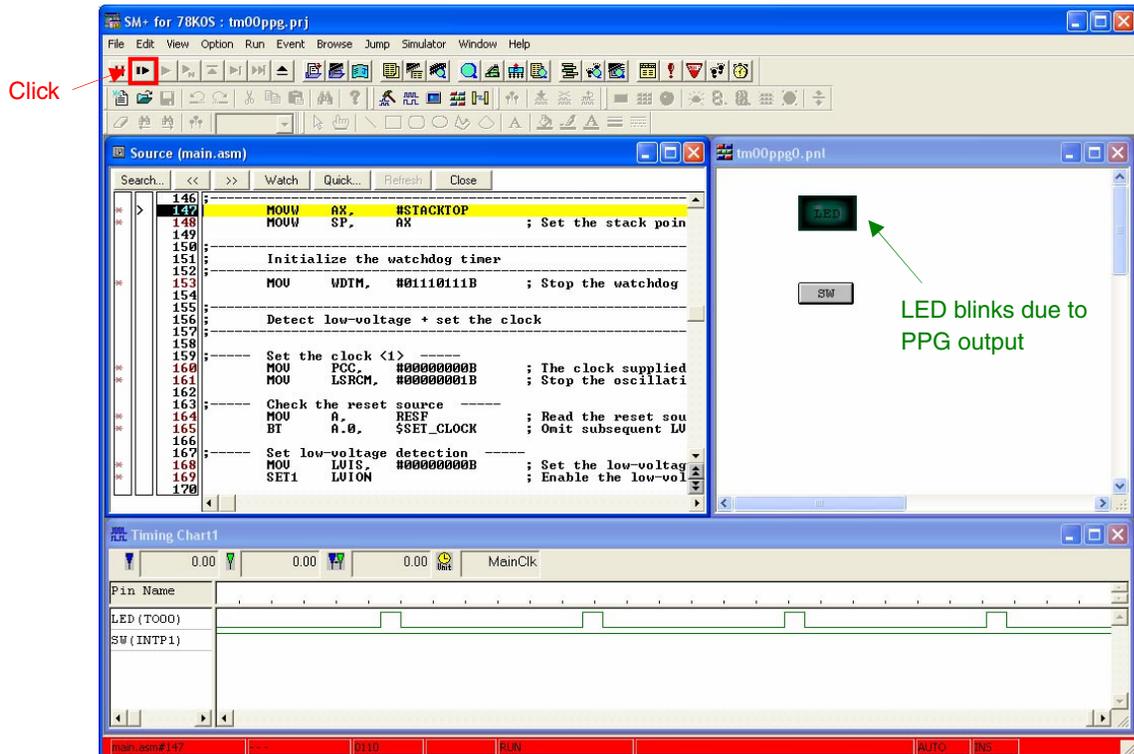
This section describes examples of checking the operation on the I/O panel window or timing chart window of SM+. For the details of how to operate SM+, refer to the [SM+ System Simulator Operation User's Manual](#).

- <R>
- (1) When SM+ for 78K0S/Kx1+ W1.02 ("SM+" hereafter) is used in the environment of PM+ Ver. 6.30, SM+ cannot be selected as the debugger. In this case, start SM+ via method (a) or (b) described below, while keeping PM+ running after completing building a project.
    - (a) When starting SM+ in PM+
      - <1> Select [Register Ex-tool] from the [Tool] menu and register "SM+ for 78K0S/Kx1+".
      - <2> Select [Ex-tool Bar] from the [View] menu and add the SM+ icon to the PM+ toolbar.
      - <3> Click the SM+ icon and start SM+.
 (See the PM+ help for details on how to register external tools.)
    - (b) When not starting SM+ in PM+
      - Start SM+ from the Windows start menu.

- (2) The following screen will be displayed when SM+ is started. (This is a sample screenshot of when an assembly language source file downloaded by clicking the  icon was used.)

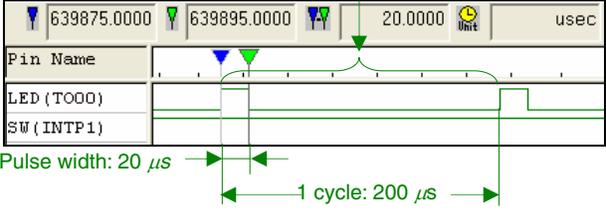
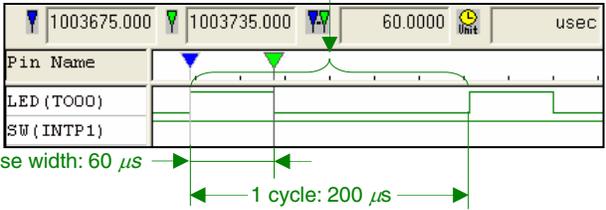
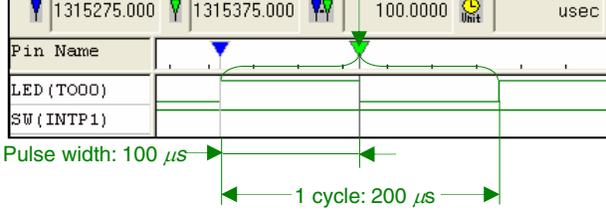
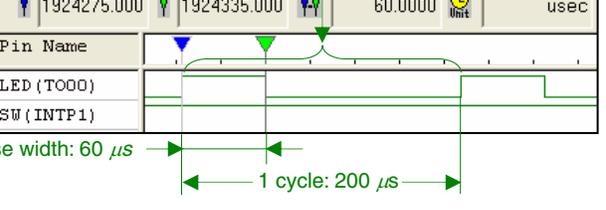


- (3) Click  ([Restart] button). The program will be executed after the CPU is reset and the following screen will be displayed.



This turns red during program execution.

- (4) Click the [SW] button in the I/O panel window, during program execution.  
 Change the PPG output duty according to the number of [SW] button inputs.  
 Check that the brightness of [LED] in the I/O panel window and the waveforms in the timing chart window change, depending on the number of [SW] button inputs.

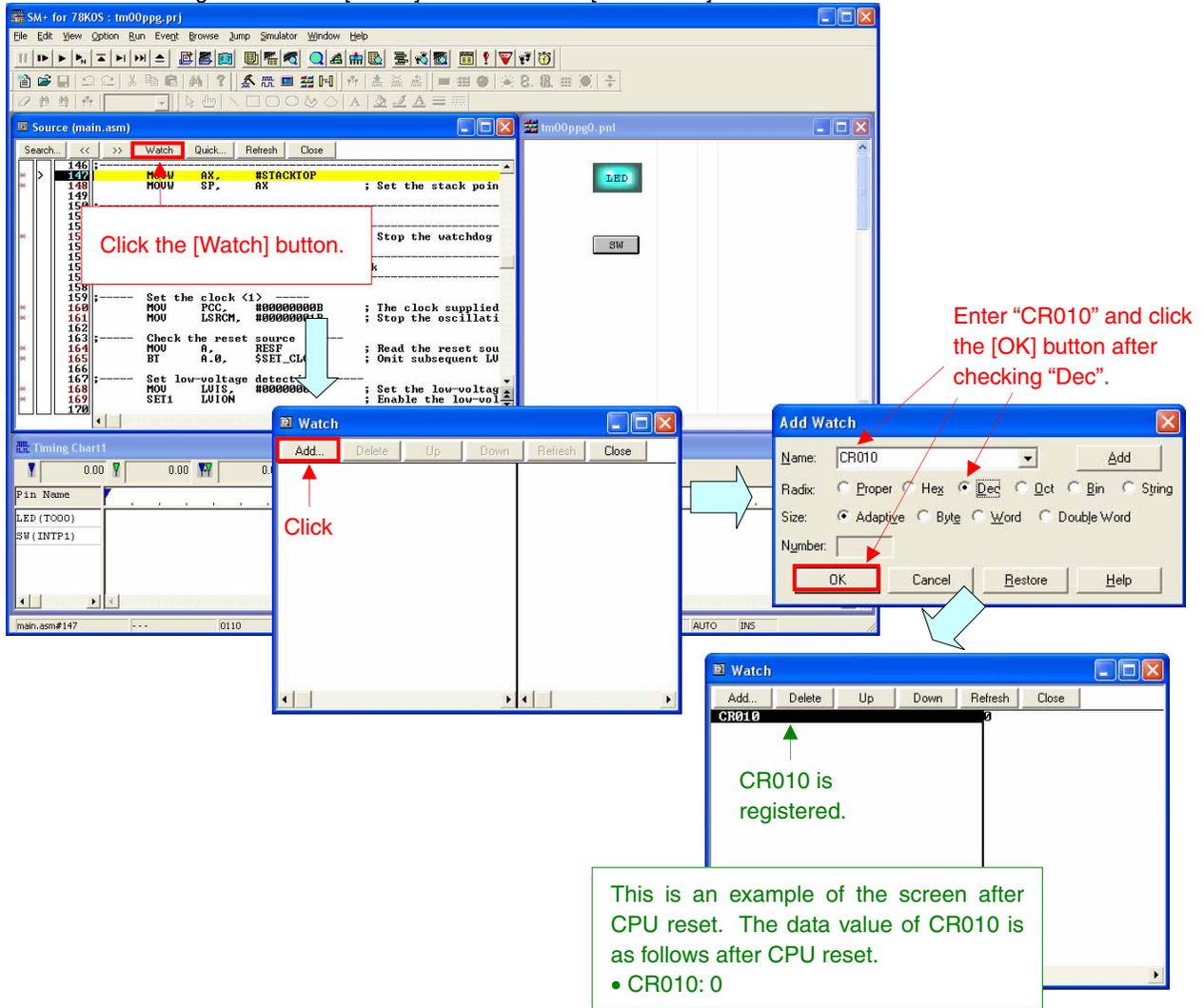
I/O panel window	Timing chart window
<p>Brightness: 90% </p> <p>Do not click. </p>	<p>PPG output at duty 10%</p> 
<p>Brightness: 70% </p> <p>Click once. </p>	<p>PPG output at duty 30%</p> 
<p>Brightness: 50% </p> <p>Click twice. </p>	<p>PPG output at duty 50%</p> 
<p>•</p> <p>•</p> <p>•</p>	<p>•</p> <p>•</p> <p>•</p>
<p>Brightness: 70% </p> <p>Click seven times. </p>	<p>PPG output at duty 30%</p> 

Note

**Note** The PPG output duty from the zeroth switch input is repeated after the eighth switch input.

[Supplement 1] The changes in the data value of the CR010 register can be checked by using the SM+ watch function.

- <1> Click the [Watch] button in the source window to open the [Watch] window.
- <2> Click [Add] to open the [Add Watch] window. (At this time, the [Watch] window is kept opened.)
- <3> Enter “CR010” in the [Name] field and click the [OK] button after checking “Dec” under Radix. “CR010” will be registered in the [Watch] window and the [Add Watch] window will be closed.



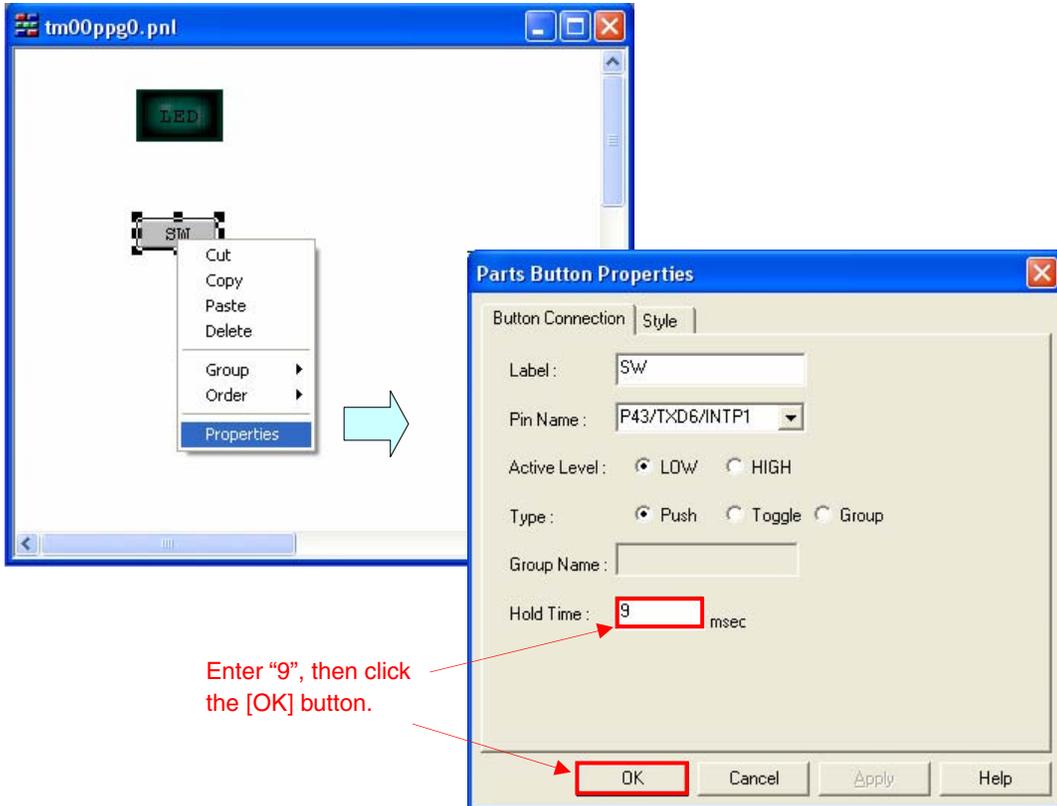
- <4> Execute the program and click the [SW] button in the I/O panel window. Check that the data value of CR010 in the [Watch] window changes, depending on the number of [SW] button inputs.

Number of [SW] Button Inputs <sup>Note</sup>	Data Value in [Watch] Window
0	CR010: 39
1	CR010: 119
2	CR010: 199
3	CR010: 279
4	CR010: 359
5	CR010: 279
6	CR010: 199
7	CR010: 119

**Note** The PPG output duty from the zeroth switch input is repeated after the eighth switch input.

[Supplement 2] The [SW] button hold time can be set to less than 10 ms to check whether chattering is being detected.

- <1> Select  on the toolbar.
- <2> Right-click the [SW] button in the I/O panel window and select [Properties].
- <3> Enter "9" for the Hold Time and click the [OK] button.



- <4> Select  on the toolbar.
- <5> Execute the program and click the [SW] button. Even if the [SW] button is clicked, chattering will be identified and the PPG output duty will not change, because the button hold time is 9 ms.

## CHAPTER 6 RELATED DOCUMENTS

Document Name		Japanese/English
78K0S/KU1+ User's Manual		<a href="#">PDF</a>
78K0S/KY1+ User's Manual		<a href="#">PDF</a>
78K0S/KA1+ User's Manual		<a href="#">PDF</a>
78K0S/KB1+ User's Manual		<a href="#">PDF</a>
78K/0S Series Instructions User's Manual		<a href="#">PDF</a>
RA78K0S Assembler Package User's Manual	Language	<a href="#">PDF</a>
	Operation	<a href="#">PDF</a>
CC78K0S C Compiler User's Manual	Language	<a href="#">PDF</a>
	Operation	<a href="#">PDF</a>
PM+ Project Manager User's Manual		<a href="#">PDF</a>
SM+ System Simulator Operation User's Manual		<a href="#">PDF</a>
Flash Programming Manual (Basic) MINICUBE2 version	78K0S/KU1+	<a href="#">PDF</a>
	78K0S/KY1+	<a href="#">PDF</a>
	78K0S/KA1+	<a href="#">PDF</a>
	78K0S/KB1+	<a href="#">PDF</a>
78K0S/Kx1+ Application Note	Sample Program Startup Guide	<a href="#">PDF</a>
	Sample Program (Initial Settings) LED Lighting Switch Control	<a href="#">PDF</a>
	Sample Program (Interrupt) External Interrupt Generated by Switch Input	<a href="#">PDF</a>
	Sample Program (Low-Voltage Detection) Reset Generation During Detection at Less than 2.7 V	<a href="#">PDF</a>
	Sample Program (16-bit Timer/Event Counter 00) Interval Timer	<a href="#">PDF</a>
	Sample Program (16-bit Timer/Event Counter 00) External Event Counter	<a href="#">PDF</a>
	Sample Program (16-bit Timer/Event Counter 00) Pulse Width Measurement	<a href="#">PDF</a>
	Sample Program (16-bit Timer/Event Counter 00) One-Shot Pulse Output	<a href="#">PDF</a>

<R>

## APPENDIX A PROGRAM LIST

As a program list example, the 78K0S/KB1+ microcontroller source program is shown below.

### ● main.asm (Assembly language version)

```
;*****  
;  
;   NEC Electronics      78K0S/KB1+  
;  
;*****  
;   78K0S/KB1+  Sample program  
;*****  
;   16-bit timer 00 (PPG output)  
;*****  
;<<History>>  
;   2007.7.--  Release  
;*****  
;  
;<<Overview>>  
;  
;This sample program presents an example of using the PPG output function  
;of 16-bit timer 00. Perform PPG output at a 200 us cycle and change the  
;duty width by 20% for every switch input.  
;  
;  
; <Principal setting contents>  
;  
; - Stop the watchdog timer operation  
; - Set the low-voltage detection voltage (VLVI) to 4.3 V +/-0.2 V  
; - Generate an internal reset signal (low-voltage detector) when VDD <  
VLVI after VDD >= VLVI  
; - Set the CPU clock to 8 MHz  
; - Set the clock supplied to the peripheral hardware to 8 MHz  
; - Set the valid edge of external interrupt INTP1 to falling edge  
; - Set the chattering detection time during switch input to 10 ms  
; - Use the HL register for interrupt servicing (similarly as a global  
variable)  
;  
;  
; <16-bit timer 00 settings>  
; - Operation mode: Clear & start the timer count upon a match between TM00  
and CR000  
; - Count clock = fxp/4 (2 MHz)  
; - Use CR000 and CR010 as compare registers  
; - Initialize the cycle of CR000 to 200 us  
; - Initialize the pulse width of CR010 to 20 us
```

```

; - Reverse the output upon a match between CR000 and CR010 (PPG output)
; - Set the initial output value (= active level) to High
; - Set the output latch of P31 to Low (to use the T000 output)
; - Set P31 to the output mode (to use the T000 output)
;
;
; <Number of switch inputs and PPG output duty>
;
; +-----+
; | SW Inputs | PPG Output | LED |
; |           | Duty       | Brightness |
; |-----|-----|-----|
; | 0 times  | 10%       | 90% |
; | 1 time   | 30%       | 70% |
; | 2 times  | 50%       | 50% |
; | 3 times  | 70%       | 30% |
; | 4 times  | 90%       | 10% |
; | 5 times  | 70%       | 30% |
; | 6 times  | 50%       | 50% |
; | 7 times  | 30%       | 70% |
; |-----|-----|-----|
; # The PPG output duty from the zeroth switch input is repeated after
the eighth switch input.
;
;
;<<I/O port settings>>
;
; Input: P43
; Output: P00-P03, P20-P23, P30-P33, P40-P42, P44-P47, P120-P123, P130
; # All unused ports are set as the output mode.
;
;*****

;=====
;
; Define the symbol
;
;=====
OFFSETT EQU 43/4 ; For the correction of servicing time
corresponding to that of INTTM010

;=====
;
; Vector table
;
;=====
XVCTCSEG AT 0000H

```

```

DW   RESET_START      ;(00) RESET
DW   RESET_START      ;(02) --
DW   RESET_START      ;(04) --
DW   RESET_START      ;(06) INTLVI
DW   RESET_START      ;(08) INTP0
DW   INTERRUPT_P1     ;(0A) INTP1
DW   RESET_START      ;(0C) INTTMH1
DW   INTERRUPT_TM000  ;(0E) INTTM000
DW   INTERRUPT_TM010  ;(10) INTTM010
DW   RESET_START      ;(12) INTAD
DW   RESET_START      ;(14) --
DW   RESET_START      ;(16) INTP2
DW   RESET_START      ;(18) INTP3
DW   RESET_START      ;(1A) INTTM80
DW   RESET_START      ;(1C) INTSRE6
DW   RESET_START      ;(1E) INTSR6
DW   RESET_START      ;(20) INTST6

;=====
;
;   Define the ROM data table
;
;=====
XROMCSEG  AT      0100H
PWIDTH:   DW      40-1      ; Pulse width 20 us (duty 10%)
          DW      40*3-1    ; Pulse width 60 us (duty 30%)
          DW      40*5-1    ; Pulse width 100 us (duty 50%)
          DW      40*7-1    ; Pulse width 140 us (duty 70%)
          DW      40*9-1    ; Pulse width 180 us (duty 90%)
          DW      40*7-1    ; Pulse width 140 us (duty 70%)
          DW      40*5-1    ; Pulse width 100 us (duty 50%)
          DW      40*3-1    ; Pulse width 60 us (duty 30%)

;=====
;
;   Define the RAM
;
;=====
XRAMDSEG  SADDRP
NEXTD:    DS      2      ; For storing the new pulse width (duty)
CURRENTD: DS      2      ; For storing the current pulse width (duty)

;=====
;
;   Define the memory stack area
;
;=====
XSTKDSEG  AT      0FEE0H

```

```

STACKEND:
    DS    20H                ; Memory stack area = 32 bytes
STACKTOP:                    ; Start address of the memory stack area = FF00H

;*****
;
;   Initialization after RESET
;
;*****
XMAIN    CSEG    UNIT
RESET_START:
;-----
;   Initialize the stack pointer
;-----
    MOVW  AX,    #STACKTOP
    MOVW  SP,    AX        ; Set the stack pointer

;-----
;   Initialize the watchdog timer
;-----
    MOV   WDTM, #01110111B ; Stop the watchdog timer operation

;-----
;   Detect low-voltage + set the clock
;-----

;----- Set the clock <1> -----
    MOV   PCC,  #00000000B ; The clock supplied to the CPU (fcpu) = fxp (=
fx/4 = 2 MHz)
    MOV   LSRCM, #00000001B ; Stop the oscillation of the low-speed
internal oscillator

;----- Check the reset source -----
    MOV   A,    RESF        ; Read the reset source
    BT   A.0,   $SET_CLOCK ; Omit subsequent LVI-related processing and go
to SET_CLOCK during LVI reset

;----- Set low-voltage detection -----
    MOV   LVIS, #00000000B ; Set the low-voltage detection level (VLVI) to
4.3 V +/-0.2 V
    SET1  LVION            ; Enable the low-voltage detector operation

    MOV   A,    #40        ; Assign the 200 us wait count value
;----- 200 us wait -----
WAIT_200US:
    DEC   A
    BNZ  $WAIT_200US      ; 0.5[us/cclk] x 10[cclk] x 40[count] = 200[us]

```

```

;----- VDD >= VLVI wait processing -----
WAIT_LVI:
    NOP
    BT    LVIF, $WAIT_LVI    ; Branch if VDD < VLVI

    SET1  LVIMD              ; Set so that an internal reset signal is
generated when VDD < VLVI

;----- Set the clock <2> -----
SET_CLOCK:
    MOV   PPCC, #00000000B   ; The clock supplied to the peripheral hardware
(fxp) = fx (= 8 MHz)
                                ; -> The clock supplied to the CPU (fcpu) = fxp
= 8 MHz

;-----
;   Initialize the port 0
;-----
    MOV   P0,    #00000000B   ; Set output latches of P00-P03 as low
    MOV   PM0,   #11110000B   ; Set P00-P03 as output mode

;-----
;   Initialize the port 2
;-----
    MOV   P2,    #00000000B   ; Set output latches of P20-P23 as low
    MOV   PM2,   #11110000B   ; Set P20-P23 as output mode

;-----
;   Initialize the port 3
;-----
    MOV   P3,    #00000000B   ; Set output latches of P30-P33 as low
    MOV   PM3,   #11110000B   ; Set P30-P33 as output mode

;-----
;   Initialize the port 4
;-----
    MOV   P4,    #00000000B   ; Set output latches of P40-P47 as low
    MOV   PU4,   #00001000B   ; Connect on-chip pull-up resistor to P43
    MOV   PM4,   #00001000B   ; Set P40-P42 and P44-P47 as output mode, P43 as
input mode

;-----
;   Initialize the port 12
;-----
    MOV   P12,   #00000000B   ; Set output latches of P120-P123 as low
    MOV   PM12,  #11110000B   ; Set P120-P123 as output mode

;-----

```

```

; Initialize the port 13
;-----
MOV   P13,  #00000001B ; Set output latch of P130 as high

;-----
; Set 16-bit timer 00
;-----
MOV   CRC00,  #00000000B ; Use CR000 and CR010 as compare registers
MOVW  AX,    #400-1
MOVW  CR000,  AX          ; CR000 is used for setting the cycle
MOVW  AX,    #40-1
MOVW  CR010,  AX          ; CR010 is used for setting the pulse
width
MOV   PRM00,  #00000001B ; Count clock = fxp/4 (= 2 MHz)
MOV   TOC00,  #00011011B ; Reverse the output upon a match between
CR000 and CR010, set the initial output value to High, and enable timer output
MOV   TMC00,  #00001100B ; Start the timer operation (clear & start
upon a match between TM00 and CR000)

;-----
; Initialize the RAM and general-purpose register
;-----
MOVW  CURRENTD, AX        ; Save the initial pulse width value as
the current pulse width
MOVW  HL,    #PWIDTH      ; Specify the pulse width table address

;-----
; Set the interrupt
;-----
MOV   INTM0,  #00000000B ; Set the valid edge of INTP1 to falling
edge
MOV   IF0,   #00H        ; Clear invalid interrupt requests in advance
CLR1  PMK1              ; Unmask INTP1 interrupts

EI                                ; Enable vector interrupt

;*****
;
; Main loop
;
;*****
MAIN_LOOP:
    NOP
    BR   $MAIN_LOOP        ; Go to the MAIN_LOOP

;*****
;
; External interrupt INTP1

```

```

;
;*****
INTERRUPT_P1:
    PUSH  AX                ; Save the AX register data to the stack

;----- 10 ms wait to handle chattering -----
    MOV  A,    #50+1        ; 200 us x 50 = 10 ms
WAIT_CHAT:
    CLR1  TMIF000            ; Clear the INTTM000 interrupt request flag
WAIT_INT:
    BF    TMIF000, $WAIT_INT; Wait for the INTTM000 interrupt
    DEC  A                  ; Decrement the A register by 1
    BNZ  $WAIT_CHAT         ; Branch if not A = 0

    CLR1  PIF1              ; Clear the INTP1 interrupt request

;----- Identification of chattering detection -----
    BT    P4.3, $END_INTP1 ; Branch if there is no switch input

;----- Read the new pulse width -----
    MOV  A,    L            ; Read the table address
    ADD  A,    #2           ; Increment the table address by 2
    AND  A,    #00001111B  ; Mask bits other than bits 0 to 3
    MOV  L,    A
    MOV  A,    [HL]        ; Read the lower 8 bits of the pulse width
    XCH  A,    X
    MOV  A,    [HL+1]      ; Read the higher 8 bits of the pulse width
    MOVW NEXTD,    AX      ; Save the new pulse width

;----- Compare with the current pulse width -----
    MOVW AX,    CURRENTD   ; Read the current pulse width
    ADDW AX,    #OFFSETT   ; Add the offset time
    XCH  A,    X
    SUB  A,    [HL]        ; Compare with the new pulse width
    XCH  A,    X
    SUBC A,    [HL+1]

;----- Specify the interrupt servicing that changes the pulse width -----
    BNC  $DECDUTY          ; Branch if the new pulse width < current pulse
width
    CLR1  TMIF000            ; Clear invalid interrupt requests
    CLR1  TMMK000            ; Unmask INTTM000 interrupts
    SET1  TMMK010            ; Mask INTTM010 interrupts
    BR    $END_INTP1        ; Branch to END_INTP1
DECDUTY:
    CLR1  TMIF010            ; Clear invalid interrupt requests
    CLR1  TMMK010            ; Unmask INTTM010 interrupts
    SET1  TMMK000            ; Mask INTTM000 interrupts

```

```

END_INTPl:
    POP  AX          ; Restore the AX register data
    RETI            ; Return from interrupt servicing

;*****
;
;  Interrupt INTTM000
;
;*****
INTERRUPT_TM000:
    PUSH AX          ; Save the AX register data to the stack

    CLR1 TOC00.4    ; Disable output reversal
    MOVW AX, NEXTD
    MOVW CR010, AX  ; Change the pulse width (duty)
    NOP             ; Wait for one timer 00 count clock cycle
    NOP
    SET1 TOC00.4    ; Enable output reversal

    MOVW CURRENTD, AX; Save the new pulse width as the current pulse width
    SET1 TMMK000    ; Mask INTTM000 interrupts

    POP  AX          ; Restore the AX register data
    RETI            ; Return from interrupt servicing

;*****
;
;  Interrupt INTTM010
;
;*****
INTERRUPT_TM010:
    PUSH AX          ; Save the AX register data to the stack

    CLR1 TOC00.4    ; Disable output reversal
    MOVW AX, NEXTD
    MOVW CR010, AX  ; Change the pulse width (duty)
    NOP             ; Wait for one timer 00 count clock cycle
    NOP
    SET1 TOC00.4    ; Enable output reversal

    MOVW CURRENTD, AX; Save the new pulse width as the current pulse width
    SET1 TMMK010    ; Mask INTTM010 interrupts

    POP  AX          ; Restore the AX register data
    RETI            ; Return from interrupt servicing

end

```

● main.c (C language version)

```

/*****
 NEC Electronics      78K0S/KB1+

*****
 78K0S/KB1+ Sample program
*****
 16-bit timer 00 (PPG output)
*****
<<History>>
 2007.7.-- Release
*****

```

<<Overview>>

This sample program presents an example of using the PPG output function of 16-bit timer 00. Perform PPG output at a 200 us cycle and change the duty width by 20% for every switch input.

<Principal setting contents>

- Declare a function run by an interrupt: INTP1 -> fn\_intp1()
- Declare a function run by an interrupt: INTTM000 -> fn\_inttm000()
- Declare a function run by an interrupt: INTTM010 -> fn\_inttm010()
- Stop the watchdog timer operation
- Set the low-voltage detection voltage (VLVI) to 4.3 V +/-0.2 V
- Generate an internal reset signal (low-voltage detector) when VDD < VLVI after VDD >= VLVI
- Set the CPU clock to 8 MHz
- Set the clock supplied to the peripheral hardware to 8 MHz
- Set the valid edge of external interrupt INTP1 to falling edge
- Set the chattering detection time during switch input to 10 ms

<16-bit timer 00 settings>

- Operation mode: Clear & start the timer count upon a match between TM00 and CR000
- Count clock = f<sub>xp</sub>/4 (2 MHz)
- Use CR000 and CR010 as compare registers
- Initialize the cycle of CR000 to 200 us
- Initialize the pulse width of CR010 to 20 us
- Reverse the output upon a match between CR000 and CR010 (PPG output)
- Set the initial output value (= active level) to High
- Set the output latch of P31 to Low (to use the T000 output)
- Set P31 to the output mode (to use the T000 output)

<Number of switch inputs and PPG output duty>

SW Inputs	PPG Output Duty	LED Brightness
0 times	10%	90%
1 time	30%	70%
2 times	50%	50%
3 times	70%	30%
4 times	90%	10%
5 times	70%	30%
6 times	50%	50%
7 times	30%	70%

# The PPG output duty from the zeroth switch input is repeated after the eighth switch input.

<<I/O port settings>>

Input: P43

Output: P00-P03, P20-P23, P30-P33, P40-P42, P44-P47, P120-P123, P130

# All unused ports are set as the output mode.

\*\*\*\*\*

/\*=====

Preprocessing directive (#pragma)

=====\*/

```
#pragma SFR /* SFR names can be described at the C
source level */
#pragma EI /* EI instructions can be described at the
C source level */
#pragma DI /* DI instructions can be described at the
C source level */
#pragma NOP /* NOP instructions can be described at
the C source level */
#pragma interrupt INTPl fn_intpl /* Interrupt function declaration:INTPl */
#pragma interrupt INTTM000 fn_inttm000 /* Interrupt function
declaration:INTTM000 */
#pragma interrupt INTTM010 fn_inttm010 /* Interrupt function
declaration:INTTM010 */
```

```

#define offsetT 56/4          /* For the correction of servicing time
corresponding to that of INTTM010 */

/*=====

Define the global variables and constant table

=====*/
sreg unsigned char g_ucSWcnt = 0; /* 8-bit variable for counting the number
of switch inputs */
sreg unsigned int g_unNextD = 40-1; /* 16-bit variable for saving the
new pulse width (duty) */
sreg unsigned int g_unCurrentD = 40-1; /* 16-bit variable for saving the
current pulse width (duty) */
const unsigned int g_unOutData[8] = {40-1,40*3-1,40*5-1,40*7-1,40*9-1,40*7-
1,40*5-1,40*3-1}; /* Output duty table */

/*****

Initialization after RESET

*****/
void hdwinit(void){
    unsigned char ucCnt200us; /* 8-bit variable for 200 us wait */

/*-----
Initialize the watchdog timer + detect low-voltage + set the clock
-----*/

/* Initialize the watchdog timer */
WDTM = 0b01110111; /* Stop the watchdog timer operation */

/* Set the clock <1> */
PCC = 0b00000000; /* The clock supplied to the CPU (fcpu) =
fxp (= fx/4 = 2 MHz) */
LSRCM = 0b00000001; /* Stop the oscillation of the low-speed
internal oscillator */

/* Check the reset source */
if (!(RESF & 0b00000001)){ /* Omit subsequent LVI-related processing
during LVI reset */

    /* Set low-voltage detection */
    LVIS = 0b00000000; /* Set the low-voltage detection level
(VLVI) to 4.3 V +-0.2 V */
    LVION = 1; /* Enable the low-voltage detector operation */

```

```

        for (ucCnt200us = 0; ucCnt200us < 9; ucCnt200us++){ /* Wait of
about 200 us */
            NOP();
        }

        while (LVIF){ /* Wait for VDD >= VLVI */
            NOP();
        }

        LVIMD = 1; /* Set so that an internal reset signal is
generated when VDD < VLVI */
    }

    /* Set the clock <2> */
    PPCC = 0b00000000; /* The clock supplied to the peripheral
hardware (fxp) = fx (= 8 MHz)
                                -> The clock supplied to the CPU
(fcpu) = fxp = 8 MHz */

/*-----
    Initialize the port 0
-----*/
    P0    = 0b00000000; /* Set output latches of P00-P03 as low */
    PM0   = 0b11110000; /* Set P00-P03 as output mode */

/*-----
    Initialize the port 2
-----*/
    P2    = 0b00000000; /* Set output latches of P20-P23 as low */
    PM2   = 0b11110000; /* Set P20-P23 as output mode */

/*-----
    Initialize the port 3
-----*/
    P3    = 0b00000000; /* Set output latches of P30-P33 as low */
    PM3   = 0b11110000; /* Set P30-P33 as output mode */

/*-----
    Initialize the port 4
-----*/
    P4    = 0b00000000; /* Set output latches of P40-P47 as low */
    PU4   = 0b00001000; /* Connect on-chip pull-up resistor to P43
*/
    PM4   = 0b00001000; /* Set P40-P42 and P44-P47 as output mode,
P43 as input mode */

/*-----
    Initialize the port 12

```

```

-----*/
    P12  = 0b00000000;          /* Set output latches of P120-P123 as low
*/
    PM12 = 0b11110000;          /* Set P120-P123 as output mode */

/*-----
    Initialize the port 13
-----*/
    P13  = 0b00000001;          /* Set output latch of P130 as high */

/*-----
    Set 16-bit timer 00
-----*/
    CRC00 = 0b00000000;          /* Use CR000 and CR010 as compare
registers */
    CR000 = 400-1;              /* CR000 is used for setting the cycle */
    CR010 = 40-1;              /* CR010 is used for setting the pulse
width */
    PRM00 = 0b00000001;          /* Count clock = fxp/4 (= 2 MHz) */
    TOC00 = 0b00011011;          /* Reverse the output upon a match between
CR000 and CR010, set the initial output value to High, and enable timer output
*/
    TMC00 = 0b00001100;          /* Start the timer operation (clear &
start upon a match between TM00 and CR000) */

/*-----
    Set the interrupt
-----*/
    INTM0 = 0b00000000;          /* Set the valid edge of INTP1 to falling
edge */
    IF0 = 0x00;                 /* Clear invalid interrupt requests */
    PMK1 = 0;                   /* Unmask INTP1 interrupts */

    return;
}

/*****

Main loop

*****/
void main(void){

    EI();                        /* Enable vector interrupt */

    while (1){
        NOP();
    }
}

```

```

}

/*****

Interrupt INTPl

*****/
__interrupt void fn_intpl(){
    unsigned char ucChat;          /* 8-bit variable for removing chattering
*/

    for (ucChat = 0; ucChat <50+1; ucChat++){ /* Loop for a 10 ms wait */
        TMIF000 = 0;                /* Clear the INTTM000 interrupt flag */
        while (!TMIF000){          /* Wait for the INTTM000 interrupt */
            NOP();
        }
    }

    PIF1 = 0;                       /* Clear the INTPl interrupt request */

    if (!P4.3){                    /* Processing performed if SW is on for 10 ms or more
*/
        g_ucSWcnt += 1;            /* Increment the number of switch inputs
by 1 */
        g_ucSWcnt &= 0b00000111;   /* Mask bits other than bits 0 to 2
*/

        g_unNextD = g_unOutData[g_ucSWcnt]; /* Save the new pulse width */

        if (g_unNextD > g_unCurrentD + offsetT){ /* Use an INTTM000
interrupt to increase the pulse width */
            TMIF000 = 0;          /* Clear invalid interrupt requests
*/

            TMMK000 = 0;         /* Unmask INTTM000 interrupts */
            TMMK010 = 1;        /* Mask INTTM010 interrupts */
        }
        else {                  /* Use an INTTM010 interrupt to decrease
the pulse width */
            TMIF010 = 0;        /* Clear invalid interrupt requests
*/

            TMMK010 = 0;        /* Unmask INTTM010 interrupts */
            TMMK000 = 1;        /* Mask INTTM000 interrupts */
        }
    }

    return;
}

```

```

/*****

Interrupt INTTM000

*****/
__interrupt void fn_inttm000(){

    TOC00.4 = 0;          /* Disable output reversal */
    CR010 = g_unNextD;    /* Change the pulse width (duty) */
    NOP();                /* Wait for one timer 00 count clock cycle
*/
    NOP();
    TOC00.4 = 1;          /* Enable output reversal */

    g_unCurrentD = g_unNextD; /* Save the new pulse width as the current
pulse width */
    TMMK000 = 1;          /* Mask INTTM000 interrupts */

    return;
}

```

```

/*****

Interrupt INTTM010

*****/
__interrupt void fn_inttm010(){

    TOC00.4 = 0;          /* Disable output reversal */
    CR010 = g_unNextD;    /* Change the pulse width (duty) */
    NOP();                /* Wait for one timer 00 count clock cycle
*/
    NOP();
    TOC00.4 = 1;          /* Enable output reversal */

    g_unCurrentD = g_unNextD; /* Save the new pulse width as the current
pulse width */
    TMMK010 = 1;          /* Mask INTTM010 interrupts */

    return;
}

```

● op.asm (Common to assembly language and C language versions)

```

;=====
;
; Option byte
;

```

```

;=====
OPBTCSEG AT 0080H
          DB 10011100B      ; Option byte area
;          |||
;          |||+----- Low-speed internal oscillator can be
stopped by software
;          |++----- High-speed internal oscillation clock (8
MHz) is selected for system clock source
;          +----- P34/RESET pin is used as RESET pin

          DB 11111111B      ; Protect byte area (for the self
programming mode)
;          |||||
;          ++++++----- All blocks can be written or erased

end

```

## APPENDIX B REVISION HISTORY

The mark "<R>" shows major revised points. The revised points can be easily searched by copying an "<R>" in the PDF file and specifying it in the "Find what." field.

Edition	Date Published	Page	Revision
1st edition	December 2007	–	–
2nd edition	September 2008	p.29	CHAPTER 5 OPERATION CHECK USING SYSTEM SIMULATOR SM+ • Modification of description in Caution ((as of September 2007) → (as of July 2008))
		pp.29 to 31	Modification of 5.1 Building the Sample Program
		p.31	5.2 Operation with SM+ • Addition of (1)
		p.36	CHAPTER 6 RELATED DOCUMENTS • Addition of Flash Programming Manual (Basic) MINICUBE2 version

*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/>