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

78K0S/Kx1+

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

Pulse Width Measurement

This document describes an operation overview of the sample program and how to use it, as well as how to set and use the pulse width measurement function of 16-bit timer/event counter 00. In the sample program, the pulse width of the signal input from the TI000 pin is measured by using the pulse width measurement function of 16-bit timer/event counter 00.

Target devices

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

CONTENTS

CHAPTER 1 OVERVIEW **3**
1.1 Main Contents of the Initial Settings 3
1.2 Contents Following the Main Loop 4
CHAPTER 2 CIRCUIT DIAGRAM **5**
CHAPTER 3 SOFTWARE **6**
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
CHAPTER 4 SETTING METHODS **10**
4.1 Setting the Pulse Width Measurement Function of 16-bit Timer/ Event Counter 00 10
4.2 Timing When Generation of INTTM000 and INTTM010 Interrupts Conflict (When Measuring the Width of a Pulse that is Longer than a TM00 Counter Cycle) 32
CHAPTER 5 OPERATION CHECK USING THE DEVICE **33**
5.1 Building the Sample Program 33
5.2 Operation with the Device 36
CHAPTER 6 RELATED DOCUMENTS **38**
APPENDIX A PROGRAM LIST **39**
APPENDIX B REVISION HISTORY **53**

Document No. U18889EJ2V0AN00 (2nd edition)
Date Published September 2008 NS

© NEC Electronics Corporation 2007
Printed in Japan

- **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).

M8E 02. 11-1

CHAPTER 1 OVERVIEW

An example of using the pulse width measurement function of 16-bit timer/event counter 00 is presented in this sample program. The pulse width of the signal input from the TI000 pin is measured eight times.

1.1 Main Contents of the Initial Settings

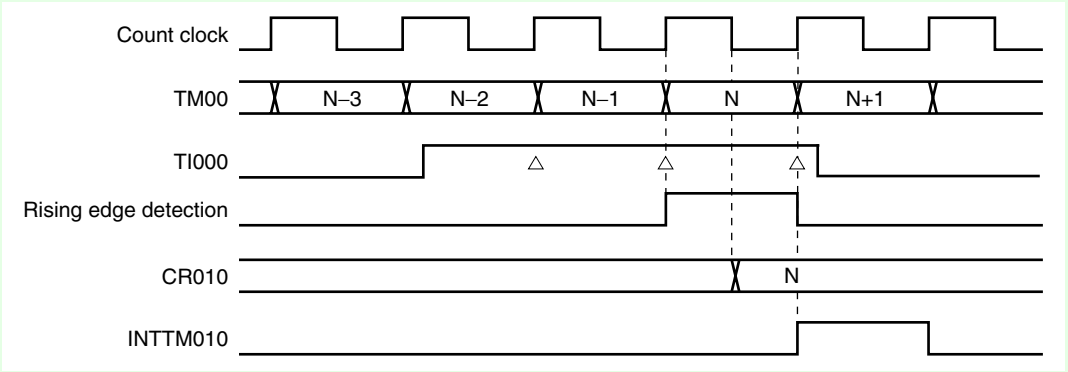
The main contents of the initial settings are as follows.

- Selecting the high-speed internal oscillator as the system clock source[Note]
- Stopping watchdog timer operation
- Setting V_{LVI} (low-voltage detection voltage) to 4.3 V ±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 the operation modes of CR000 and CR010 as a compare register and a capture register, respectively
  - Setting “FFFFH” to CR000
  - Setting the valid edge of the TI000 pin to both the falling and rising edges and setting the count clock to $f_{XP}/2^2$ (2 MHz)
  - Setting the operation mode to clear & start upon detection of the valid edge of the TI000 pin
- Enabling INTTM000 and INTTM010 interrupts

Note This is set by using the option byte.

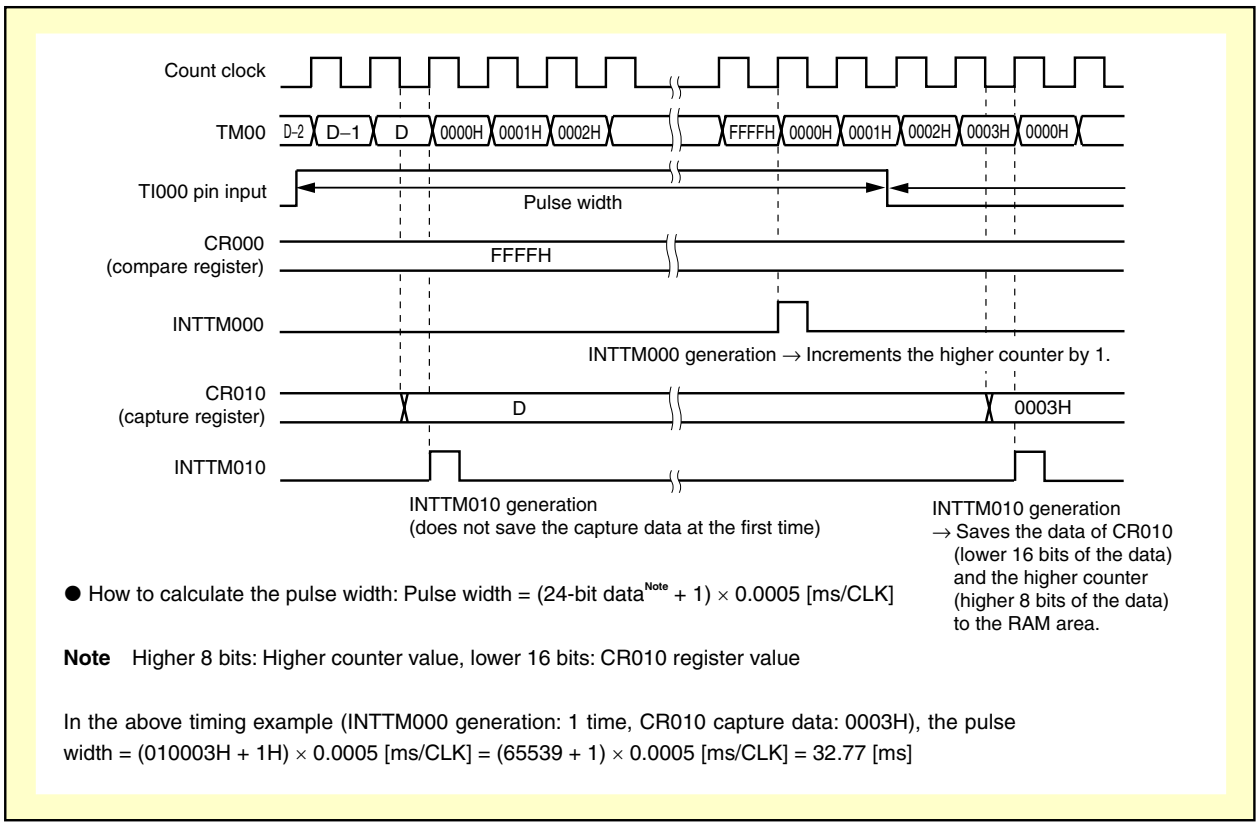
[Column] Capture operation timing

With 16-bit timer/event counter 00, capture operation is performed for the first time when the valid level of the TI000 pin or TI010 pin is detected twice, in order to eliminate the noise with a short pulse width. An input pulse that is longer than two count clock cycles is therefore required. The following diagram shows an operation example in which CR010 is captured when the rising edge is specified.



1.2 Contents Following the Main Loop

The pulse width of the signal input from the TI000 pin is measured eight times by using the generation of 16-bit timer/event counter 00 interrupts (INTTM000 and INTTM010), after completion of the initial settings.



● How to calculate the pulse width: Pulse width = (24-bit data[Note] + 1) × 0.0005 [ms/CLK]

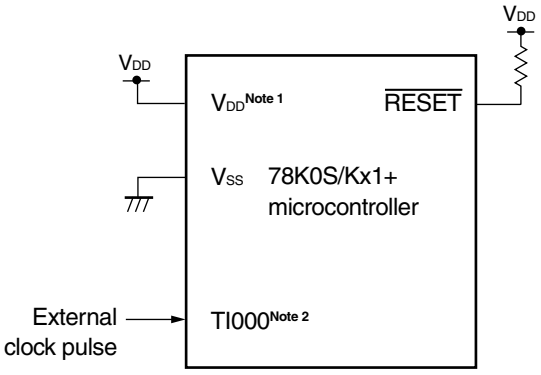
Note Higher 8 bits: Higher counter value, lower 16 bits: CR010 register value

In the above timing example (INTTM000 generation: 1 time, CR010 capture data: 0003H), the pulse width = (010003H + 1H) × 0.0005 [ms/CLK] = (65539 + 1) × 0.0005 [ms/CLK] = 32.77 [ms]

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

CHAPTER 2 CIRCUIT DIAGRAM

This chapter describes a circuit diagram to be used in this sample program.



Notes 1. Use this in a voltage range of 4.5 V ≤ V_{DD} ≤ 5.5 V.

2. TI000/INTP0/P30: 78K0S/KA1+ and 78K0S/KB1+ microcontrollers
TI000/ANI0/TOH1/P20: 78K0S/KY1+ and 78K0S/KU1+ microcontrollers

Cautions 1. Connect the AV_{REF} pin directly to V_{DD} (only for the 78K0S/KA1+ and 78K0S/KB1+ microcontrollers).

2. Connect the AV_{SS} 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_{REF} and AV_{SS} pins.

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	●Note
op.asm	Assembler source file for setting the option byte (sets the system clock source)	●	●
tm00cap.prw	Work space file for integrated development environment PM+		●
tm00cap.prj	Project file for integrated development environment PM+		●

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

Remark : Only the source file is included.

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

3.2 Internal Peripheral Functions to Be Used

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

- Pulse width measurement function: 16-bit timer/event counter 00
- $V_{DD} < V_{LVI}$ detection: Low-voltage detector (LVI)
- External pulse input: TI000[Note]

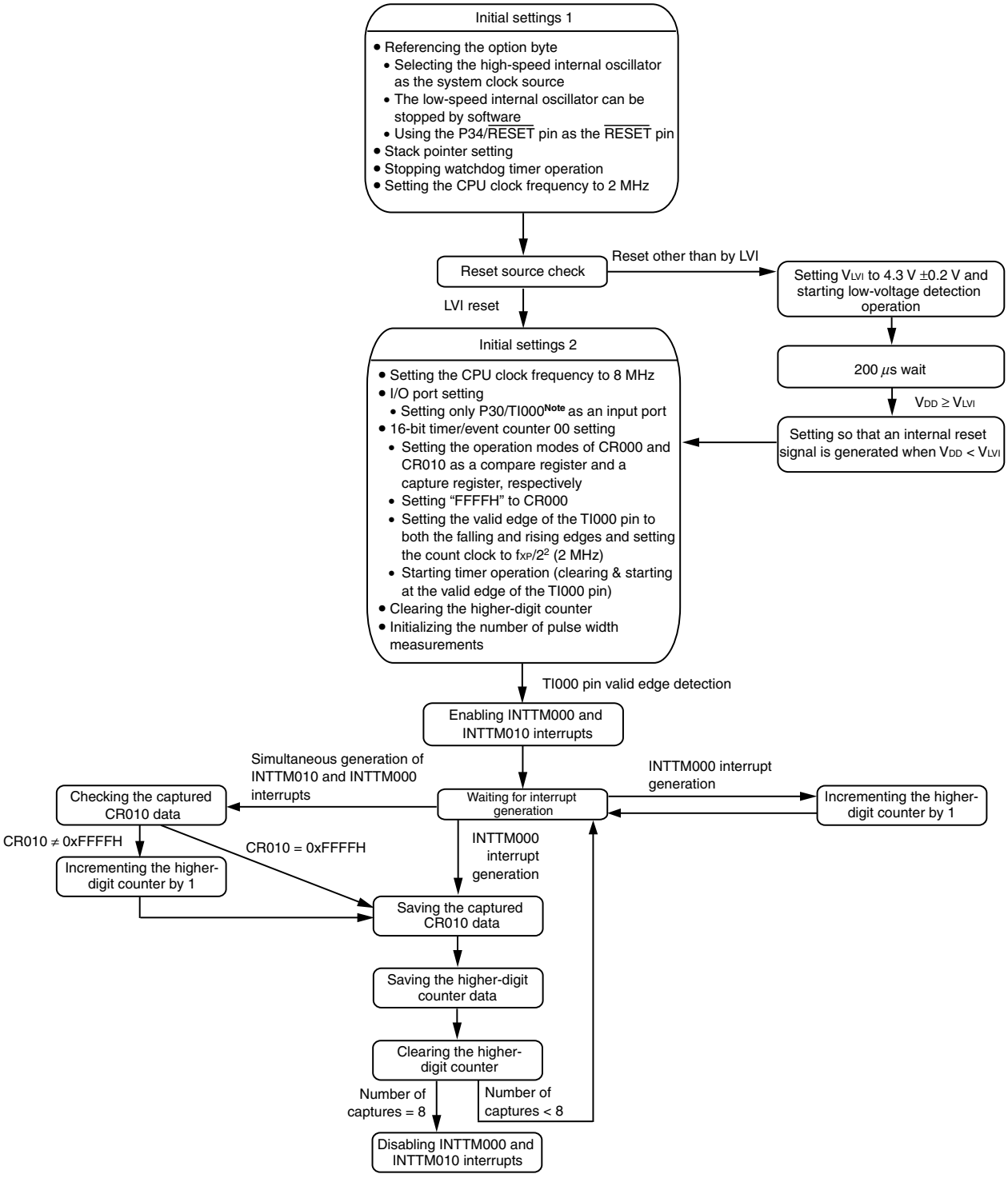
Note TI000/INTP0/P30: 78K0S/KA1+ and 78K0S/KB1+ microcontrollers
TI000/ANI0/TOH1/P20: 78K0S/KY1+ and 78K0S/KU1+ microcontrollers

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 (pulse width measurement function), and setting of interrupts are performed.

The pulse width of the signal input from the TI000 pin is measured eight times by using the generation of 16-bit timer/event counter 00 interrupts (INTTM000 and INTTM010), after completion of the initial settings.

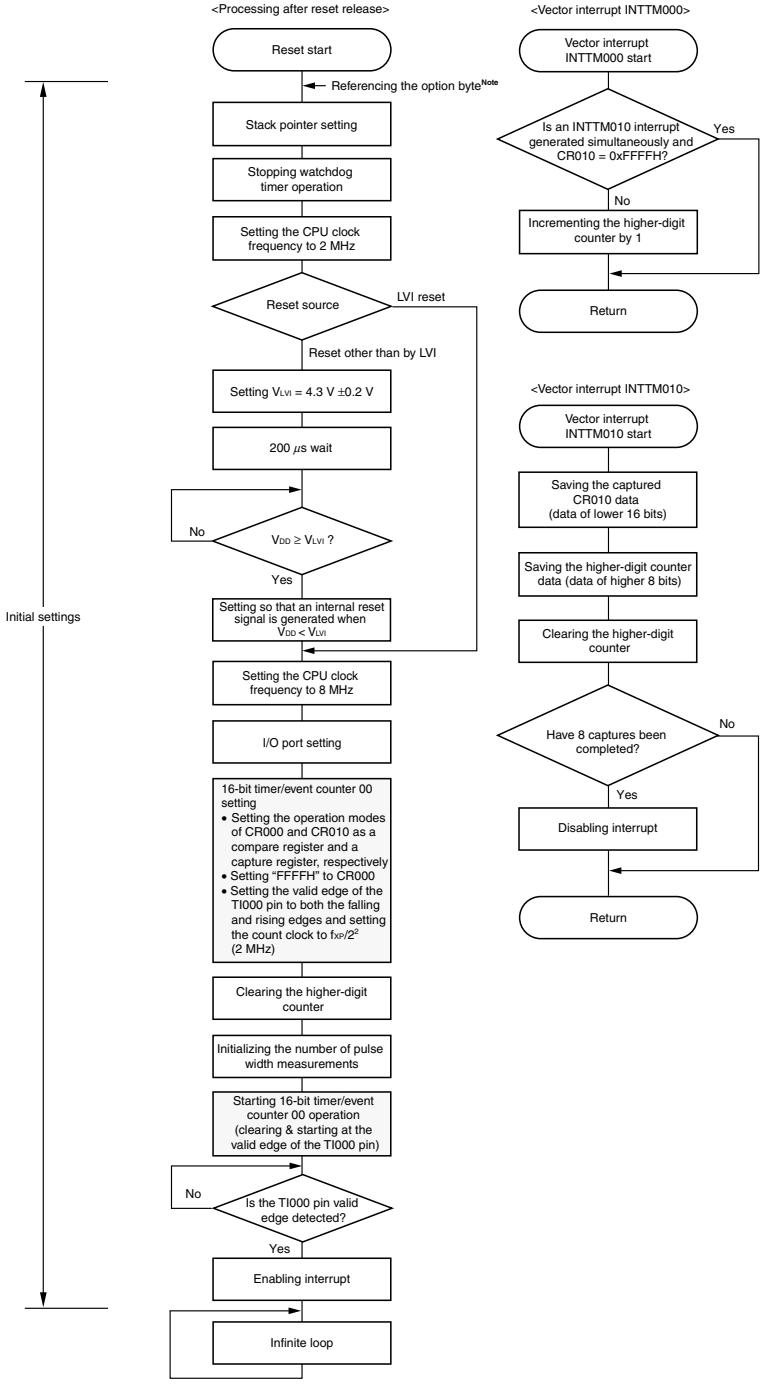
The details are described in the status transition diagram shown below.



Note TI000/P30: 78K0S/KA1+ and 78K0S/KB1+ microcontrollers
TI000/P20: 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/$\overline{\text{RESET}}$ pin as the $\overline{\text{RESET}}$ pin

CHAPTER 4 SETTING METHODS

This chapter describes the pulse width measurement 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 Pulse Width Measurement Function of 16-bit Timer/Event Counter 00

The following seven types of registers are set when using the pulse width measurement function of 16-bit timer/event counter 00.

- Capture/compare control register 00 (CRC00)
- Prescaler mode register 00 (PRM00)
- 16-bit timer mode control register 00 (TMC00)
- 16-bit timer capture/compare register 000 (CR000)
- 16-bit timer capture/compare register 010 (CR010)
- Port mode register x (PMx)[Note]
- Port mode control register x (PMCx)[Note]

Note Set as follows, because the pulse width measurement function uses only the TI000 pin or the TI000 pin and TI010 pin as the timer input.

- TI000 pin

	PMx Register	PMCx Register
78K0S/KA1+ and 78K0S/KB1+ microcontrollers	PM30 = 1	Setting not required
78K0S/KY1+ and 78K0S/KU1+ microcontrollers	PM20 = 1	PMC20 = 0

- TI010 pin

	PMx Register	PMCx Register
78K0S/KA1+ and 78K0S/KB1+ microcontrollers	PM31 = 1	Setting not required
78K0S/KY1+ and 78K0S/KU1+ microcontrollers	PM21 = 1	PMC21 = 0

<Example of the basic operation setting procedure when using 16-bit timer/event counter 00 for pulse width measurement>

<1> Setting the CRC00 register

<2> Setting the count clock using the PRM00 register

<3> Setting the TMC00 register: starting operation

Caution Steps <1> and <2> 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)

CRC00

0	0	0	0	0	CRC002	CRC001	CRC000

CR000 operation mode selection

0	Operates as a compare register.
1	Operates as a capture register.

CR000 capture trigger selection

0	Captures at the valid edge of the TI010 pin.
1	Captures at the reverse phase of the valid edge of the TI000 pin[Note].

CR010 operation mode selection

0	Operates as a compare register.
1	Operates as a capture register.

Note If CRC000 is 1, when both the falling and rising edges are selected as the valid edge of the TI000 pin, the CR000 register cannot perform capture operation.

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.

3. The capture trigger requires a pulse that is longer than 2 cycles of the count clock selected by using prescaler mode register 00 (PRM00) for surely performing capture.

(2) 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 4-2. Format of Prescaler Mode Register 00 (PRM00)

PRM00

ES110	ES100	ES010	ES000	0	0	PRM001	PRM000

Count clock (fsam) selection

0	0	f_{XP}
0	1	$f_{XP}/2^2$
1	0	$f_{XP}/2^8$
1	1	TI000 pin valid edge[Note]

TI000 pin valid edge selection

0	0	Falling edge
0	1	Rising edge
1	0	Setting prohibited
1	1	Both rising and falling edges

TI010 pin valid edge selection

0	0	Falling edge
0	1	Rising edge
1	0	Setting prohibited
1	1	Both rising and falling edges

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 as the capture trigger, it is sampled with the count clock that is selected with prescaler mode register 00 (PRM00) 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.

(3) 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 4-3. Format of 16-bit Timer Mode Control Register 00 (TMC00)

TMC00

0	0	0	0	TMC003	TMC002	TMC001	OVF00

Overflow detection of 16-bit timer counter 00 (TM00)

0	Overflow not detected.
1	Overflow detected.

TMC003	TMC002	TMC001	Operation mode and clear mode selection	TO00 inversion timing selection	Interrupt request generation
0	0	0	Operation stop (TM00 cleared to 0)	No change	Not generated
0	0	1			
0	1	0	Free-running mode	Match between TM00 and CR000 or match between TM00 and CR010	<When used as compare register> Generated upon match between TM00 and CR000, or match between TM00 and CR010 <When used as capture register> Generated at valid edge of TI000 pin or TI010 pin
0	1	1		Match between TM00 and CR000, match between TM00 and CR010, or valid edge of TI000 pin	
1	0	0	Clear & start occurs at valid edge of TI000 pin	–	
1	0	1			
1	1	0	Clear & start occurs upon match between TM00 and CR000	Match between TM00 and CR000 or match between TM00 and CR010	
1	1	1		Match between TM00 and CR000, match between TM00 and CR010, or valid edge of TI000 pin	

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.

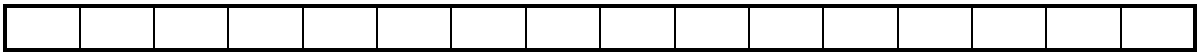
8. Capture operation is performed at the fall of the count clock. An interrupt request (INTTM0n0: n = 0, 1), however, occurs at the rise of the next count clock.

(4) Setting the CR000 register

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

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

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.
- When using CR000 as a capture register
  The valid edges of the TI000 pin or TI010 pin can be selected as the capture trigger. The valid edges of the TI000 and TI010 pins are set by using the **PRM00 register**.

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 CR000 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 CR000 register value is less than the TM00 counter value, the TM00 counter continues counting, overflows, and then starts counting from 0 again. If the new CR000 register value is less than the old value, therefore, the timer must be reset and restarted after the value of the CR000 register 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.

Cautions 5. When CR000 is used as the capture register, if the register read period and capture trigger input conflict, the capture trigger input takes precedence and the CR000 read data becomes undefined. If the timer count stop and capture trigger input conflict, the capture data becomes undefined.

6. Changing the CR000 register setting during TM00 counter operation may cause a malfunction.

(5) Setting the CR010 register

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

Figure 4-5. Format of 16-bit Timer Capture/Compare Register 010 (CR010)

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.
- When using CR010 as a capture register
  The valid edge of the TI000 pin can be selected as the capture trigger. The valid edge of the TI000 pin is set by using the **PRM00 register**.

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 CR010 register value is less than the TM00 counter value, the TM00 counter continues counting, overflows, and then starts counting from 0 again. If the new CR010 register value is less than the old value, therefore, the timer must be reset and restarted after the value of the CR010 register 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. When CR010 is used as the capture register, if the register read period and capture trigger input conflict, the capture trigger input takes precedence and the CR010 read data becomes undefined. If the timer count stop and capture trigger input conflict, the capture data becomes undefined.

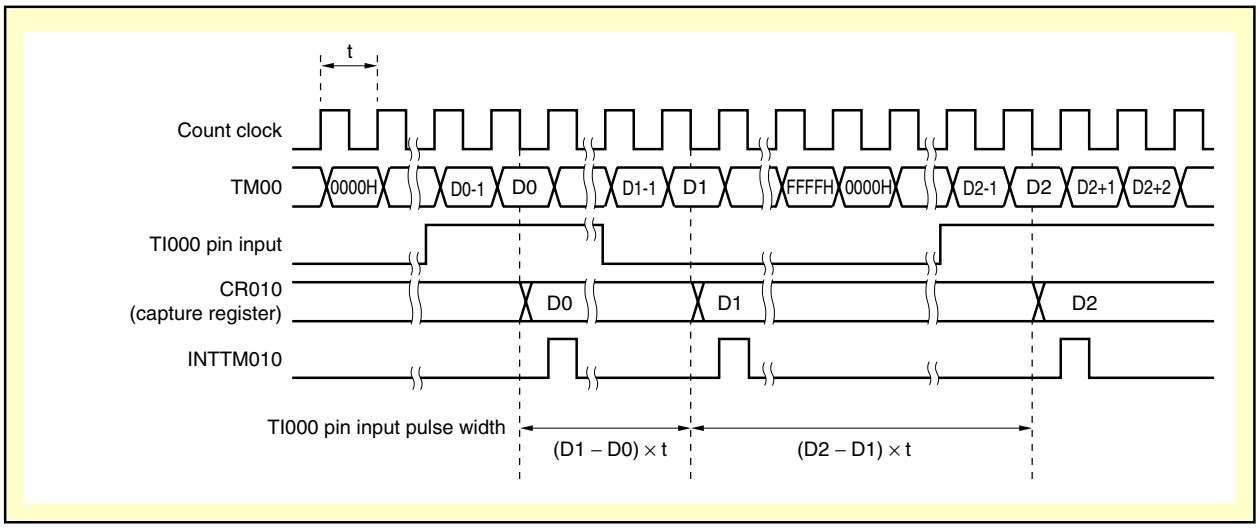
6. Changing the CR010 register setting during TM00 counter operation may cause a malfunction.

[Example 1] Measuring the pulse width with the input signal of the TI000 pin (using the CR010 register as a capture register, free-running mode)

The pulse width of the signal input to the TI000 pin is measured when the TM00 counter is operated in the free-running mode. The TM00 counter count value is captured to the CR010 register when the valid edge of the TI000 pin is detected.

Caution The pulse width that can be measured in this operation example is up to one timer counter cycle.

Figure 4-6. Timing Example When Measuring the Pulse Width with the Input Signal of the TI000 Pin (Free-Running Mode, Both-Edge Specification)



(1) Register settings

<1> CRC00

0	0	0	0	0	1	x	0

CR000 operation mode selection

0	Operates as a compare register.

CR000 capture trigger selection

x	Invalid setting

CR010 operation mode selection

1	Operates as a capture register.

<2> PRM00

x	x	0/1	0/1	0	0	0/1	0/1

Count clock (fsam) selection (Setting “1, 1” is prohibited.)

0	0	f_{XP}
0	1	$f_{XP}/2^2$
1	0	$f_{XP}/2^8$

TI000 pin valid edge selection (Setting “1, 0” is prohibited.)

0	0	Falling edge
0	1	Rising edge
1	1	Both rising and falling edges

TI010 pin valid edge selection

x	x	Invalid setting (Setting “1, 0” is prohibited.)

<3> TMC00

0	0	0	0	0	1	x	0

Overflow detection of 16-bit timer counter 00 (TM00)

0	Overflow not detected.

Operation mode and clear mode selection

0	1	0	Free-running mode
0	1	1	

<4> PMx, PMCx

	PMx Register	PMCx Register
78K0S/KA1+ and 78K0S/KB1+ microcontrollers	PM30 = 1	Setting not required
78K0S/KY1+ and 78K0S/KU1+ microcontrollers	PM20 = 1	PMC20 = 0

(2) Sample program

In the example below, “x” in **(1) Register settings** is set to “0”. Furthermore, the valid edge of the TI000 pin is set to both edges and the count clock is set to f_{XP} (system clock frequency).

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

```
SET1   PM3.0
MOV    CRC00, #00000100B
MOV    PRM00, #00110000B
MOV    TMC00, #00000100B
```

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

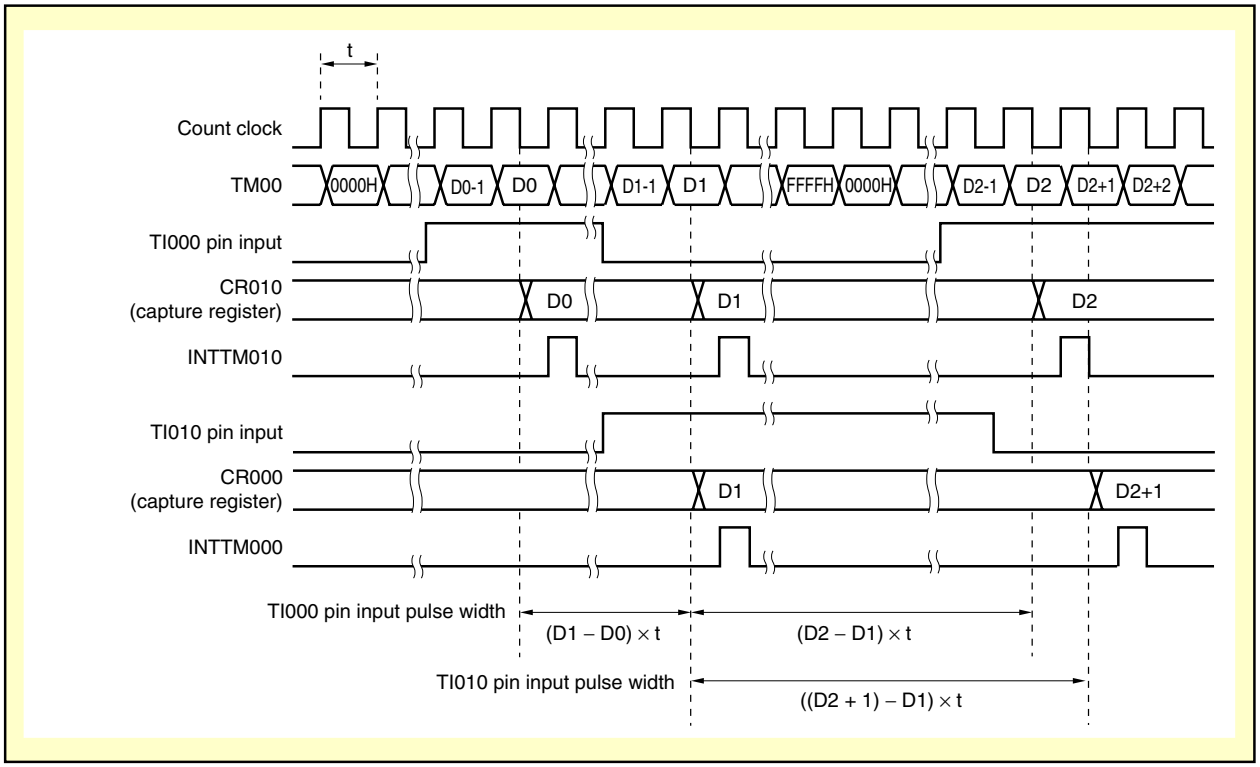
```
PM3.0 = 1;
CRC00 = 0b00000100;
PRM00 = 0b00110000;
TMC00 = 0b00000100;
```

[Example 2] Measuring the pulse width with the input signals of the TI000 pin and TI010 pin (using the CR000 register and CR010 register as capture registers, free-running mode)

The pulse widths of the two signals input to the TI000 pin and TI010 pin are measured simultaneously when the TM00 counter is operated in the free-running mode. The TM00 counter count value is captured to the CR010 register when the valid edge of the TI000 pin is detected and the TM00 counter count value is captured to the CR000 register when the valid edge of the TI010 pin is detected.

Caution The pulse width that can be measured in this operation example is up to one timer counter cycle.

Figure 4-7. Timing Example When Measuring the Pulse Width with the Input Signals of the TI000 Pin and TI010 Pin (Free-Running Mode, Both-Edge Specification)



(1) Register settings

<1> CRC00

0	0	0	0	0	1	0	1

CR000 operation mode selection

1	Operates as a capture register.

CR000 capture trigger selection

0	Captures at the valid edge of the TI010 pin.

CR010 operation mode selection

1	Operates as a capture register.

<2> PRM00

0/1	0/1	0/1	0/1	0	0	0/1	0/1

Count clock (fsam) selection (Setting “1, 1” is prohibited.)

0	0	f_{XP}
0	1	$f_{XP}/2^2$
1	0	$f_{XP}/2^8$

TI000 pin valid edge selection (Setting “1, 0” is prohibited.)

0	0	Falling edge
0	1	Rising edge
1	1	Both rising and falling edges

TI010 pin valid edge selection (Setting “1, 0” is prohibited.)

0	0	Falling edge
0	1	Rising edge
1	1	Both rising and falling edges

<3> TMC00

0	0	0	0	0	1	x	0

Overflow detection of 16-bit timer counter 00 (TM00)

0	Overflow not detected.

Operation mode and clear mode selection

0	1	0	Free-running mode
0	1	1	

<4> PMx, PMCx

	PMx Register	PMCx Register
78K0S/KA1+ and 78K0S/KB1+ microcontrollers	PM30 = 1, PM31 = 1	Setting not required
78K0S/KY1+ and 78K0S/KU1+ microcontrollers	PM20 = 1, PM21 = 1	PMC20 = 0, PMC21 = 0

(2) Sample program

In the example below, “x” in **(1) Register settings** is set to “0”. Furthermore, the valid edges of the TI000 pin and TI010 pin are set to both edges and the count clock is set to f_{XP} (system clock frequency).

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

```
SET1   PM3.0
SET1   PM3.1
MOV    CRC00, #00000101B
MOV    PRM00, #11110000B
MOV    TMC00, #00000100B
```

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

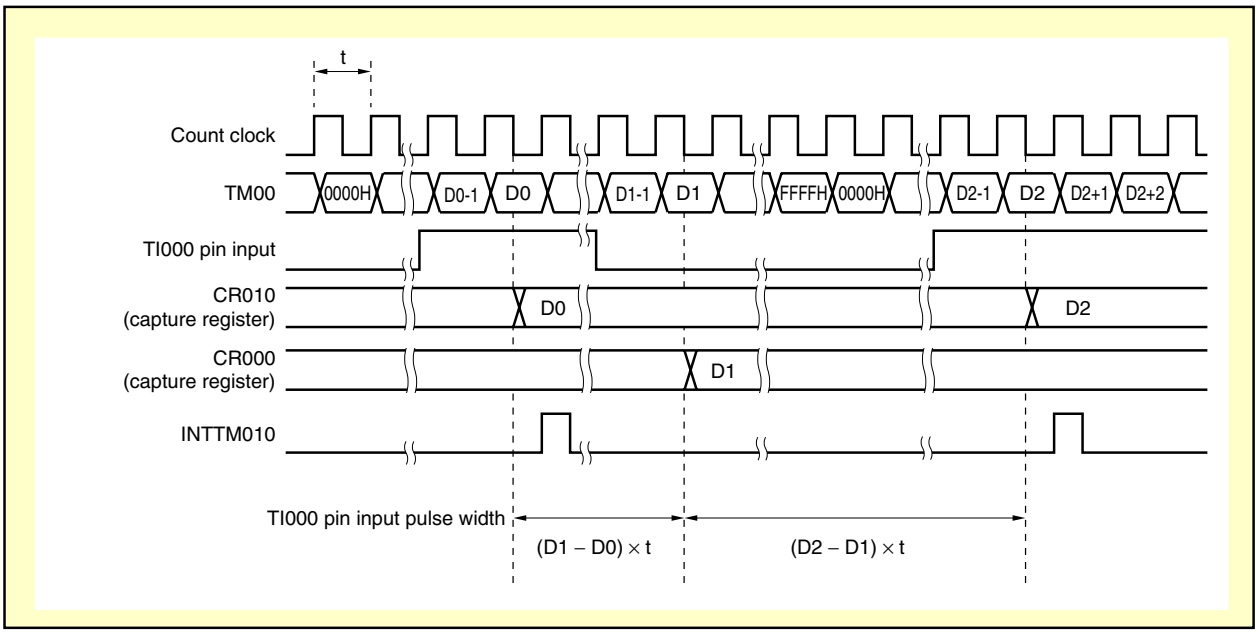
```
PM3.0 = 1;
PM3.1 = 1;
CRC00 = 0b00000101;
PRM00 = 0b11110000;
TMC00 = 0b00000100;
```

[Example 3] Measuring the pulse width with the input signal of the TI000 pin (using the CR000 register and CR010 register as capture registers, free-running mode)

The pulse width of the signal input to the TI000 pin is measured when the TM00 counter is operated in the free-running mode. The TM00 counter count value is captured to the CR010 register when the valid edge of the TI000 pin is detected and the TM00 counter count value is captured to the CR000 register by the phase reverse to that when the valid edge of the TI000 pin is detected. Set the valid edge detection of the TI000 pin as the rising edge or falling edge.

Caution The pulse width that can be measured in this operation example is up to one timer counter cycle.

Figure 4-8. Timing Example When Measuring the Pulse Width with the Input Signal of the TI000 Pin (Free-Running Mode, Rising-Edge Specification)



(1) Register settings

<1> CRC00

0	0	0	0	0	1	1	1

CR000 operation mode selection

1	Operates as a capture register.

CR000 capture trigger selection

1	Captures at the reverse phase of the valid edge of the TI000 pin.[Note]

CR010 operation mode selection

1	Operates as a capture register.

Note If CRC000 is 1, when both the falling and rising edges are selected as the valid edge of the TI000 pin, the CR000 register cannot perform capture operation. If CRC001 is 1, capture to the CR000 register by the valid edge of the TI010 pin cannot be performed; however, the TI010 pin can be used as an external interrupt, because INTTM000 is generated.

<2> PRM00

x	x	0	0/1	0	0	0/1	0/1

Count clock (fsam) selection (Setting “1, 1” is prohibited.)

0	0	f_{XP}
0	1	$f_{XP}/2^2$
1	0	$f_{XP}/2^8$

TI000 pin valid edge selection

0	0	Falling edge
0	1	Rising edge

TI010 pin valid edge selection

x	x	Invalid setting (Setting “1, 0” is prohibited.)

<3> TMC00

0	0	0	0	0	1	x	0

Overflow detection of 16-bit timer counter 00 (TM00)

0	Overflow not detected.

Operation mode and clear mode selection

0	1	0	Free-running mode
0	1	1	

<4> PMx, PMCx

	PMx Register	PMCx Register
78K0S/KA1+ and 78K0S/KB1+ microcontrollers	PM30 = 1	Setting not required
78K0S/KY1+ and 78K0S/KU1+ microcontrollers	PM20 = 1	PMC20 = 0

(2) Sample program

In the example below, “x” in **(1) Register settings** is set to “0”. Furthermore, the valid edge of the TI000 pin is set to rising edge and the count clock is set to f_{XP} (system clock frequency).

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

```
SET1   PM3.0
MOV    CRC00, #00000111B
MOV    PRM00, #00010000B
MOV    TMC00, #00000100B
```

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

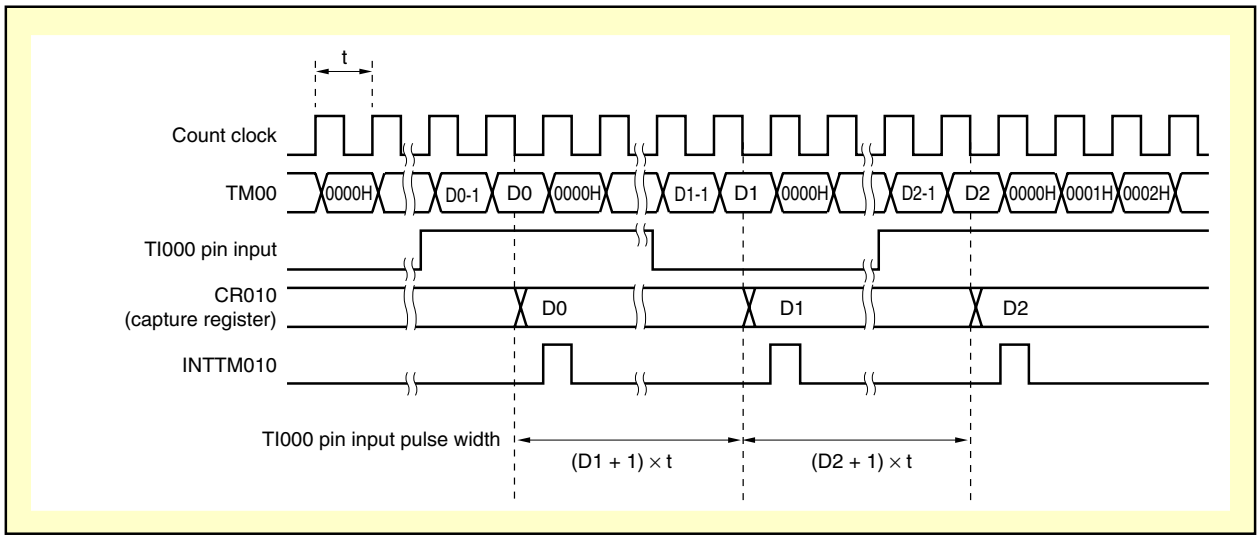
```
PM3.0 = 1;
CRC00 = 0b00000111;
PRM00 = 0b00010000;
TMC00 = 0b00000100;
```

[Example 4] Measuring the pulse width with the input signal of the TI000 pin (using the CR000 register as a capture register, clear & start mode entered by inputting the valid edge of the TI000 pin)

The pulse width of the signal input to the TI000 pin is measured when the TM00 counter is operated in the clear & start mode entered by inputting the valid edge of the TI000 pin. The TM00 counter is cleared, counting is restarted, and the pulse width of the signal input to the TI000 pin is measured after the TM00 counter count value is captured to the CR010 register by detecting the valid edge of the TI000 pin.

Caution The pulse width that can be measured in this operation example is up to one timer counter cycle.

Figure 4-9. Timing Example When Measuring the Pulse Width with the Input Signal of the TI000 Pin (Clear & Start Mode Entered by TI000 Pin Valid Edge Input, Both-Edge Specification)



(1) Register settings

<1> CRC00

0	0	0	0	0	1	x	0

CR000 operation mode selection

0	Operates as a compare register

CR000 capture trigger selection

x	Invalid setting

CR010 operation mode selection

1	Operates as a capture register.

<2> PRM00

x	x	0/1	0/1	0	0	0/1	0/1

Count clock (fsam) selection (Setting “1, 1” is prohibited.)

0	0	f_{XP}
0	1	$f_{XP}/2^2$
1	0	$f_{XP}/2^8$

TI000 pin valid edge selection (Setting “1, 0” is prohibited.)

0	0	Falling edge
0	1	Rising edge
1	1	Both rising and falling edges

TI010 pin valid edge selection

x	x	Invalid setting (Setting “1, 0” is prohibited.)

<3> TMC00

0	0	0	0	1	0	x	0

Overflow detection of 16-bit timer counter 00 (TM00)

0	Overflow not detected.

Operation mode and clear mode selection

1	0	0	Clear & start occurs at valid edge of TI000 pin
1	0	1	

<4> PMx, PMCx

	PMx Register	PMCx Register
78K0S/KA1+ and 78K0S/KB1+ microcontrollers	PM30 = 1	Setting not required
78K0S/KY1+ and 78K0S/KU1+ microcontrollers	PM20 = 1	PMC20 = 0

(2) Sample program

In the example below, “x” in **(1) Register settings** is set to “0”. Furthermore, the valid edge of the TI000 pin is set to both edges and the count clock is set to f_{XP} (system clock frequency).

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

```
SET1   PM3.0
MOV    CRC00, #00000100B
MOV    PRM00, #00110000B
MOV    TMC00, #00001000B
```

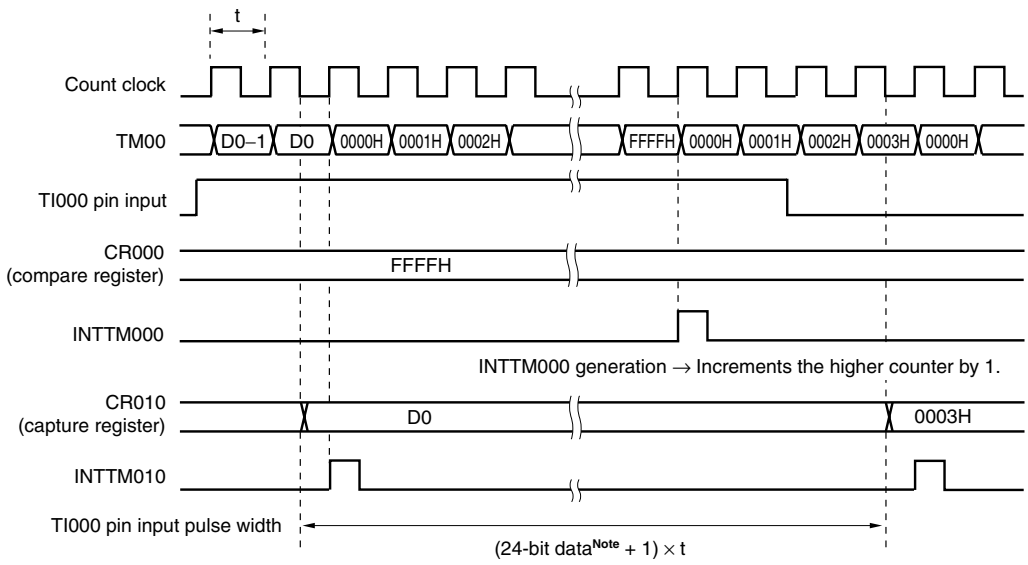
<2> C language (when using the 78K0S/KA1+ and 78K0S/KB1+ microcontrollers)

```
PM3.0 = 1;
CRC00 = 0b00000100;
PRM00 = 0b00110000;
TMC00 = 0b00001000;
```

[Example 5] Measuring with the input signal of the TI000 pin the width of a pulse that is longer than a TM00 counter cycle (using the CR010 register as a capture register and the CR000 register as a compare register, clear & start mode entered by inputting the valid edge of the TI000 pin)
(Same contents as in this sample program source)

The pulse width of the signal input to the TI000 pin is measured when the TM00 counter is operated in the clear & start mode entered by inputting the valid edge of the TI000 pin.
The values of the CR000 register and TM00 counter match when the operation mode of the CR000 register is set as a compare register and CR000 is set to FFFFH, and an INTTM000 interrupt is generated when the TM00 counter changes from FFFFH to 0000H. The width of a pulse that is longer than a TM00 counter cycle can be measured by using this interrupt as an overflow interrupt and counting up the higher digits when an INTTM000 interrupt is generated.
The TM00 counter is cleared, counting is restarted, and the pulse width of the signal input to the TI000 pin is measured after the TM00 counter count value is captured to the CR010 register by detecting the valid edge of the TI000 pin.

Figure 4-10. Timing Example When Measuring with the Input Signal of the TI000 Pin the Width of a Pulse that is Longer than a TM00 Counter Cycle (Clear & Start Mode Entered by TI000 Pin Valid Edge Input, Both-Edge Specification)



Note Higher 8 bits: Higher counter value, lower 16 bits: CR010 register value

*In the above-mentioned timing example (INTTM000 generation: 1 time, CR010 capture data: 0003H), the pulse width = (010003H + 1H) × t = (65539 + 1) × t

(1) Register settings

<1> CRC00

0	0	0	0	0	1	x	0

CR000 operation mode selection

0	Operates as a compare register

CR000 capture trigger selection

x	Invalid setting

CR010 operation mode selection

1	Operates as a capture register.

<2> PRM00

x	x	0/1	0/1	0	0	0/1	0/1

Count clock (fsam) selection (Setting “1, 1” is prohibited.)

0	0	f_{XP}
0	1	$f_{XP}/2^2$
1	0	$f_{XP}/2^8$

TI000 pin valid edge selection (Setting “1, 0” is prohibited.)

0	0	Falling edge
0	1	Rising edge
1	1	Both rising and falling edges

TI010 pin valid edge selection

x	x	Invalid setting (Setting “1, 0” is prohibited.)

<3> TMC00

0	0	0	0	1	0	x	0

Overflow detection of 16-bit timer counter 00 (TM00)

0	Overflow not detected.

Operation mode and clear mode selection

1	0	0	Clear & start occurs at valid edge of TI000 pin
1	0	1	

<4> PMx, PMCx

	PMx Register	PMCx Register
78K0S/KA1+ and 78K0S/KB1+ microcontrollers	PM30 = 1	Setting not required
78K0S/KY1+ and 78K0S/KU1+ microcontrollers	PM20 = 1	PMC20 = 0

<5> CR000: FFFFH

(2) Sample program

In the example below, “x” in **(1) Register settings** is set to “0”. Furthermore, the valid edge of the TI000 pin is set to both edges and the count clock (fsam) is set to $f_{XP}/2^2$.

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

```
SET1   PM3.0
MOV    CRC00, #00000100B
MOVW   AX, #0FFFFH
MOVW   CR000, AX
MOV    PRM00, #00110001B
MOV    TMC00, #00001000B
```

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

```
PM3.0 = 1;
CRC00 = 0b00000100;
CR000 = 0xFFFF;
PRM00 = 0b00110001;
TMC00 = 0b00001000;
```

[Excerpt from this sample program source]

An excerpt from **APPENDIX A PROGRAM LIST**, which is related to the 16-bit timer/event counter 00 function, is shown below (same contents as in [**Example 5**] mentioned above).

(1) Assembly language

```
XMAIN  CSEG   UNIT
RESET_START:
                    •
                    •
                    •
        MOV    CRC00, #00000100B    ; Use CR010 as a capture register
        MOVW   AX,    #0FFFFH
        MOVW   CR000, AX            ; Use CR000 for overflow detection
        MOV    PRM00, #00110001B    ; Specify the valid edge of the TI000 pin to both
edges, set the count clock to fxp/4
        MOV    TOC00, #00000000B    ; Do not perform timer output

        MOV    MK0,   #0FFH         ; Mask all interrupts at first
        MOV    IF0,   #00H          ; Clear invalid interrupt requests in advance
                    •
                    •
                    •
        MOV    TMC00, #00001000B    ; Start the timer operation (clear & start by the
valid edge of the TI000 pin)
WAITCAP:
        NOP
        BF     TMIF010, $WAITCAP    ; Wait for completion of the first capture

        MOV    IF0,   #00H          ; Clear the interrupt request flag
        CLR1   TMMK000              ; Unmask INTTM000 interrupts
        CLR1   TMMK010              ; Unmask INTTM010 interrupts

        EI;                         ; Enable vector interrupt

MAIN_LOOP:
        NOP
        BR     $MAIN_LOOP           ; Go to the MAIN_LOOP

INTERRUPT_TM000:
        PUSH   AX                   ; Save the AX register data to the stack
                    •
                    •
                    •
        INC    HIGHCOUNT            ; Increment the higher digit count by 1
                    •
                    •
                    •
INTERRUPT_TM010:
        PUSH   AX                   ; Save the AX register data to the stack
        MOVW   AX,    CR010         ; Read the captured value
                    •
                    •
                    •
        MOV    A,     HIGHCOUNT     ; Read the higher digits
                    •
                    •
                    •
end
```

Setting FFFFH to CR000

Setting the operation mode of CR010 as a capture register, that of CR000 as a compare register

Setting the count clock and the valid edge of the TI000 pin

Starting timer operation

Clearing the INTTM000 and INTTM010 interrupt request flags

Enabling INTTM000 interrupt servicing

Enabling INTTM010 interrupt servicing

Starting interrupt servicing by INTTM000 interrupt generation

Incrementing the higher-digit counter by 1

Starting interrupt servicing by INTTM010 interrupt generation

Reading the CR010 register capture data

Reading the higher-digit counter

(2) C language

```
void hdwinit(void){
        unsigned char ucCnt200us;       /* 8-bit variable for 200 us wait */
                        •
                        •
                        •
        CRC00 = 0b00000100;             /* Use CR010 as a capture register */
        CR000 = 0xFFFF;                 /* Use CR000 for overflow detection */
        PRM00 = 0b00110001;             /* Specify the valid edge of the TI000 pin to
both edges, set the count clock to fxp/4 */
        TOC00 = 0b00000000;             /* Do not perform timer output */

        MK0   = 0xFF;                   /* Mask all interrupts at first */
        IF0   = 0x00;                   /* Clear invalid interrupt requests */

        return;
}

void main(void){
        g_ucHighCount = 0;              /* Clear the higher-digit counter */
        g_ucTimes = 8;                  /* Initialize the number of measurements */
        TMC00 = 0b00001000;             /* Start the timer operation (clear & start by
the valid edge of the TI000 pin) */

        while(TMIF010==0){              /* Wait for completion of the first capture */
                NOP();
        }

        IF0 = 0x00;                     /* Clear the interrupt request flag */
        TMMK000 = 0;                    /* Unmask INTTM000 interrupts */
        TMMK010 = 0;                    /* Unmask INTTM010 interrupts */

        EI();                           /* Enable vector interrupt */

        while(1){                       /* Infinite loop */
                NOP();
                NOP();
        }
}

__interrupt void fn_inttm000(){
        if (!(TMIF010 && (CR010 == 0xFFFF))){   /*  Do  not  count  if  interrupts  are
generated simultaneously and if CR010 == 0xFFFF */
                g_ucHighCount++;        /* Increment the higher digit count by 1 */
        }

        return;
}
__interrupt void fn_inttm010(){
        g_unLowLength[8 - g_ucTimes] = CR010;   /* Read the captured value */
        g_unHighLength[8 - g_ucTimes] = g_ucHighCount; /* Read the higher digits */
                        •
                        •
                        •
        return;
}
```

Setting FFFFH to CR000

Setting the operation mode of CR010 as a capture register, that of CR000 as a compare register

Setting the count clock and the valid edge of the TI000 pin

Starting timer operation

Clearing the INTTM000 and INTTM010 interrupt request flags

Enabling INTTM000 interrupt servicing

Enabling INTTM010 interrupt servicing

Starting interrupt servicing by INTTM000 interrupt generation

Incrementing the higher-digit counter by 1

Starting interrupt servicing by INTTM010 interrupt generation

Reading the CR010 register capture data

Reading the higher-digit counter

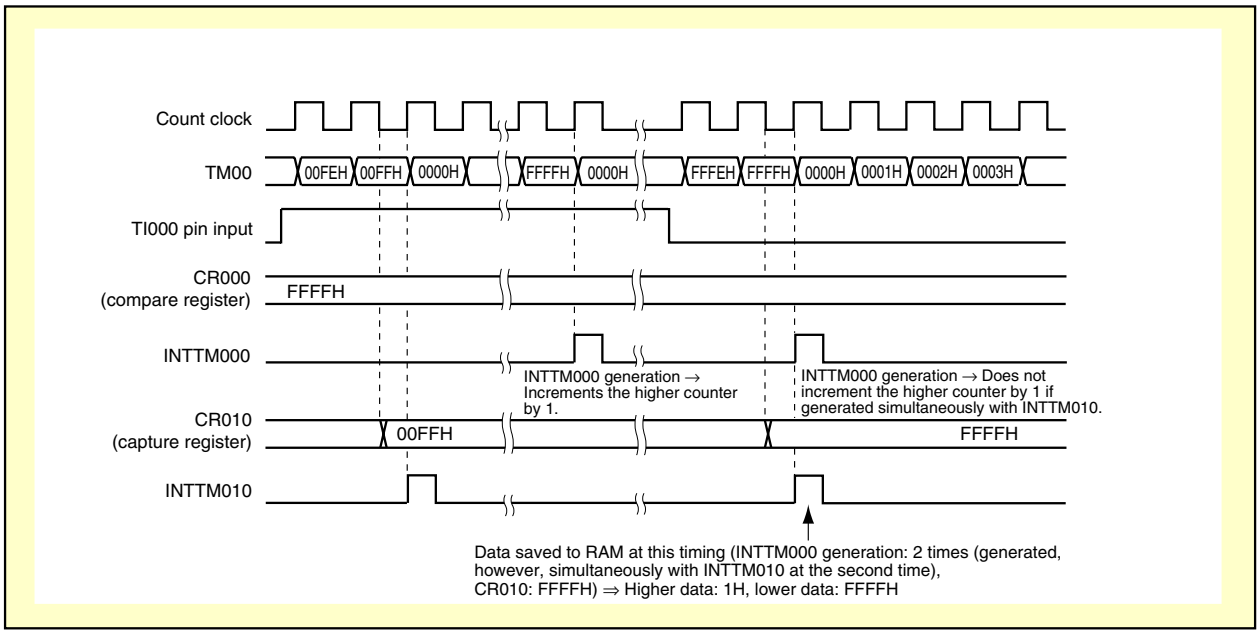
4.2 Timing When Generation of INTTM000 and INTTM010 Interrupts Conflict (When Measuring the Width of a Pulse that is Longer than a TM00 Counter Cycle)

The operation modes of the CR010 and CR000 registers are set as a capture register and as a compare register, respectively, and the CR000 register is set to FFFFH in this sample program. The clear & start mode entered by the valid edge of the TI000 pin is set (see [**Example 5**] above).

The values of the CR000 register and TM00 counter match due to these settings and an INTTM000 interrupt is generated when the TM00 counter changes from FFFFH to 0000H. The width of a pulse that is longer than a TM00 counter cycle can be measured by using this interrupt as an overflow interrupt and counting up the higher digits when an INTTM000 interrupt is generated.

When an INTTM000 interrupt is generated simultaneously with an INTTM010 interrupt, INTTM000 interrupt servicing takes precedence. At this time, The CR010 register capture data becomes FFFFH and the higher digits need not to be counted up. The higher digits, therefore, are not counted up, and the higher digit data and CR010 register capture data (FFFFH) are saved to the RAM area when an INTTM000 interrupt is generated simultaneously with an INTTM010 interrupt.

Figure 4-11. Timing Example When INTTM000 and INTTM010 Interrupts Conflict (Clear & Start Mode Entered by TI000 Pin Valid Edge Input, Both-Edge Specification)



CHAPTER 5 OPERATION CHECK USING THE DEVICE

This chapter describes the flow from building to the operation check using the device, using the downloaded sample program.

<R> 5.1 Building the Sample Program

This section describes how to build sample programs, using the 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.

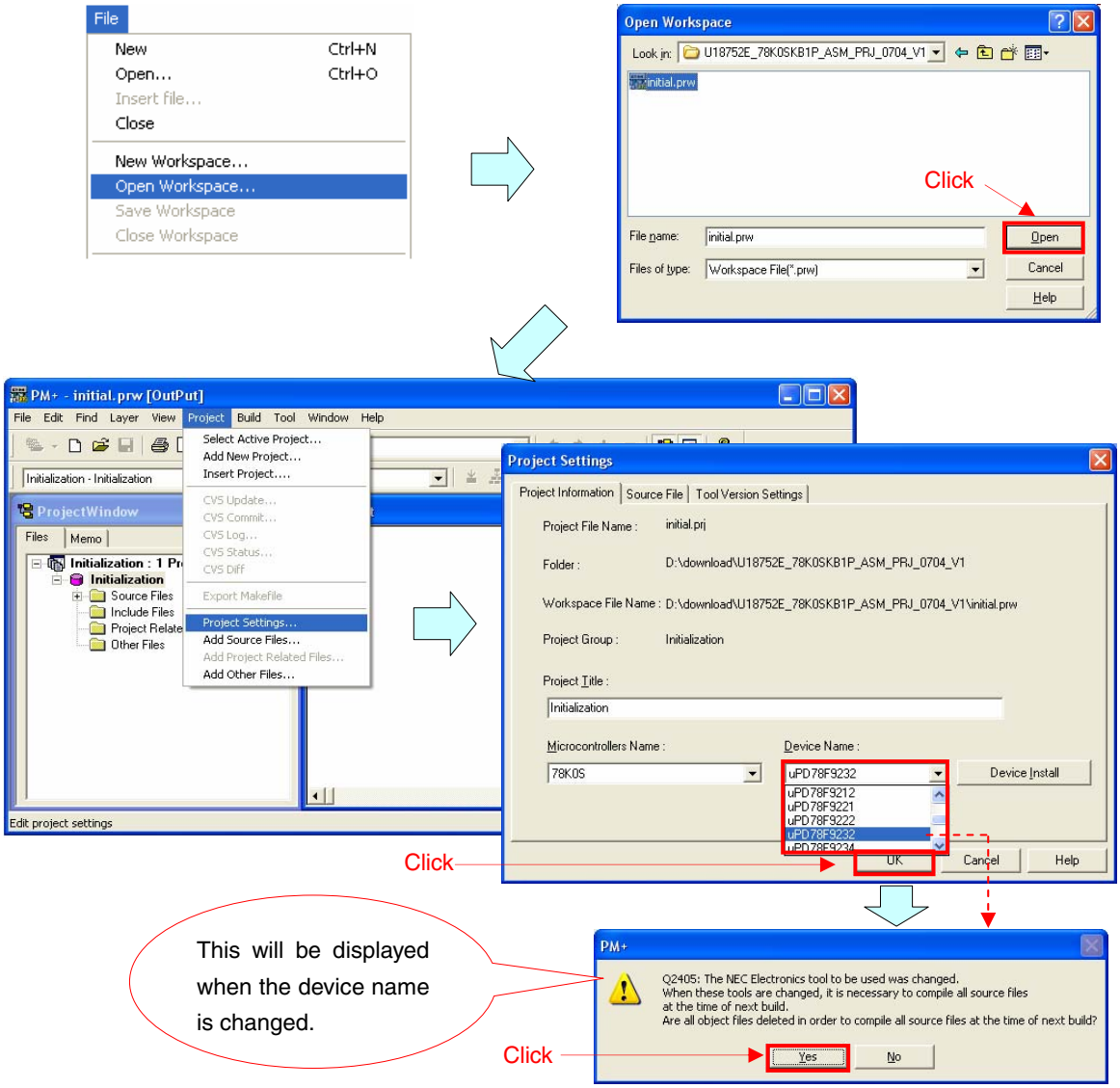
The [Using Fixed Area of Standard Library] check box is unchecked by default when the file that has been downloaded by clicking the icon is used in this sample program.

(1) Start PM+.

(2) Select “tm00cap.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].

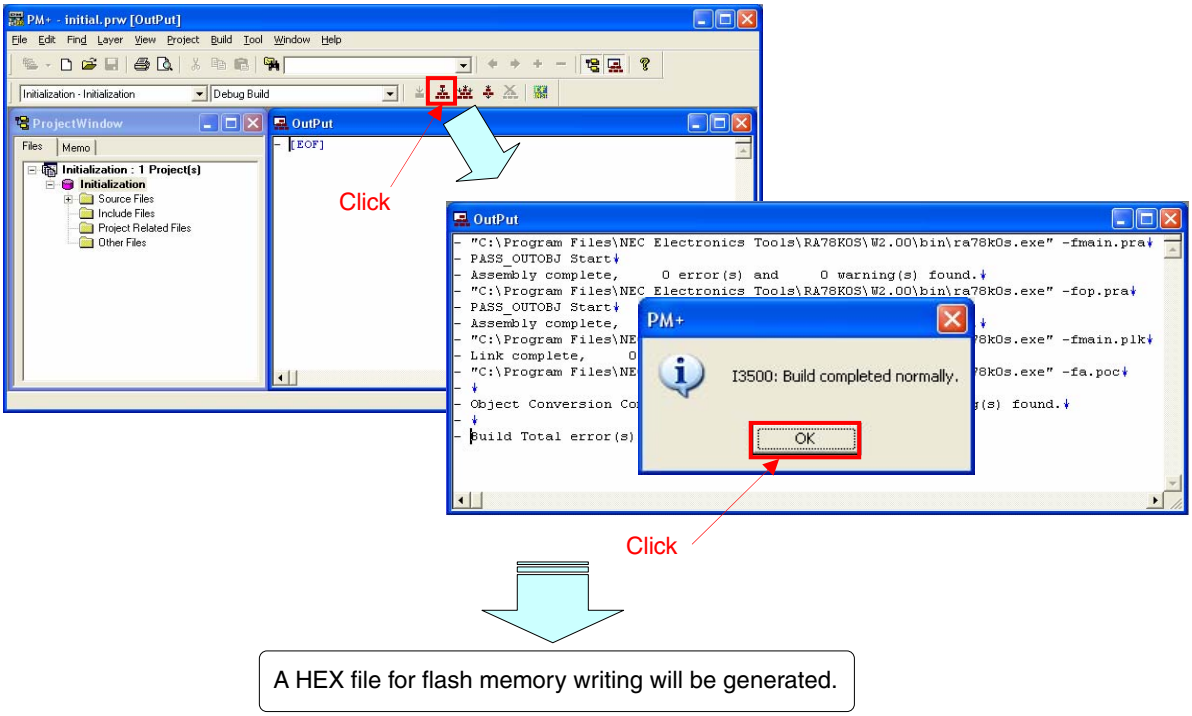
Remark Screenshots of the Sample Program (Initial Settings) LED Lighting Switch Control are shown below.



(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.



A HEX file for flash memory writing will be generated.

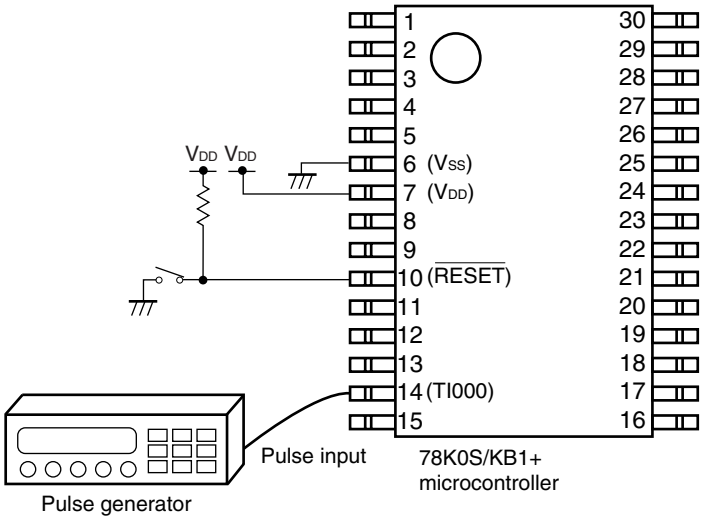
5.2 Operation with the Device

This section describes an example of an operation check using the device.

The HEX file generated by executing build can be written to the flash memory of the device.

For how to write to the flash memory of the device, refer to the **Flash Programming Manual (Basic) MINICUBE2 version** of each product **(78K0S/KU1+, 78K0S/KY1+, 78K0S/KA1+, 78K0S/KB1+)**.

An example of how to connect the device and peripheral hardware (pulse generator) to be used is shown below.



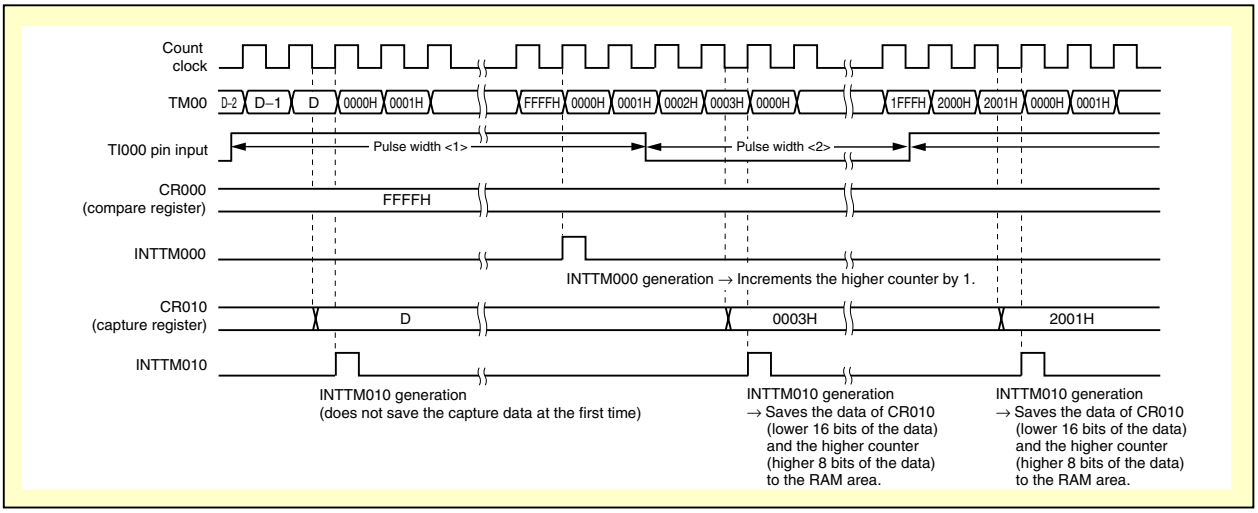
An operation example of the device that wrote this sample program is shown below.

(1) Operation during detection of the first valid edge

When a pulse is input to the TI000 pin, the first valid edge acts as a trigger and pulse width measurement is started.

(2) Operation during detection of the second and following valid edges

The high-level pulse width and low-level pulse width are captured alternately and the captured data is stored in the RAM, because the valid edge of the TI000 pin is detected both on the falling and rising edges.



<Expression for calculating the pulse width of this sample program>

Pulse width = (24-bit data[Note] + 1) × 0.0005 [ms/CLK]

Note Higher 8 bits: Higher counter value, lower 16 bits: CR010 register value

Pulse width <1> and pulse width <2> in the above-mentioned operation example are as follows.

- Pulse width <1> (INTTM000 generation: 1 time, CR010 capture data: 0003H)
  = (010003H + 1H) × 0.0005 [ms/CLK] = (65539 + 1) × 0.0005 [ms/CLK] = 32.77 [ms])
- Pulse width <2> (INTTM000 generation: 0 times, CR010 capture data: 2001H)
  = (002001H + 1H) × 0.0005 [ms/CLK] = (8193 + 1) × 0.0005 [ms/CLK] = 4.097 [ms])

(3) Operation following the eighth pulse width measurement

When eight pulse width values have been stored to the RAM, no further pulse width measurements will be performed.

CHAPTER 6 RELATED DOCUMENTS

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

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 (pulse width measurement)
;*******************************************************************************
;<<History>>
;   2007.7.--   Release
;*******************************************************************************
;
;<<Overview>>
;
;This sample program presents an example of using the pulse width measurement
;function of 16-bit timer 00.  The timer count value when the count clock is
;set to fxp/4 (= 2 MHz) and both the rising and falling edges of the signal
;input to the TI000 pin are detected is captured to the CRC010 register.
;The width or interval of the pulse input to the TI000 pin can be known from
;the captured value.  The width or interval of a pulse whose length is at
;least the timer cycle can be measured by setting the CR000 register to
;0xFFFF and generating an overflow interrupt.
;
;
;  <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
;  - Use the DE and HL registers for interrupt servicing (similarly as a
global variable)
;
;
;  <16-bit timer 00 settings>
;  - Operation mode: Clear & start the timer count at the valid edge of the
TI000 pin
;  - Use CR000 as an overflow compare register
```

```
;  - Use CR010 as a capture register
;  - Capture the pulse width by detecting both edges of the TI000 pin
;  - Count clock = fxp/4
;  - Not performing timer output
;
;
;  <Calculating the pulse width>
;
;  The length of the data to be captured is 24 bits, which is a combination
;  of the 16-bit data of the CR010 register and higher 8-bit data.
;  They are each saved to the RAM area, as shown in the table below.
;
;  The pulse width is calculated from the 24-bit data, as shown below.
;
;    24-bit data x 0.0005 [ms/clk] = pulse width [ms]
;
;  # Here, 1 [clk] is the timer 00 count clock (2 MHz).
;
;  +-------------------------------------------------------------------+
;  |     Address   | Data Length |             Data Type               |
;  |-------------------------------------------------------------------|
;  | LOWLENGTH     | 16 bits     | Pulse width lower data (1st time)   |
;  | LOWLENGTH + 2 | 16 bits     | Pulse width lower data (2nd time)   |
;  | LOWLENGTH + 4 | 16 bits     | Pulse width lower data (3rd time)   |
;  | LOWLENGTH + 6 | 16 bits     | Pulse width lower data (4th time)   |
;  | LOWLENGTH + 8 | 16 bits     | Pulse width lower data (5th time)   |
;  | LOWLENGTH +10 | 16 bits     | Pulse width lower data (6th time)   |
;  | LOWLENGTH +12 | 16 bits     | Pulse width lower data (7th time)   |
;  | LOWLENGTH +14 | 16 bits     | Pulse width lower data (8th time)   |
;  |-------------------------------------------------------------------|
;  | HIGHLENGTH    |  8 bits     | Pulse width higher data (1st time)  |
;  | HIGHLENGTH +1 |  8 bits     | Pulse width higher data (2nd time)  |
;  | HIGHLENGTH +2 |  8 bits     | Pulse width higher data (3rd time)  |
;  | HIGHLENGTH +3 |  8 bits     | Pulse width higher data (4th time)  |
;  | HIGHLENGTH +4 |  8 bits     | Pulse width higher data (5th time)  |
;  | HIGHLENGTH +5 |  8 bits     | Pulse width higher data (6th time)  |
;  | HIGHLENGTH +6 |  8 bits     | Pulse width higher data (7th time)  |
;  | HIGHLENGTH +7 |  8 bits     | Pulse width higher data (8th time)  |
;  +-------------------------------------------------------------------+
;
;
;<<I/O port settings>>
;
;  Input: P30
;  Output: P00-P03, P20-P23, P31-P33, P40-P47, P120-P123, P130
;  # All unused ports are set as the output mode.
;
;************************************************************************
```

```
  ;=============================================================================
  ;
  ;   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    RESET_START       ;(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 RAM
  ;
  ;=============================================================================
  XRAMDSEG  SADDRP
  LOWLENGTH:DS    16          ;  Table  for  storing  the  pulse  width  lower  16
bits
  HIGHLENGTH:     DS    8     ;  Table  for  storing  the  pulse  width  higher  8
bits
  HIGHCOUNT:DS    1           ; For counting the pulse width higher digits
  TIMES:          DS    1     ; For counting the number of measurements

  ;=============================================================================
  ;
  ;   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/clk] x 10[clk] 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,  #11110001B  ; Set P31-P33 as output mode, P30/TI000 as input
mode

  ;---------------------------------------------------------------------------
  ;   Initialize the port 4
  ;---------------------------------------------------------------------------
      MOV   P4,   #00000000B  ; Set output latches of P40-P47 as low
      MOV   PM4,  #00000000B  ; Set P40-P47 as output 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,      #00000100B  ; Use CR010 as a capture register
      MOVW  AX,   #0FFFFH
      MOVW  CR000,      AX          ; Use CR000 for overflow detection
      MOV   PRM00,      #00110001B  ; Specify the valid edge of the TI000 pin
to both edges, set the count clock to fxp/4
      MOV   TOC00,      #00000000B  ; Do not perform timer output

  ;-------------------------------------------------------------------------
  ;   Set the interrupt
  ;-------------------------------------------------------------------------
      MOV   MK0,  #0FFH       ; Mask all interrupts at first
      MOV   IF0,  #00H        ; Clear invalid interrupt requests in advance

  ;*************************************************************************
  ;
  ;   Main loop
  ;
  ;*************************************************************************
      MOVW  HL,   #LOWLENGTH  ;  Initialize  the  address  of  the  table  for
storing the lower 16 bits
      MOVW  DE,   #HIGHLENGTH ;  Initialize  the  address  of  the  table  for
storing the higher 8 bits
      MOV   HIGHCOUNT, #0     ; Clear the higher counter
      MOV   TIMES,      #8    ; Initialize the stored number of results

      MOV   TMC00,      #00001000B  ; Start the timer operation (clear & start
by the valid edge of the TI000 pin)
  WAITCAP:
      NOP
      BF    TMIF010, $WAITCAP ; Wait for completion of the first capture

      MOV   IF0,  #00H        ; Clear the interrupt request flag
      CLR1  TMMK000           ; Unmask INTTM000 interrupts
      CLR1  TMMK010           ; Unmask INTTM010 interrupts

      EI                      ; Enable vector interrupt

  MAIN_LOOP:
      NOP
      BR    $MAIN_LOOP        ; Go to the MAIN_LOOP

  ;*************************************************************************
  ;
  ;   Interrupt INTTM000
  ;
```

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

    BF    TMIF010, $INCHIGH ; Branch if not generated simultaneously with
the INTTM000 interrupt
    MOVW  AX,   CR010       ; Read the captured value
    CMPW  AX,   #0FFFFH
    BZ    $ENDINTTM000      ; Do not count if interrupts are generated
simultaneously and if CR010 == 0xFFFF

INCHIGH:
    INC   HIGHCOUNT         ; Increment the higher digit count by 1

ENDINTTM000:
    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

    MOVW  AX,   CR010       ; Read the captured value
    MOV   [HL+1],     A     ; Save to the table for storing the lower 16
bits
    XCH   A,    X
    MOV   [HL], A           ; Save to the table for storing the lower 16
bits
    MOV   A,    HIGHCOUNT   ; Read the higher digits
    MOV   [DE], A           ; Save to the table for storing the higher 8
bits

    INC   L                 ; Increment by 2 the address of the table for
storing the lower 16 bits
    INC   L
    INC   E                 ; Increment by 1 the address of the table for
storing the higher 8 bits
    MOV   HIGHCOUNT, #0     ; Clear the higher digits

    DBNZ  TIMES,      $ENDINTTM000      ;   Branch   if   the   number   of
measurements is smaller than 8
    SET1  TMMK000           ; Mask INTTM000 interrupts
    SET1  TMMK010           ; Mask INTTM010 interrupts
```

```
ENDINTTM010:
    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 (pulse width measurement)
************************************************************************
<<History>>
      2007.7.--   Release
************************************************************************

<<Overview>>

This sample program presents an example of using the pulse width measurement
function of 16-bit timer 00.  The timer count value when the count clock is
set to fxp/4 (= 2 MHz) and both the rising and falling edges of the signal
input to the TI000 pin are detected is captured to the CRC010 register.
The width or interval of the pulse input to the TI000 pin can be known from
the captured value.  The width or interval of a pulse whose length is at
least the timer cycle can be measured by setting the CR000 register to
0xFFFF and generating an overflow interrupt.


  <Principal setting contents>

  - 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


  <16-bit timer 00 settings>

  - Operation mode: Clear & start the timer count at the valid edge of the
TI000 pin
  - Use CR000 as an overflow compare register
  - Use CR010 as a capture register
  - Capture the pulse width by detecting both edges of the TI000 pin
  - Count clock = fxp/4
  - Not performing timer output


  <Calculating the pulse width>

  The length of the data to be captured is 24 bits, which is a combination
  of the 16-bit data of the CR010 register and higher 8-bit data.
  They are each saved to the RAM area, as shown in the table below.

  The pulse width is calculated from the 24-bit data, as shown below.

    24-bit data x 0.0005 [ms/clk] = pulse width [ms]
```

```
  # Here, 1 [clk] is the timer 00 count clock (2 MHz).
```

Variable Name	Data Length	Data Type
g_unLowLength[0]	16 bits	Pulse width lower data (1st time)
g_unLowLength[1]	16 bits	Pulse width lower data (2nd time)
g_unLowLength[2]	16 bits	Pulse width lower data (3rd time)
g_unLowLength[3]	16 bits	Pulse width lower data (4th time)
g_unLowLength[4]	16 bits	Pulse width lower data (5th time)
g_unLowLength[5]	16 bits	Pulse width lower data (6th time)
g_unLowLength[6]	16 bits	Pulse width lower data (7th time)
g_unLowLength[7]	16 bits	Pulse width lower data (8th time)
g_unHighLength[0]	8 bits	Pulse width higher data (1st time)
g_unHighLength[1]	8 bits	Pulse width higher data (2nd time)
g_unHighLength[2]	8 bits	Pulse width higher data (3rd time)
g_unHighLength[3]	8 bits	Pulse width higher data (4th time)
g_unHighLength[4]	8 bits	Pulse width higher data (5th time)
g_unHighLength[5]	8 bits	Pulse width higher data (6th time)
g_unHighLength[6]	8 bits	Pulse width higher data (7th time)
g_unHighLength[7]	8 bits	Pulse width higher data (8th time)

```
<<I/O port settings>>

  Input: P30
  Output: P00-P03, P20-P23, P31-P33, P40-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     NOP                     /* NOP instructions can be described at
the C source level */
#pragma interrupt INTTM000 fn_inttm000    /* Interrupt function
declaration:INTTM000 */
#pragma interrupt INTTM010 fn_inttm010    /* Interrupt function
declaration:INTTM010 */

/*=======================================================================

      Define the global variables

=======================================================================*/
sreg static unsigned int g_unLowLength[8];      /* 16-bit variable for storing
the pulse width lower 16 bits */
```

```
sreg static unsigned char g_unHighLength[8];    /* 8-bit variable for storing
the pulse width higher digits */
sreg static unsigned char g_ucHighCount;  /* 8-bit variable for counting the
pulse width higher digits */
sreg static unsigned char g_ucTimes;      /* 8-bit variable for counting the
number of measurements */

/************************************************************************

      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   = 0b11110001;           /* Set P31-P33 as output mode, P30/TI000
as input mode */

/*-------------------------------------------------------------------------
      Initialize the port 4
-------------------------------------------------------------------------*/
      P4    = 0b00000000;           /* Set output latches of P40-P47 as low */
      PM4   = 0b00000000;           /* Set P40-P47 as output 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 = 0b00000100;           /* Use CR010 as a capture register */
      CR000 = 0xFFFF;               /* Use CR000 for overflow detection */
      PRM00 = 0b00110001;           /* Specify the valid edge of the TI000 pin
to both edges, set the count clock to fxp/4 */
      TOC00 = 0b00000000;           /* Do not perform timer output */

/*-------------------------------------------------------------------------
      Set the interrupt
-------------------------------------------------------------------------*/
      MK0   = 0xFF;                 /* Mask all interrupts at first */
      IF0   = 0x00;                 /* Clear invalid interrupt requests */

      return;
}

/*************************************************************************

      Main loop

*************************************************************************/
void main(void){
      g_ucHighCount = 0;            /* Clear the higher-digit counter */
```

```
      g_ucTimes = 8;                /* Initialize the number of measurements
*/
      TMC00 = 0b00001000;           /* Start the timer operation (clear &
start by the valid edge of the TI000 pin) */

      while(TMIF010==0){            /* Wait for completion of the first
capture */
            NOP();
      }

      IF0 = 0x00;                   /* Clear the interrupt request flag */
      TMMK000 = 0;                  /* Unmask INTTM000 interrupts */
      TMMK010 = 0;                  /* Unmask INTTM010 interrupts */

      EI();                         /* Enable vector interrupt */

      while(1){                     /* Infinite loop */
            NOP();
            NOP();
      }
}

/************************************************************************

      Interrupt INTTM000

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

      if (!(TMIF010 && (CR010 == 0xFFFF))){     /* Do not count if interrupts
are generated simultaneously and if CR010 == 0xFFFF */
            g_ucHighCount++;  /* Increment the higher digit count by 1 */
      }

      return;
}

/************************************************************************

      Interrupt INTTM010

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

      g_unLowLength[8 - g_ucTimes] = CR010;     /* Read the captured value */
      g_unHighLength[8 - g_ucTimes] = g_ucHighCount;  /* Read the higher
digits */
      g_ucHighCount = 0;            /* Clear the higher digits */
      g_ucTimes--;                  /* Decrement the number of measurements by
1 */

      if(g_ucTimes == 0){     /* When the eighth capture is completed */
            TMMK000 = 1;            /* Mask INTTM000 interrupts */
            TMMK010 = 1;            /* Mask INTTM010 interrupts */
      }

      return;
}
```

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

```
;=============================================================================
;
;     Option byte
;
;=============================================================================
OPBT       CSEG     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	November 2007	–	–
2nd edition	September 2008	pp.33 to 35	Modification of 5.1 Building the Sample Program
		p.38	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/

G0706