

## Application Note

### 78K0S/Kx1+

#### Sample Program (8-bit Timer H1)

#### Interval Timer

This document describes an operation overview of the sample program and how to use it, as well as how to set and use the interval timer function of 8-bit timer H1. In the sample program, the LEDs are blinked at fixed cycles by using the interval timer function of 8-bit timer H1. Furthermore, the blinking cycle of the LEDs is changed 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.....	4
<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 Interval Timer Function of 8-bit Timer H1.....	10
4.2 Setting the LED Blinking Cycle and Chattering Detection Time.....	17
<b>CHAPTER 5 OPERATION CHECK USING SYSTEM SIMULATOR SM+</b> ....	<b>21</b>
5.1 Building the Sample Program .....	21
5.2 Operation with SM+.....	22
<b>CHAPTER 6 RELATED DOCUMENTS</b> .....	<b>27</b>
<b>APPENDIX A PROGRAM LIST</b> .....	<b>28</b>
<b>APPENDIX B REVISION HISTORY</b> .....	<b>41</b>

- **The information in this document is current as of July, 2007. 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 interval timer function of 8-bit timer H1 is presented in this sample program. The LEDs are blinked at fixed cycles and the blinking cycle of the LEDs is changed 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.

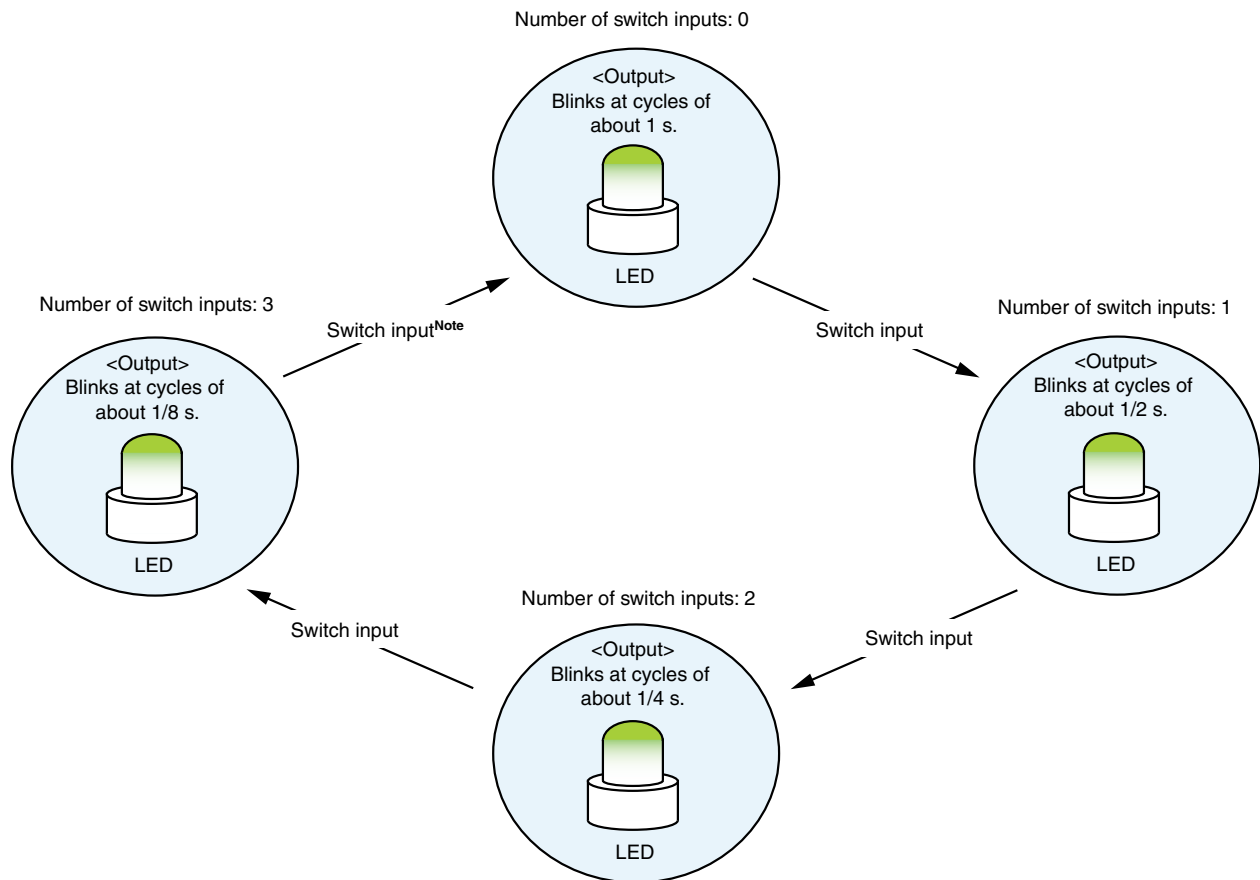
- Selecting the high-speed internal oscillator as the system clock source<sup>Note</sup>
- Stopping watchdog timer operation
- Setting  $V_{LVI}$  (low-voltage detection voltage) to 4.3 V  $\pm$ 0.2 V
- Generating an internal reset (LVI reset) signal when it is detected that  $V_{DD}$  is less than  $V_{LVI}$ , 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 8-bit timer H1
  - Setting the count clock to  $f_{XP}/2^6$  (125 kHz), setting the operation mode to the interval timer mode, and disabling the timer output from TOH1
  - Setting the interval cycle to 2 ms ( $8 \mu\text{s} \times 250$ )
- Setting the valid edge of INTP1 (external interrupt) to the falling edge
- Enabling INTP1 and INTTMH1 interrupts

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

## 1.2 Contents Following the Main Loop

The LEDs are blinked at fixed cycles by using the generation of an 8-bit timer H1 interrupt (INTTMH1), 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 blinking cycle of the LEDs is changed 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.



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

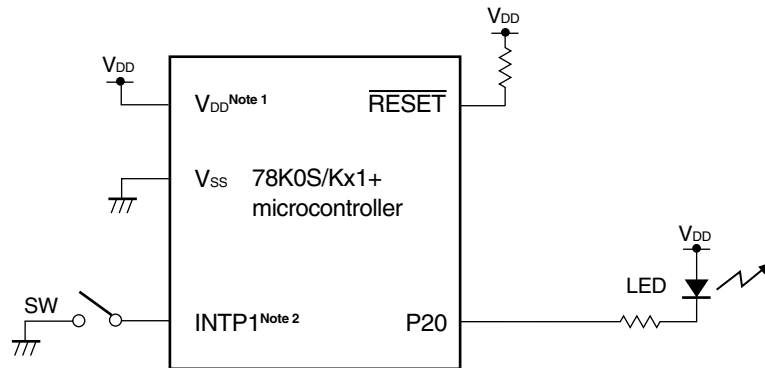
Chattering is a phenomenon in which the electric signal repeats turning on and off due to a mechanical flip-flop of the contacts, immediately after the switch has been pressed.

## 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/P43: 78K0S/KA1+ and 78K0S/KB1+ microcontrollers  
INTP1/P32: 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.

### 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 lighting of an LED.

#### (2) LED




An LED is used as an output corresponding to the interval timer function of 8-bit timer H1 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)	●	●	
tmh1.prw	Work space file for integrated development environment PM+		●	
tmh1.prj	Project file for integrated development environment PM+		●	
tmh1.pri tmh1.prs tmh1.prm	Project files for system simulator SM+ for 78K0S/Kx1+		● Note 2	
tmh10.pnl	I/O panel file for system simulator SM+ for 78K0S/Kx1+ (used for checking peripheral hardware operations)		● Note 2	●
tmh10.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.

- Interval timer function: 8-bit timer H1
- $V_{DD} < V_{LVI}$  detection: Low-voltage detector (LVI)
- Switch input: INTP1<sup>Note</sup> (external interrupt)
- LED output: P20 (output port)

**Note** INTP1/P43: 78K0S/KA1+ and 78K0S/KB1+ microcontrollers  
INTP1/P32: 78K0S/KY1+ and 78K0S/KU1+ microcontrollers

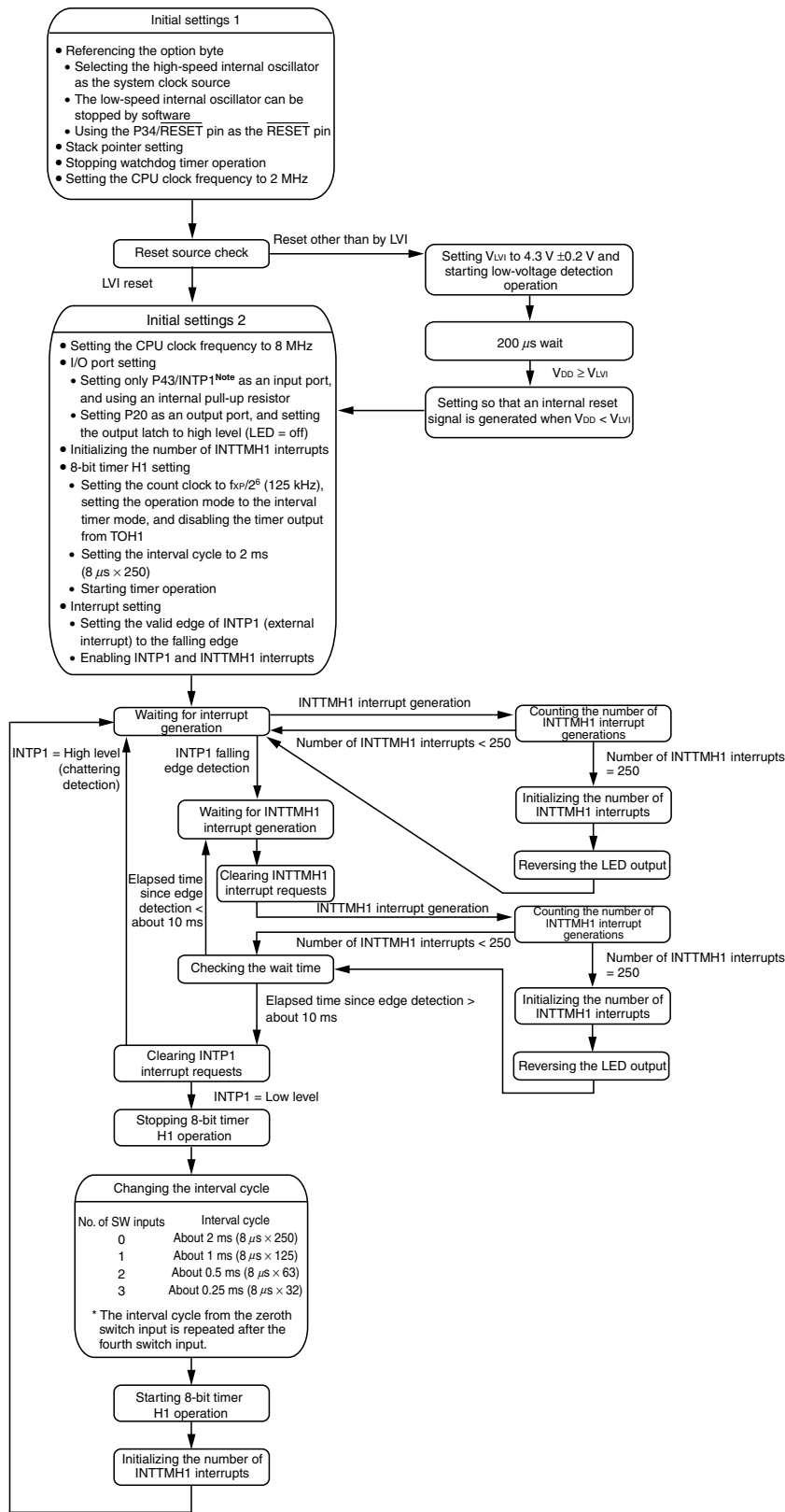
### 3.3 Initial Settings and Operation Overview

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 8-bit timer H1 (interval timer), and setting of interrupts are performed.

The LEDs are blinked at fixed cycles by using the generation of an 8-bit timer H1 interrupt (INTTMH1), 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 blinking cycle of the LEDs is changed 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.

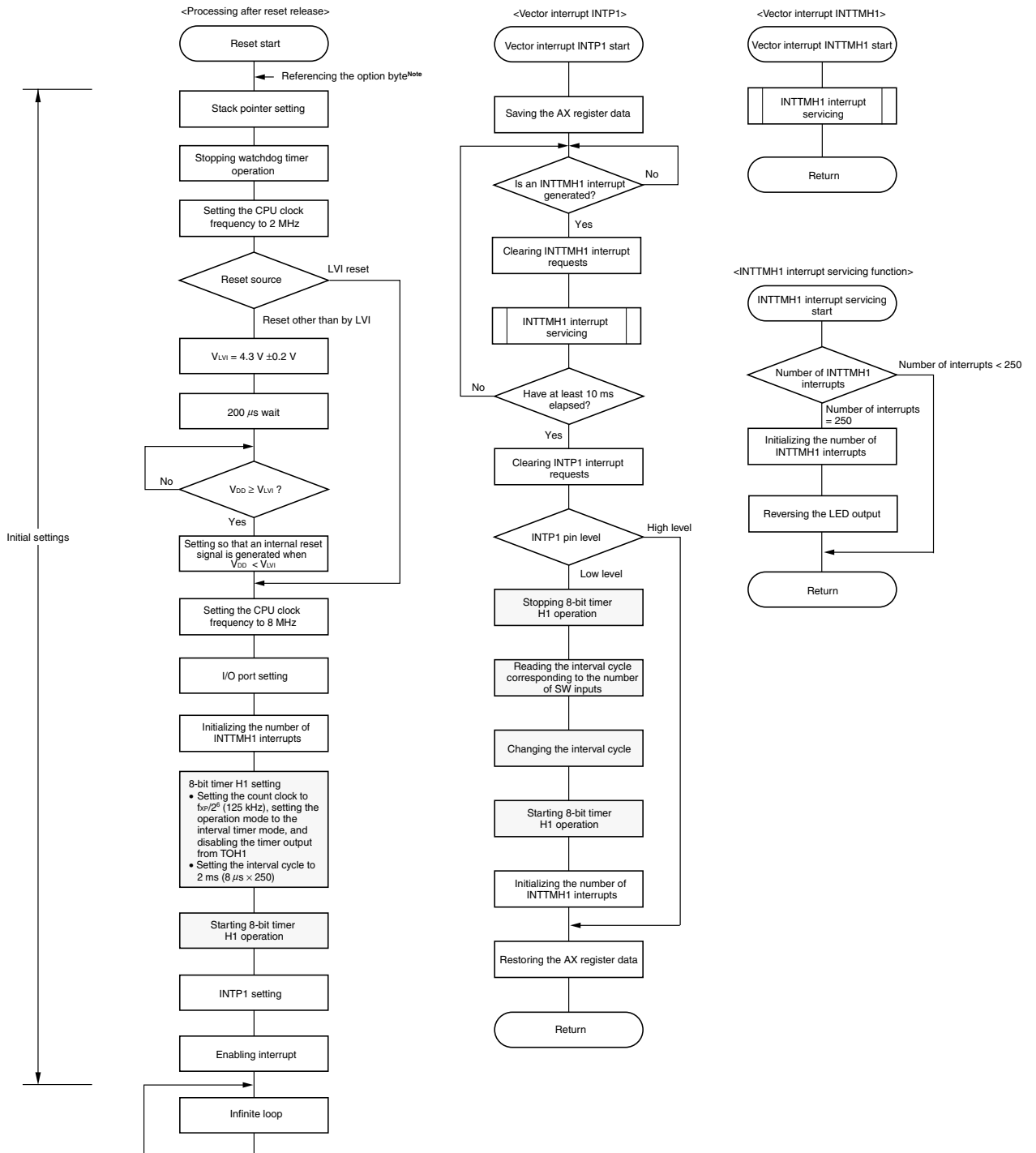


**Note** INTP1/P43: 78K0S/KA1+ and 78K0S/KB1+ microcontrollers  
 INTP1/P32: 78K0S/KY1+ and 78K0S/KU1+ microcontrollers



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 interval timer function of 8-bit timer H1.

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 Interval Timer Function of 8-bit Timer H1

The following five types of registers are set when using 8-bit timer H1.

- 8-bit timer H mode register 1 (TMHMD1)
- 8-bit timer H compare register 01 (CMP01)
- Port mode register x (PMx)<sup>Note</sup>
- Port register x (Px)<sup>Note</sup>
- Port mode control register x (PMCx)<sup>Note</sup>

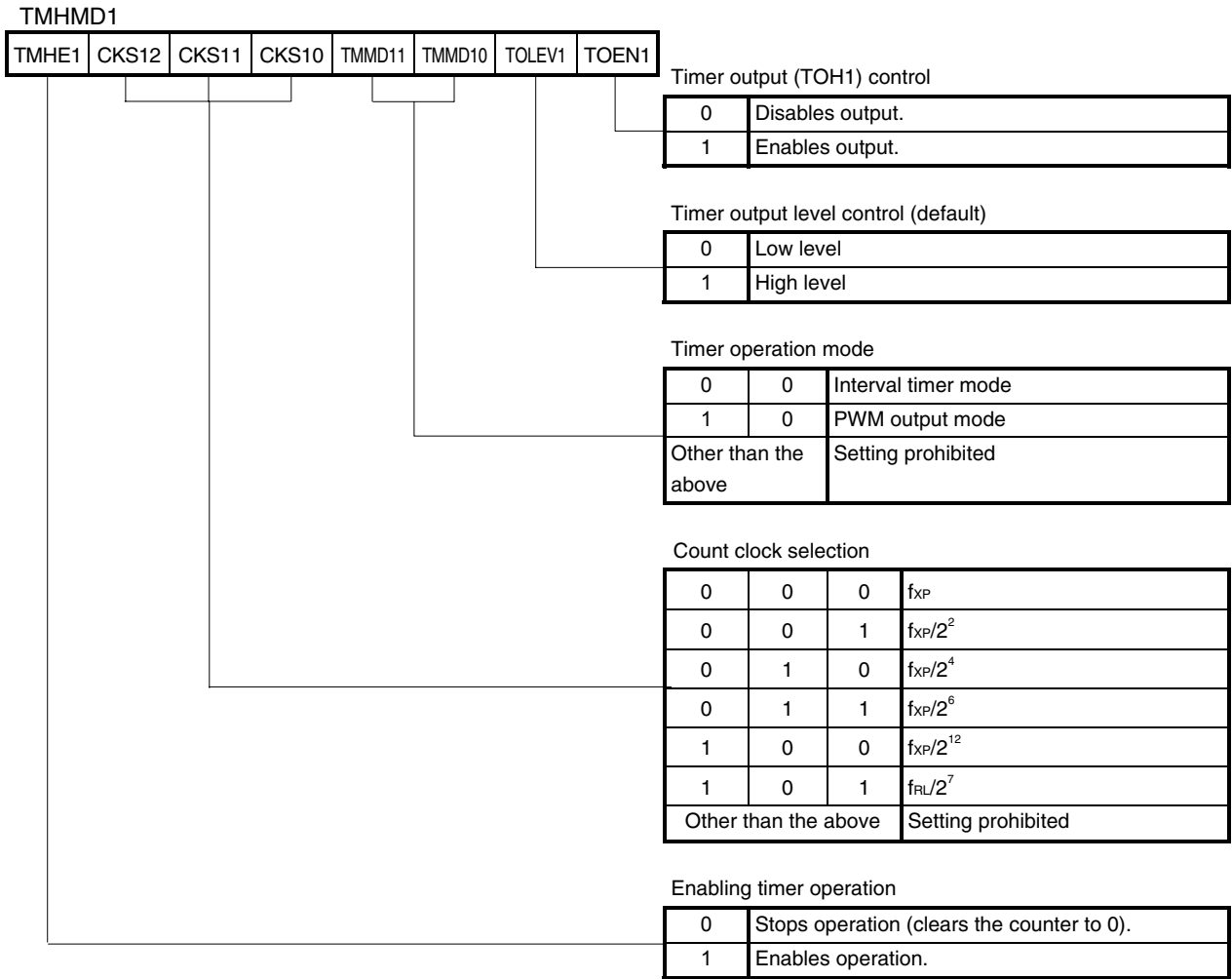
**Note** To use the TOH1 pin for timer output, set it as follows.

	Px Register	PMx Register	PMCx Register
78K0S/KA1+ and 78K0S/KB1+ microcontrollers	P42 = 0	PM42 = 0	Setting not required
78K0S/KY1+ and 78K0S/KU1+ microcontrollers	P20 = 0	PM20 = 0	PMC20 = 0

#### (1) Setting regarding the operation mode of 8-bit timer H1

The operation mode is set, the count clock is selected, and operation is controlled for 8-bit timer H1 by using 8-bit timer H mode register 1 (TMHMD1).

Figure 4-1. Format of 8-bit Timer H Mode Register 1 (TMHMD1)



**Caution** Setting the other bits of the TMHMD1 register is prohibited when TMHE1 is set to 1.

**Remark**  $f_{XP}$ : Oscillation frequency of the clock supplied to peripheral hardware  
 $f_{RL}$ : Internal low-speed oscillation clock frequency

**(2) Interval time setting**

The interval time is set by using 8-bit timer H compare register 01 (CMP01).

• Interval time =  $(N + 1)/f_{CNT}$

**Remark** N: CMP01 setting value (00H to FFH)  
 $f_{CNT}$ : Count clock frequency of 8-bit timer H1

Figure 4-2. Format of 8-bit Timer H Compare Register 01 (CMP01)



**Caution** Rewriting the CMP01 register value during timer count operation is prohibited.

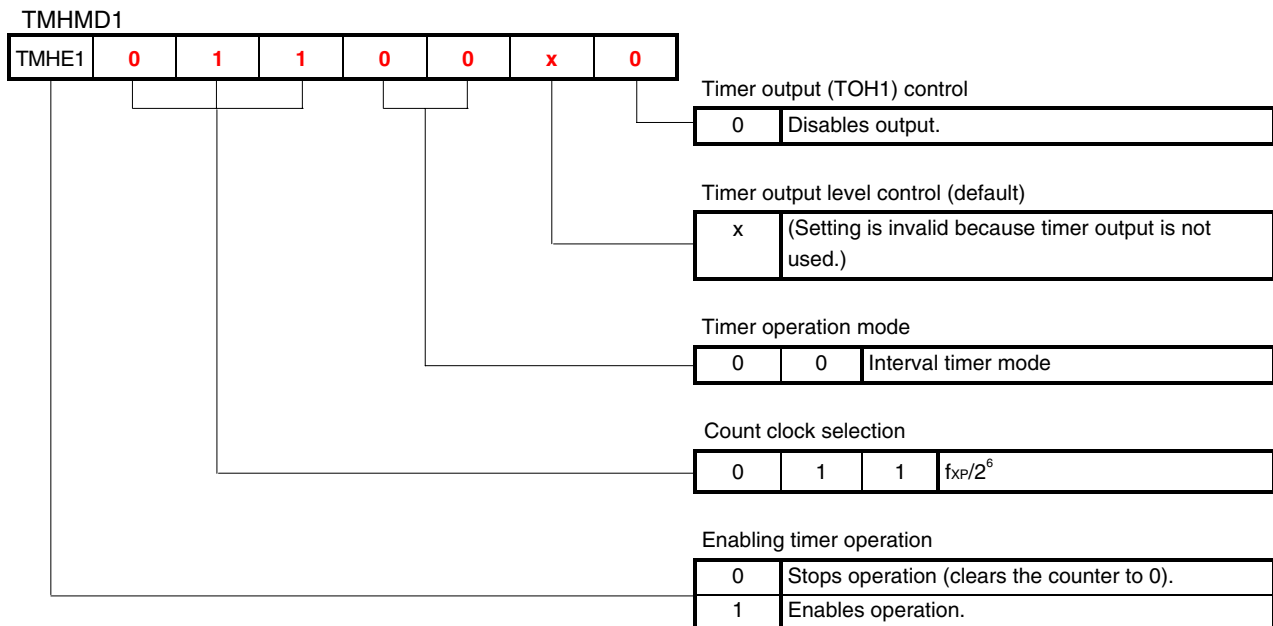
**(3) TOH1 pin setting**

When timer output is enabled in the interval timer mode, the output level of the TOH1 pin is reversed upon a match between the values of 8-bit timer counter H1 and the CMP01 register.

To use the TOH1 pin for timer output, set port register x (Px), port mode register x (PMx), and port mode control register x (PMCx) as follows.

	Px Register	PMx Register	PMCx Register
78K0S/KA1+ and 78K0S/KB1+ microcontrollers	P42 = 0	PM42 = 0	Setting not required
78K0S/KY1+ and 78K0S/KU1+ microcontrollers	P20 = 0	PM20 = 0	PMC20 = 0

- [Example 1]**
- Setting the operation mode of 8-bit timer H1 to interval timer mode, setting the count clock to  $f_{XP}/2^6$  ( $f_{XP} = 8 \text{ MHz}$ ), and disabling timer output (TOH1)
  - Setting the interval cycle to 2 ms, and starting timer operation  
(Same content as in the sample program)



CMP01 setting value (N): 249

- Count clock  $f_{CNT} = 8 \text{ MHz}/2^6 = 0.125 \text{ MHz} = 125 \text{ kHz}$
- Interval cycle 2 ms =  $(N + 1)/125 \text{ kHz}$   
→  $N = 2 \text{ ms} \times 125 \text{ kHz} - 1 = 249$

Timer operation is started by setting 1 to TMHE1 after setting “001100x0” (x: don’t care)” (“x” is set to 0 in the example shown below) to TMHMD1 and “249” to CMP01.

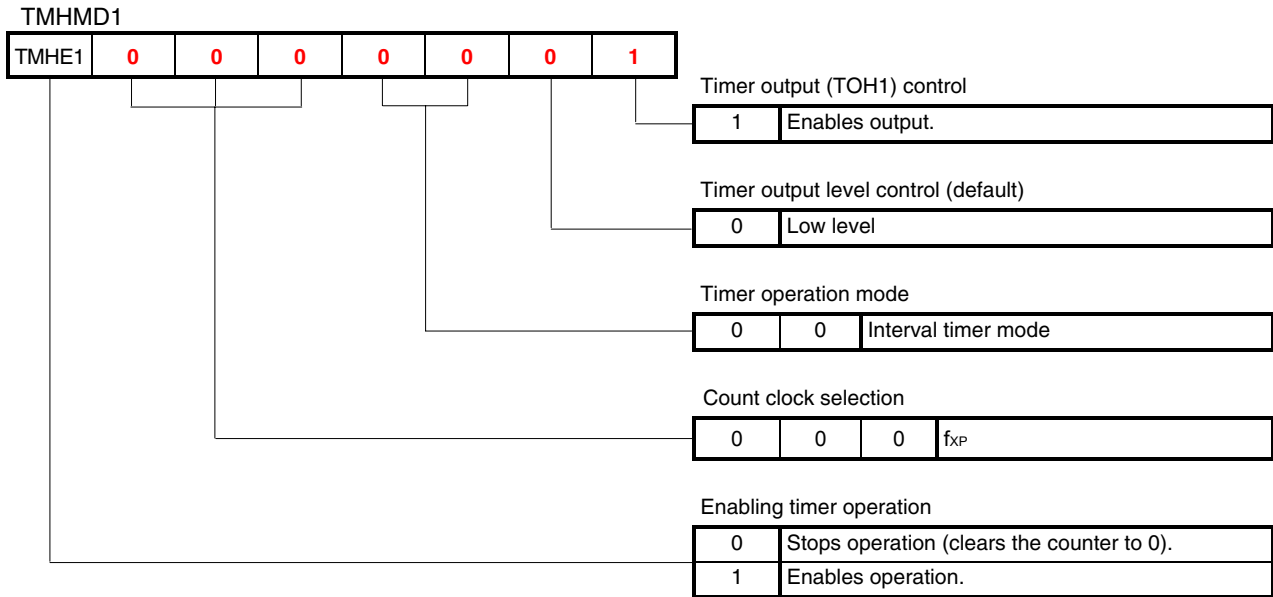
- Assembly language

```
MOV    TMHMD1, #00110000B
MOV    CMP01,  #249
SET1   TMHE1
```

- C language

```
TMHMD1 = 0b00110000;
CMP01  = 249;
TMHE1  = 1;
```

- [Example 2]**
- Setting the operation mode of 8-bit timer H1 to interval timer mode, setting the count clock to  $f_{XP}$  ( $f_{XP} = 8 \text{ MHz}$ ), enabling timer output (TOH1), and setting the timer output level (default) to low level
  - Setting the interval cycle to  $31.25 \mu\text{s}$ , and starting timer operation



CMP01 setting value (N): 249

- Count clock  $f_{CNT} = 8 \text{ MHz}$
  - Interval cycle  $31.25 \mu\text{s} = (N + 1)/8 \text{ MHz}$
- $N = 31.25 \mu\text{s} \times 8 \text{ MHz} - 1 = 249$

TOH1 pin setting

- 78K0S/KA1+ and 78K0S/KB1+ microcontrollers: P42 = 0, PM42 = 0
- 78K0S/KY1+ and 78K0S/KU1+ microcontrollers: P20 = 0, PM20 = 0, PMC20 = 0

In the case of the 78K0S/KA1+ and 78K0S/KB1+ microcontrollers, timer operation is started by setting 1 to TMHE1 after setting “0” to P42, “0” to PM42, “0000001” to TMHMD1, and “249” to CMP01.

In the case of the 78K0S/KY1+ and 78K0S/KU1+ microcontrollers, timer operation is started by setting 1 to TMHE1 after setting “0” to P20, “0” to PM20, “0” to PMC20, “0000001” to TMHMD1, and “249” to CMP01.

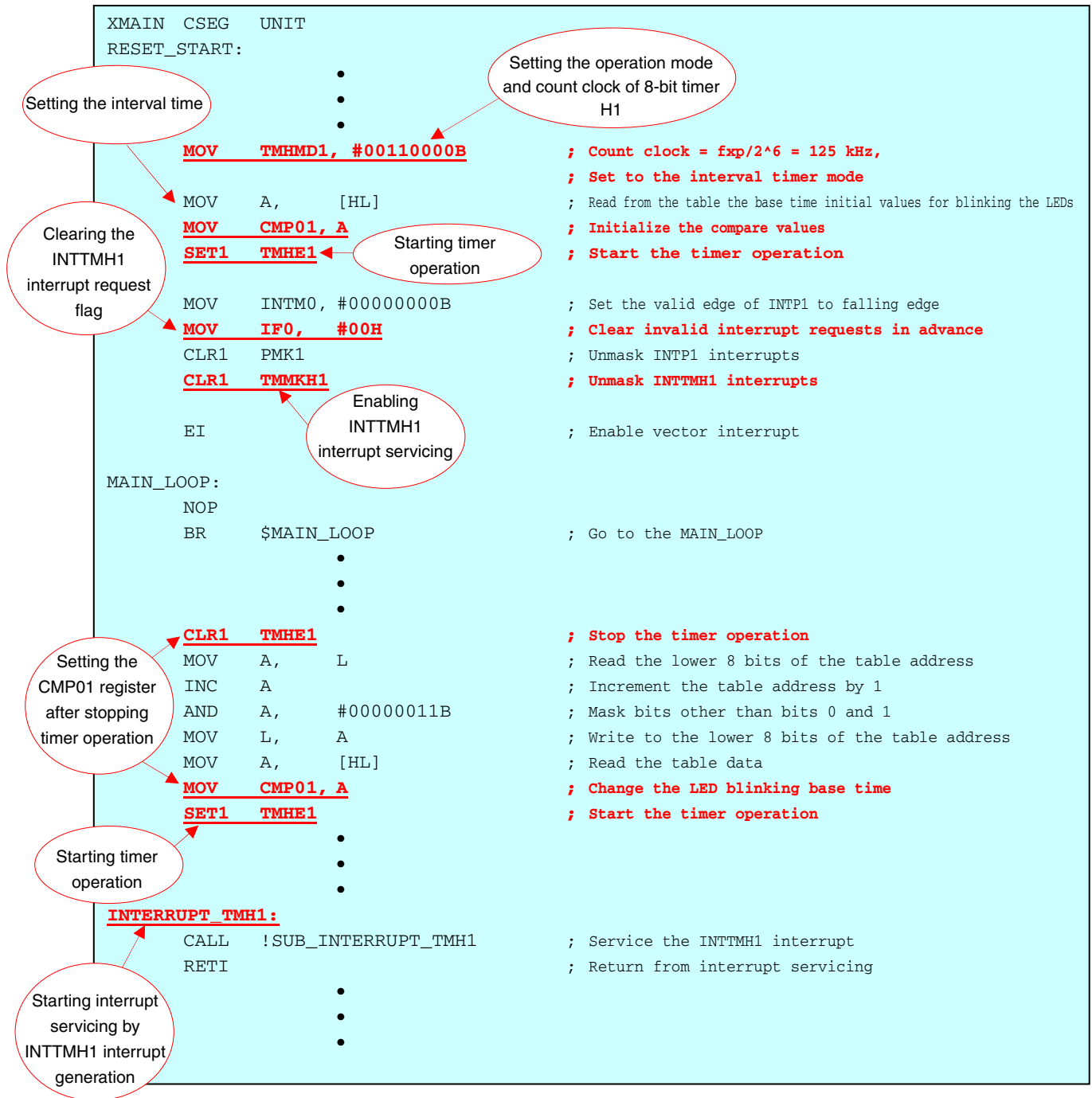
- Assembly language (when using the 78K0S/KA1+ and 78K0S/KB1+ microcontrollers)

```
CLR1  P4.2
CLR1  PM4.2
MOV   TMHMD1, #00000001B
MOV   CMP01,  #249
SET1  TMHE1
```

- C language (when using the 78K0S/KA1+ and 78K0S/KB1+ microcontrollers)

```
P4.2 = 0;
PM4.2 = 0;
TMHMD1 = 0b00000001;
CMP01 = 249;
TMHE1 = 1;
```

- Assembly language program example (same contents as in [Example 1](#)] mentioned above and the sample program)



- C language program example (same contents as in [Example 1] mentioned above and the sample program)

```

void hdwinit(void){
    unsigned char ucCnt200us; /* 8-bit variable for 200 us wait */
        .
        .
        .
    TMHMD1 = 0b00110000; /* Count clock = fxp/26 = 125 kHz, */
    CMP01 = 250-1; /* Set to the interval timer mode */
    TMHE1 = 1; /* Initialize the LED blinking base time */
    /* Start the timer operation */
    INTM0 = 0b00000000; /* Set the valid edge of INTP1 to falling edge */
    IF0 = 0x00; /* Clear invalid interrupt requests in advance */
    PMK1 = 0; /* Unmask INTP1 interrupts */
    TMMKH1 = 0; /* Unmask INTTMH1 interrupts */
    return;
}

void main(void){
    EI(); /* Enable vector interrupt */

    while (1){
        NOP();
        NOP();
    }

        .
        .
        .
    TMHE1 = 0; /* Stop the timer operation */
    CMP01 = g_ucCMPdata[g_ucSWcnt]; /* Change the LED blinking base time in accordance
with the number of switch inputs */
    TMHE1 = 1; /* Start the timer operation */
        .
        .
        .
    interrupt void fn_inttmH1(){
        fn_subinttmH1(); /* Service the INTTMH1 interrupt */
    return;
        .
        .
        .
}

```



## 4.2 Setting the LED Blinking Cycle and Chattering Detection Time

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

### (1) Setting the LED blinking cycle

The LED output is reversed every 250 generations of 8-bit timer H1 interrupts (INTTMH1) in this sample program.

- Interrupt cycle (interval time) =  $(N + 1)/f_{CNT}$
- LED output reversal cycle = Interrupt cycle  $\times$  Number of interrupts
- LED blinking cycle = LED output reversal cycle  $\times$  2

**Remark** N: CMP01 register setting value  
 $f_{CNT}$ : Count clock frequency of 8-bit timer H1

Calculation example: The following values result when the CMP01 register setting value is 249 (during operation at  $f_{CNT} = 125$  kHz).

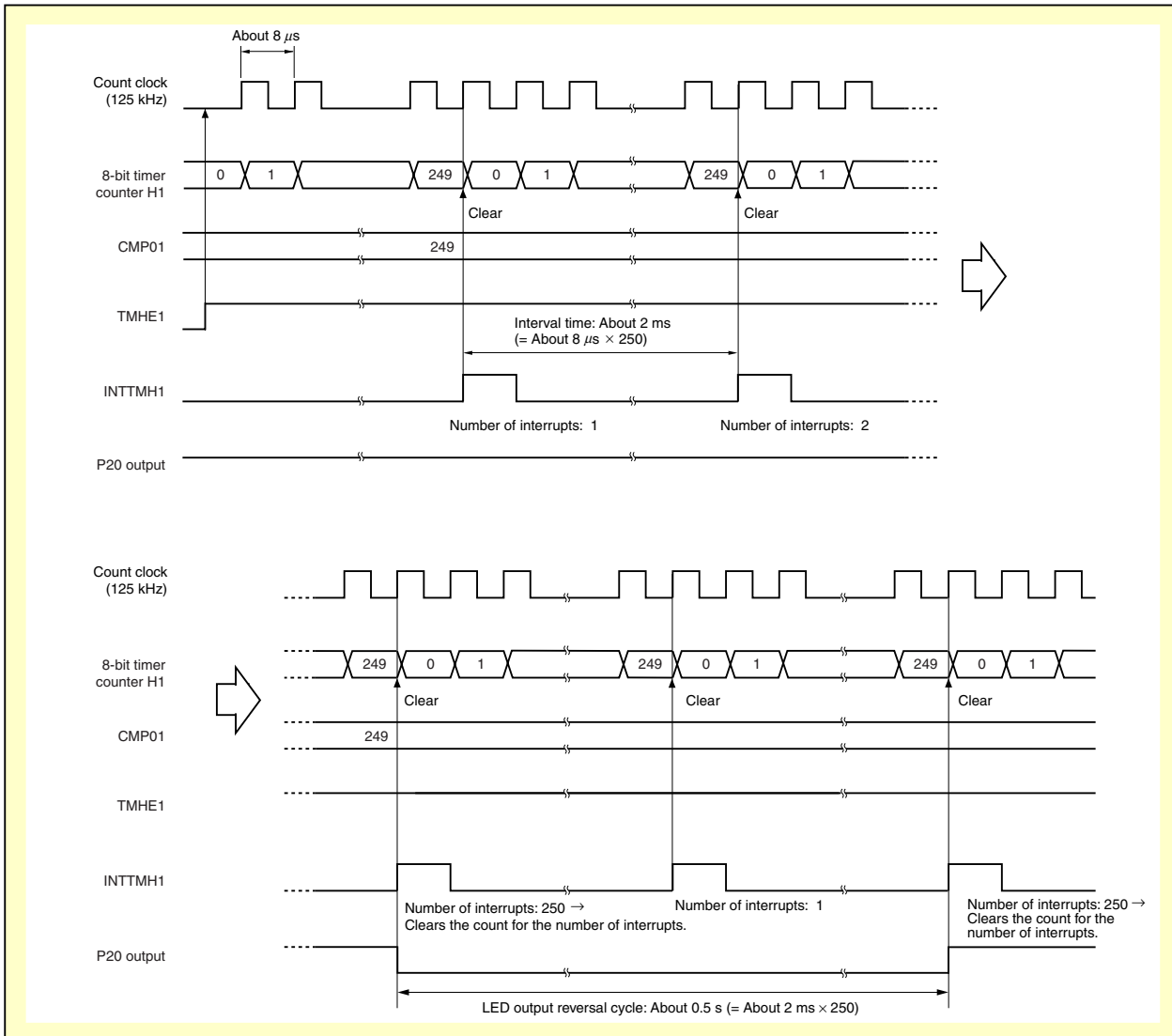
- Interrupt cycle (interval time) =  $(N + 1)/f_{CNT} = (249 + 1)/125$  kHz = 2 ms
- LED output reversal cycle = Interrupt cycle  $\times$  Number of interrupts = 2 ms  $\times$  250 = 500 ms
- LED blinking cycle = LED output reversal cycle  $\times$  2 = 500 ms  $\times$  2 = 1 s

Furthermore, the CMP01 register setting value is changed in accordance with the number of switch inputs, and the LED blinking cycle is changed.

Number of Switch Inputs <sup>Note</sup>	CMP01 Register Setting Value	Interrupt Cycle	LED Blinking Cycle
0	249	About 2 ms ((249 + 1)/125 kHz)	About 1 s (about 2 ms $\times$ 250 $\times$ 2)
1	124	About 1 ms ((124 + 1)/125 kHz)	About 0.5 s (about 1 ms $\times$ 250 $\times$ 2)
2	62	About 0.504 ms ((62 + 1)/125 kHz)	About 0.252 s (about 504 $\mu$ s $\times$ 250 $\times$ 2)
3	31	About 0.256 ms ((31 + 1)/125 kHz)	About 0.128 s (about 256 $\mu$ s $\times$ 250 $\times$ 2)

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

Figure 4-3. Timing Chart Example of the LED Blinking Cycle (When the LEDs Blink at a Cycle of About 1 s)



**Remark** The CMP01 register setting value is 124, 62, and 31 when the LEDs blink at respective cycles of about 1/2 s, 1/4 s, and 1/8 s.

**(2) Setting the chattering detection time**

The generation of 8-bit timer H1 interrupts (INTTMH1) 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.

INTTMH1 interrupts can be continuously counted even during chattering detection by using INTTMH1 interrupts for chattering detection. Consequently, offsets of the LED blinking cycle, which are caused by switch input, can be suppressed.

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

**Remark** T: INTTMH1 interrupt cycle

T': Time from the start of INTP1 edge detection until the first INTTMH1 is generated after INTP1 edge detection ( $0 < T' \leq T$ )

M: Number of INTTMH1 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 interrupt cycle (T) is 2 ms (refer to the calculation example in [\(1\) Setting the LED blinking cycle](#)), and the number of INTTMH1 interrupts after INTP1 edge detection (M) is 6

$$\begin{aligned} T_c &= T' + T \times (M - 1) \\ &= T' + 2 \text{ ms} \times (6 - 1) \\ &= T' + 10 \text{ ms} \end{aligned}$$

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

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

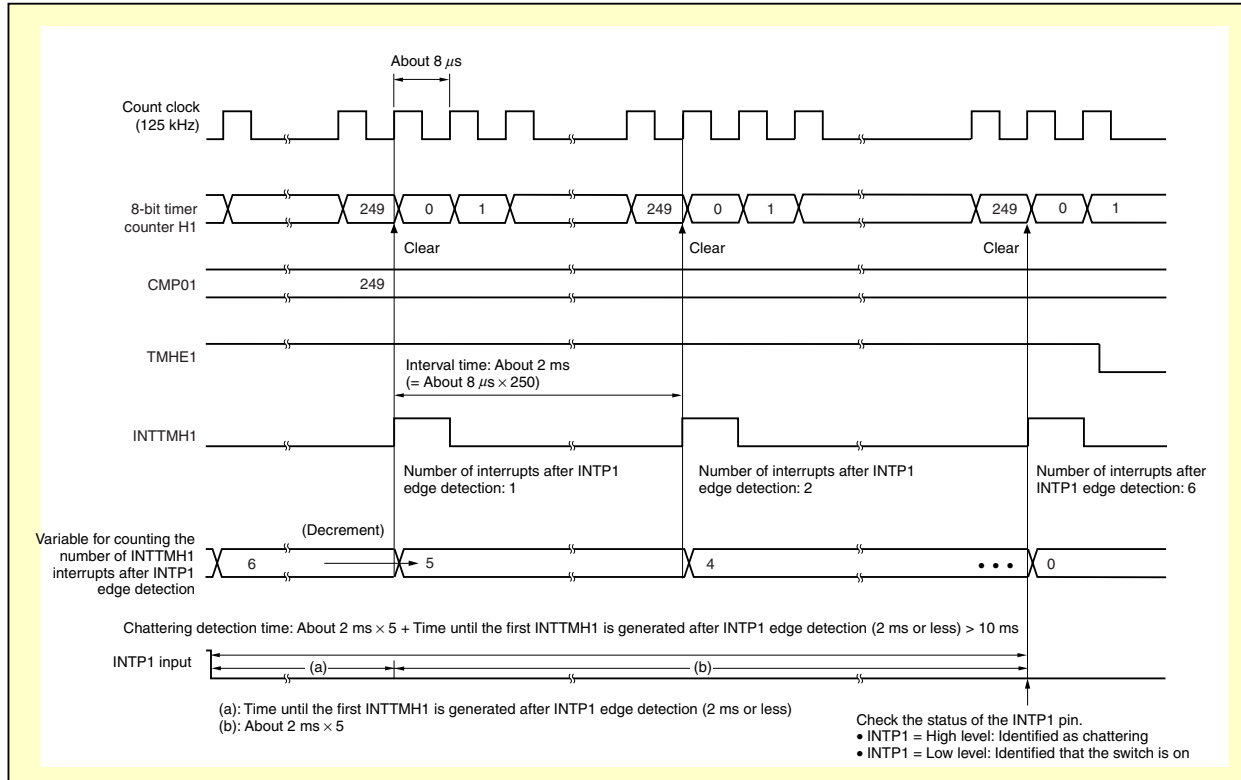


Chattering detection time ( $T_c$ ) > 10 ms

The following table shows the correspondence between the interrupt cycles during switch input and the number of INTTMH1 interrupts after INTP1 edge detection in this sample program.


LED Blinking Cycle	Interrupt Cycle	Number of INTTMH1 Interrupts After INTP1 Edge Detection	Chattering Detection Time
About 1 s	About 2 ms	6	$10 \text{ ms} < T_c \leq 12 \text{ ms}$
About 0.5 s	About 1 ms	11	$10 \text{ ms} < T_c \leq 11 \text{ ms}$
About 0.252 s	About 0.504 ms	21	$10.08 \text{ ms} < T_c \leq 10.584 \text{ ms}$
About 0.128 s	About 0.256 ms	41	$10.24 \text{ ms} < T_c \leq 10.496 \text{ ms}$

**Figure 4-4. Timing Chart Example of Chattering Detection (When the LEDs Blink at Cycles of About 1 s During Switch Input)**




**Remark** The variable for counting the number of INTTMH1 interrupts after INTP1 edge detection depends on the LED blinking cycle during switch input. The variable is 11, 21, and 41, when the LEDs blink at respective cycles of about 1/2 s, 1/4 s, and 1/8 s.

## 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.

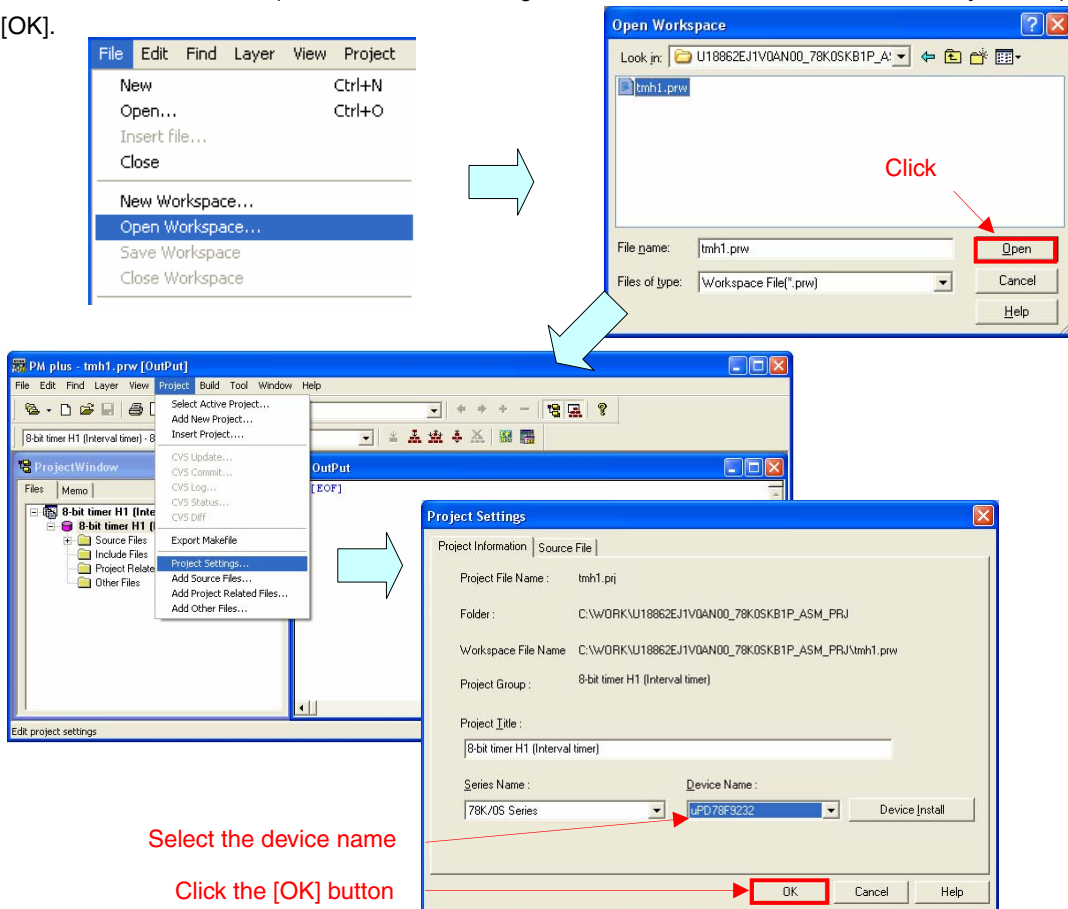
**Caution** System simulator SM+ for 78K0S/Kx1+ is not supported with the 78K0S/KU1+ microcontroller (as of July 2007). The operation of the 78K0S/KU1+ microcontroller, therefore, cannot be checked by using system simulator SM+ for 78K0S/Kx1+.


### 5.1 Building the Sample Program

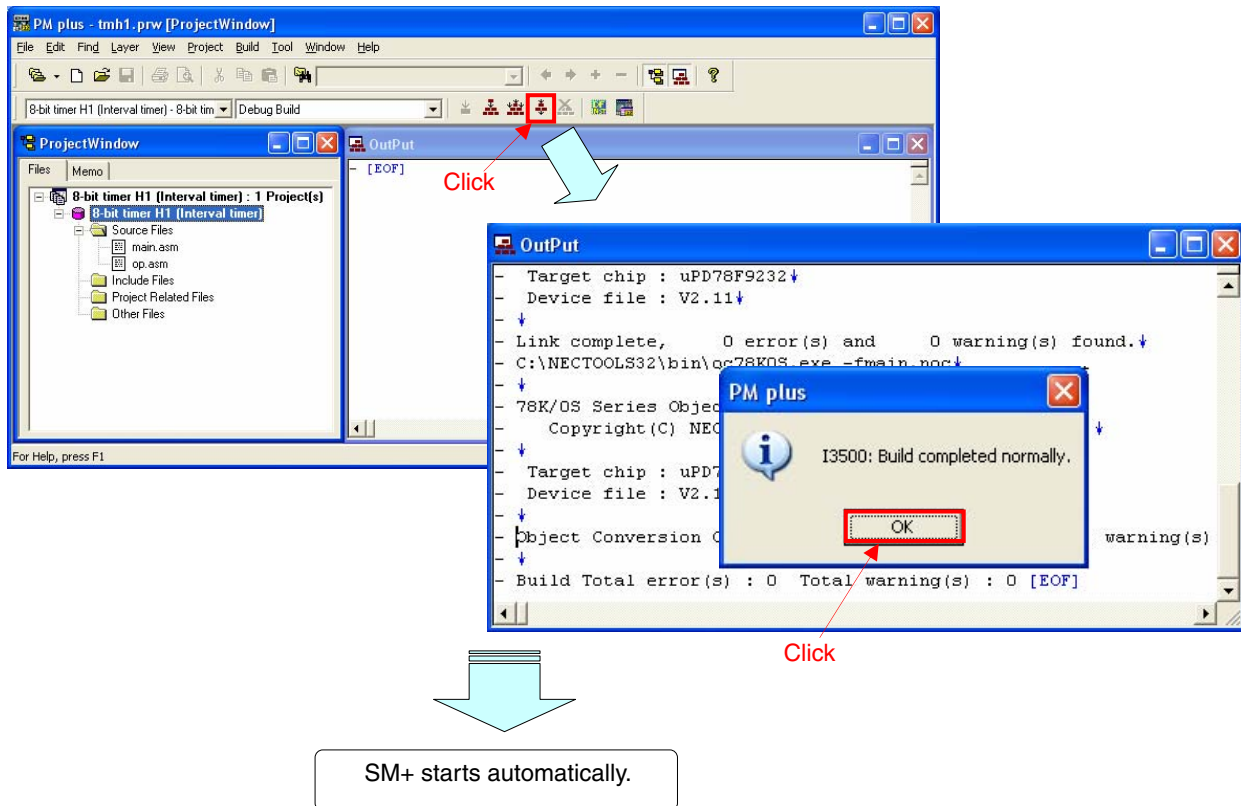
To check the operation of the sample program by using system simulator SM+ for 78K0S/Kx1+ (hereinafter referred to as “SM+”), SM+ must be started after building the sample program. This section describes an example of the operation sequence, from building the sample program with integrated development environment PM+, using the assembly language file (source files + project file) that has been downloaded by selecting , up to starting SM+. For how to build other downloaded programs, refer to **CHAPTER 3 REGISTERING INTEGRATED DEVELOPMENT ENVIRONMENT PM+ PROJECTS AND EXECUTING BUILD** in the [78K0S/Kx1+ Sample Program Startup Guide Application Note](#).

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

- (1) Start PM+.
- (2) Select “tmh1.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].



- (4) Click  ([Build → Debug] button). When the “main.asm” and “op.asm” source files are built normally, the message “I3500: Build completed normally.” will be displayed.
- (5) Click the [OK] button in the message window to automatically start SM+.



## 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](#).




### [Column] Build errors

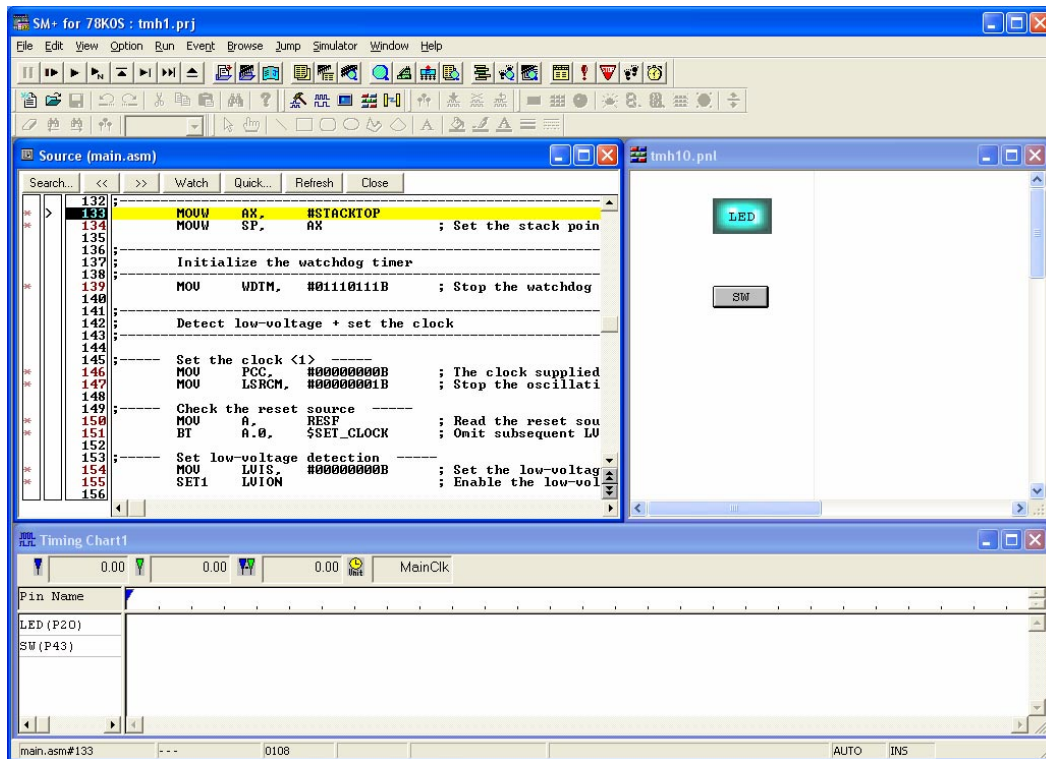
Change the compiler option setting according to the following procedure when the error message “A006 File not found ‘C:\NECTOOLS32\LIB78K0S\s0sl.rel’” or “\*\*\* ERROR F206 Segment ‘@@DATA’ can’t allocate to memory - ignored.” is displayed, when building with PM+.


- <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.)

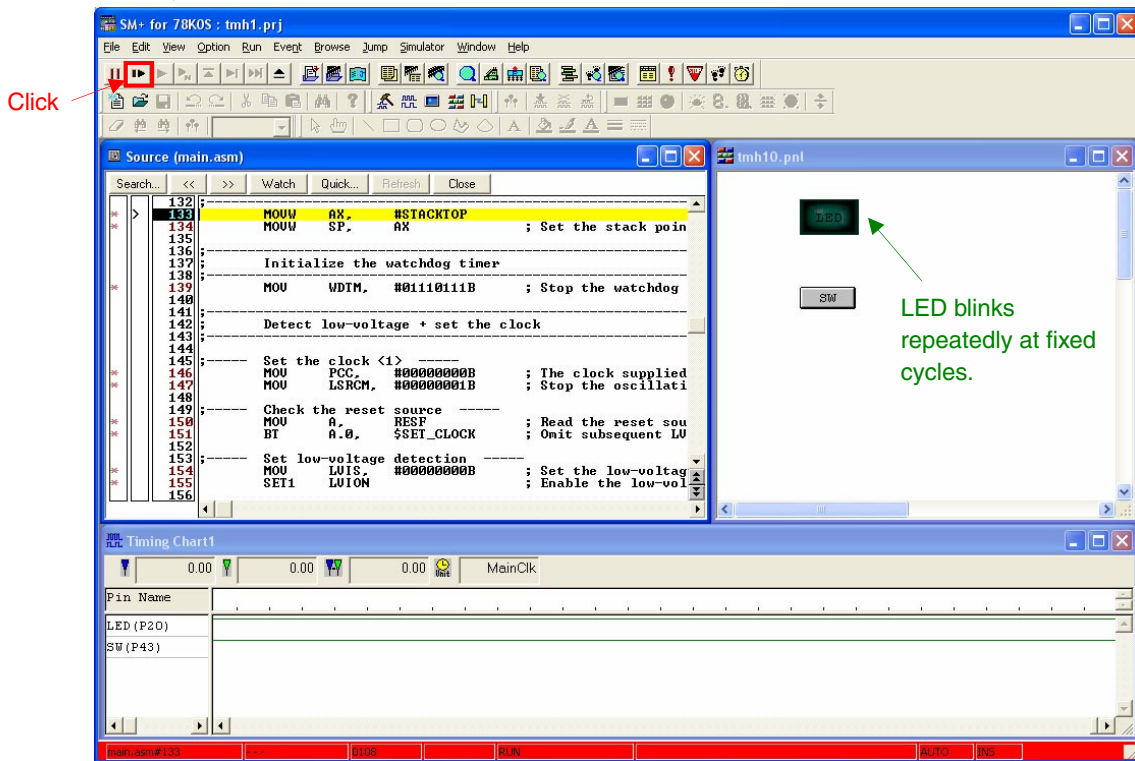
A RAM area of 118 bytes that has been secured as a fixed standard library area will be enabled for use when the [Using Fixed Area of Standard Library] check box is unchecked; however, the standard libraries (such as the getchar function and malloc function) will be disabled for use.

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) When SM+ is started by clicking [Build → Debug] on PM+ (refer to 5.1), the following screen will be displayed. (This is a screen example when the assembly language source file is used.)



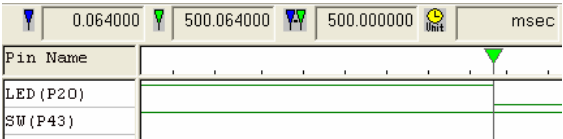


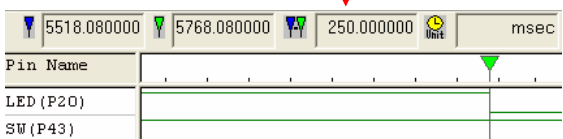


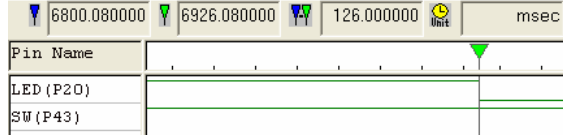


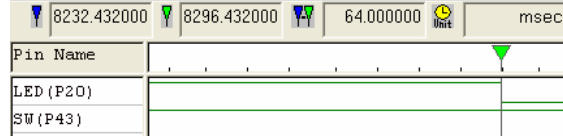


- (2) Click  ([Restart] button). The program will be executed after the CPU is reset and the following screen will be displayed.



This turns red during program execution.

- (3) Click the [SW] button in the I/O panel window, during program execution.  
 Check that the blinking cycle 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>Blinks at cycles of about 1 s<sup>Note 2</sup>.</p>  <p>Do not click.</p> 	<p>The reversal cycle of the LED (P20) is 500 ms.</p> 
<p>Blinks at cycles of about 1/2 s<sup>Note 2</sup>.</p>  <p>Click once.</p> 	<p>The reversal cycle of the LED (P20) is 250 ms.</p> 
<p>Blinks at cycles of about 1/4 s<sup>Note 2</sup>.</p>  <p>Click twice.</p> 	<p>The reversal cycle of the LED (P20) is 126 ms.</p> 
<p>Blinks at cycles of about 1/8 s<sup>Note 2</sup>.</p>  <p>Click three times.</p> 	<p>The reversal cycle of the LED (P20) is 64 ms.</p> 

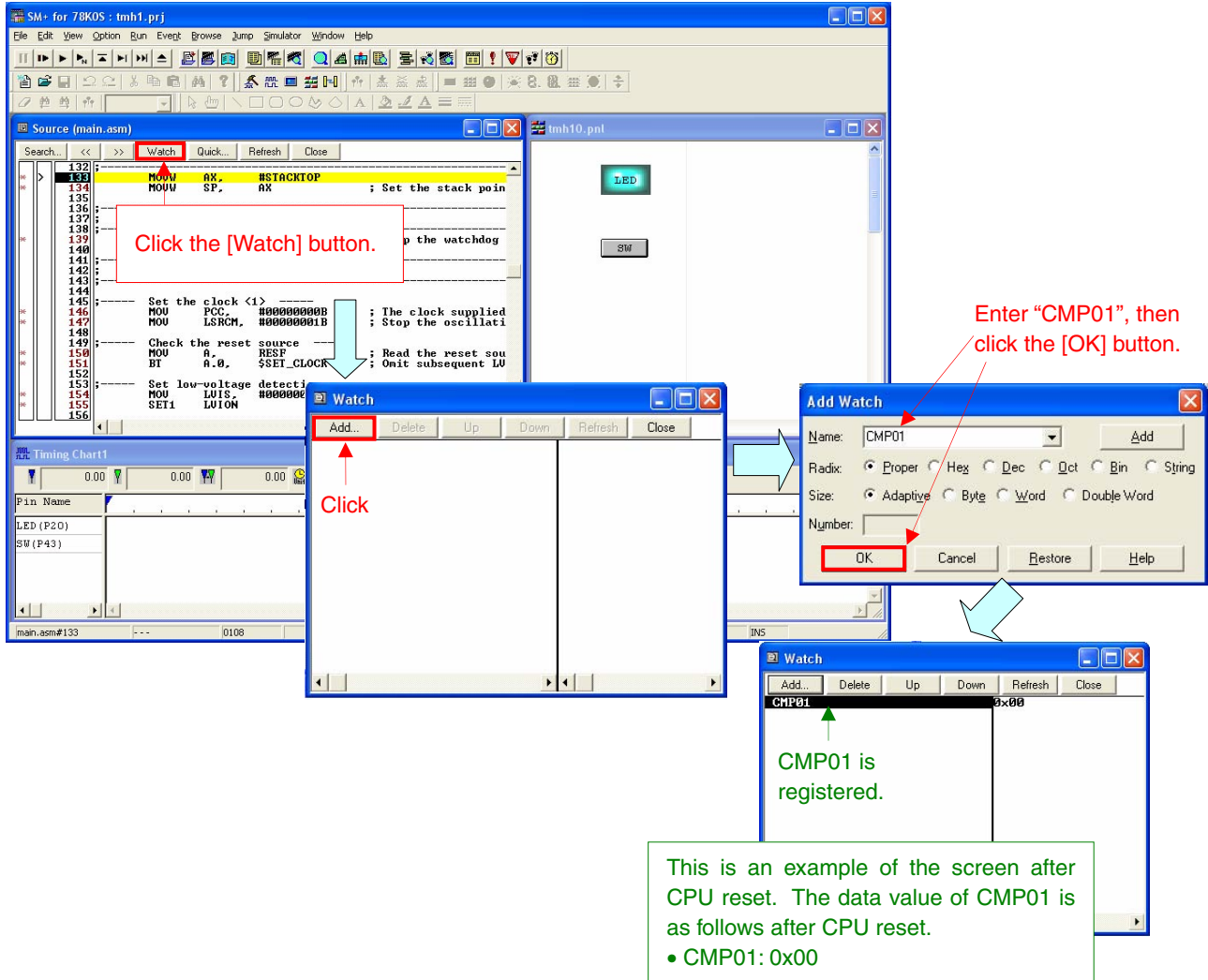
Note 1

- Notes** 1. The blinking cycle from the zeroth [SW] button input is repeated after the fourth [SW] button input.  
 2. This may differ from the actual blinking cycle, depending on the operation environment of the PC used.



[Supplement 1] The changes in the data value of the CMP01 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 "CMP01" in the [Name] field and click the [OK] button to register "CMP01" in the [Watch] window and close the [Add Watch] window.




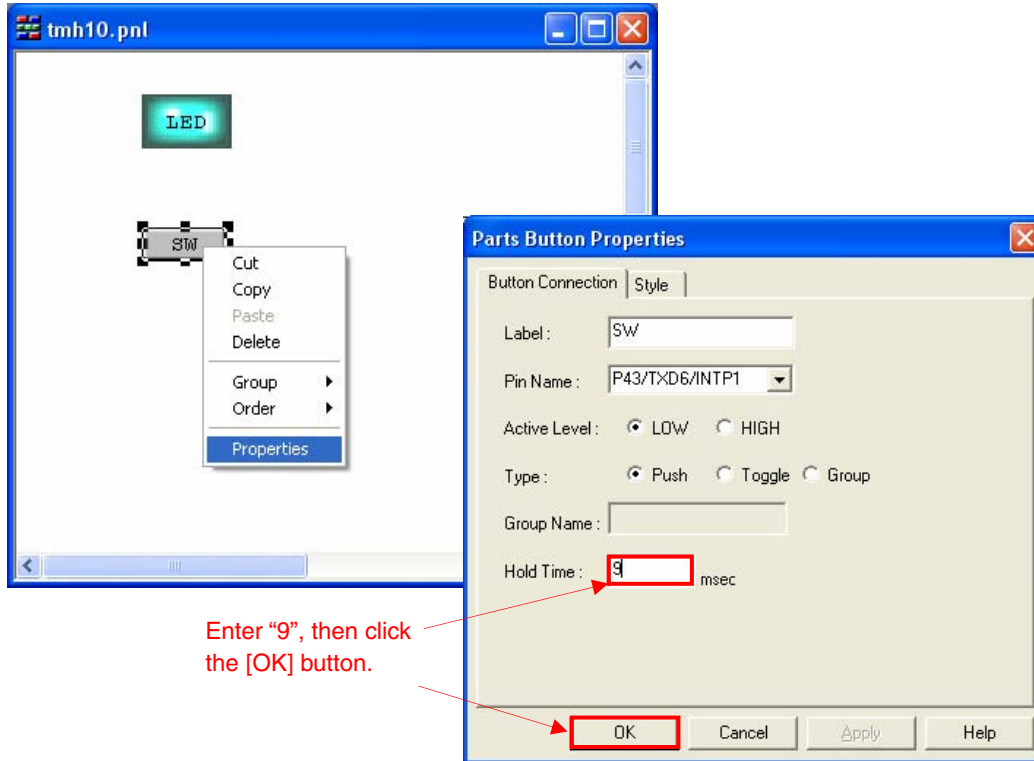
- <4> Execute the program and click the [SW] button in the I/O panel window. Check that the data value of CMP01 in the [Watch] window changes, depending on the number of [SW] button inputs.


Number of [SW] Button Inputs <sup>Note</sup>	Data Value in [Watch] Window
0	CMP01: 0xF9 (249)
1	CMP01: 0x7C (124)
2	CMP01: 0x3D (61)
3	CMP01: 0x1E (30)

**Note** The lighting patterns from the zeroth switch input are repeated after the fourth 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 LED blinking cycle 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>
78K0S/KA1+ Simplified Flash Writing Manual MINICUBE2 Information		<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 (8-bit Timer H1) PWM Output	<a href="#">PDF</a>

## 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
;*****
;   8-bit timer H1
;*****
;<<History>>
;   2007.7.-- Release
;*****
;
;<<Overview>>
;
;This sample program presents an example of using the interval timer function
;of 8-bit timer H1. The LEDs are blinked by reversing the P20 pin output
;through the use of 8-bit timer H1 interrupts. The LED blinking cycle is
;changed by rewriting the compare register of the timer when a switch input
;interrupt is generated.
;
;
; <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
;
;
; <8-bit timer H1 settings>
; - Set to the interval timer mode
; - Disable timer output of the TOH1 pin
; - Count clock = fxp/26 (125 kHz)
; - Initial value of timer cycle = 2 ms (8[us/cclk] x 250[count] = 2[ms])
;
;
; <Number of switch inputs and LED blinking cycles>
;
;
; +-----+
; | SW Inputs | LED Blinking |
; | (P43)     | Cycle (P20)  |
; +-----+ +-----+
; | 0 times   | 1 second     |
; | 1 time    | 1/2 second   |
; | 2 times   | 1/4 second   |
; | 3 times   | 1/8 second   |
; +-----+ +-----+

```

```

; +-----+
; # The blinking cycle from the zeroth switch input is repeated after the
fourth 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.
;
;*****
;=====
;
; Vector table
;
;=====
XVCT CSEG 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 INTERRUPT_TMH1 ;(0C) INTTMH1
      DW RESET_START ;(0E) INTTM000
      DW RESET_START ;(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
;
;=====
XROM CSEG AT 0100H
;----- For setting the timer H1 cycle -----
      DB 250-1 ; 2 ms interval compare value
      DB 125-1 ; 1 ms interval compare value
      DB 63-1 ; 0.5 ms interval compare value
      DB 32-1 ; 0.25 ms interval compare value
;----- For handling chattering -----
      DB 5+1 ; Count value for handling chattering (for 2 ms
interval)
      DB 10+1 ; Count value for handling chattering (for 1 ms
interval)
      DB 20+1 ; Count value for handling chattering (for 0.5 ms
interval)
      DB 40+1 ; Count value for handling chattering (for 0.25 ms
interval)

```

```

;=====
;
;   Define the RAM
;
;=====
XRAM DSEG SADDR
CNT_TMH1: DS 1 ; For counting INTTMH1 interrupt

;=====
;
;   Define the memory stack area
;
;=====
XSTK DSEG 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/clock] 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,    #00000001B ; Set output latches of P21-P23 as low, P20 as
high (turn off LED)
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

;-----
; Initialize the general-purpose register and RAM
;-----

```

```

MOV CNT_TMH1, #250 ; Initialize the number of INTTMH1 interrupts
MOVW HL, #0100H ; Specify the table address to HL (used for
INTP1 interrupt)

;-----
; Set 8-bit timer H1
;-----
MOV TMHMD1, #00110000B ; Count clock = fxp/26 = 125 kHz, set to
the interval timer mode
MOV A, [HL] ; Read from the table the base time initial
values for blinking the LEDs
MOV CMP01, A ; Initialize the compare values
SET1 TMHE1 ; Start the timer operation

;-----
; 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
CLR1 TMMKH1 ; Unmask INTTMH1 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, [HL+4] ; Read the count value corresponding to the
timer H1 cycle
WAIT_CHAT:
NOP
BF TMIFH1, $WAIT_CHAT ; Wait for the INTTMH1 interrupt
CLR1 TMIFH1 ; Clear the INTTMH1 interrupt request flag
CALL !SUB_INTERRUPT_TMH1 ; Service the INTTMH1 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

;----- Change the TMH1 interval cycle -----

```



```

CLR1  TMHE1          ; Stop the timer operation

MOV   A,    L        ; Read the lower 8 bits of the table address
INC   A              ; Increment the table address by 1
AND   A,    #00000011B ; Mask bits other than bits 0 and 1
MOV   L,    A        ; Write to the lower 8 bits of the table address
MOV   A,    [HL]     ; Read the table data
MOV   CMP01,   A     ; Change the LED blinking base time

SET1  TMHE1          ; Start the timer operation

MOV   CNT_TMH1, #250 ; Initialize the number of INTTMH1 interrupts

END_INTP1:
POP   AX            ; Restore the AX register data
RETI                          ; Return from interrupt servicing

;*****
;
;   Interrupt INTTMH1
;
;*****
INTERRUPT_TMH1:
CALL  !SUB_INTERRUPT_TMH1    ; Service the INTTMH1 interrupt
RETI                          ; Return from interrupt servicing

;-----
;   Subroutine for measuring the number of INTTMH1 interrupts
;-----
SUB_INTERRUPT_TMH1:
DBNZ  CNT_TMH1, $END_INTTMH1 ; Branch if the number of INTTMH1
interrupts < 250
MOV   CNT_TMH1, #250        ; Initialize the number of INTTMH1 interrupts

XOR   P2,    #00000001B    ; Reverse the LED output
END_INTTMH1:
RET                          ; Return from the subroutine

end

```

● main.c (C language version)

```

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

*****
      78K0S/KB1+  Sample program
*****
      8-bit timer H1
*****
<<History>>
      2007.7.--  Release
*****

```

<<Overview>>

This sample program presents an example of using the interval timer function of 8-bit timer H1. The LEDs are blinked by reversing the P20 pin output through the use of 8-bit timer H1 interrupts. The LED blinking cycle is changed by rewriting the compare register of the timer when a switch input interrupt is generated.

<Principal setting contents>

- Declare a function run by an interrupt: INTP1 -> fn\_intp1()
- Declare a function run by an interrupt: INTTMH1 -> fn\_inttmh1()
- 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

<8-bit timer H1 settings>

- Set to the interval timer mode
- Disable timer output of the TOH1 pin
- Count clock = f<sub>clk</sub>/2<sup>6</sup> (125 kHz)
- Initial value of timer cycle = 2 ms (8[us/clock] x 250[count] = 2[ms])

<Number of switch inputs and LED blinking cycles>

+-----+

SW Inputs (P43)	LED Blinking Cycle (P20)
0 times	1 second
1 time	1/2 second
2 times	1/4 second
3 times	1/8 second

# The blinking cycle from the zeroth switch input is repeated after the fourth 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 NOP /\* NOP instructions can be described at the C source level \*/

#pragma interrupt INTP1 fn\_intp1 /\* Interrupt function declaration:INTP1 \*/

#pragma interrupt INTTMH1 fn\_inttmh1 /\* Interrupt function declaration:INTTMH1 \*/

/\*=====

Declare the function prototype

=====\*/

void fn\_subinttmh1(); /\* INTTMH1 interrupt subroutine \*/

/\*=====

Define the global variables

```

=====*/
sreg unsigned char g_ucSWcnt = 0; /* 8-bit variable for counting the number
of switch inputs */
sreg unsigned char g_ucTMH1cnt = 0; /* 8-bit variable for counting the number
of INTTMH1 interrupts */
const unsigned char g_ucChat[4] = {5+1,10+1,20+1,40+1}; /* 8-bit
constant table for removing chattering */
const unsigned char g_ucCMPdata[4] = {250-1,125-1,63-1,32-1}; /* 8-bit
constant table for LED blinking base time */

/*****

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    = 0b00000001; /* Set output latches of P21-P23 as low,
P20 as high (turn off LED) */
    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 8-bit timer H1
-----*/
    TMHMD1 = 0b00110000;          /* Count clock = fxp/26 = 125 kHz, set to
the interval timer mode */
    CMP01 = 250-1;                /* Initialize the LED blinking base time
*/
    TMHE1 = 1;                    /* Start the timer operation */

/*-----
    Set the interrupt
-----*/
    INTM0 = 0b00000000;          /* Set the valid edge of INTP1 to falling
edge */
    IF0    = 0x00;                /* Clear invalid interrupt requests in
advance */
    PMK1   = 0;                   /* Unmask INTP1 interrupts */
    TMMKH1 = 0;                   /* Unmask INTTMH1 interrupts */

    return;
}

/*****

    Main loop

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

    EI();                          /* Enable vector interrupt */

    while (1){
        NOP();
        NOP();
    }
}

/*****

    External interrupt INTP1

*****/
__interrupt void fn_intp1(){
    unsigned char ucChat;          /* 8-bit variable for removing chattering
*/

```

```

    for (ucChat = g_ucChat[g_ucSWcnt] ; ucChat > 0 ; ucChat--){ /* Wait of
about 10 ms (for removing chattering) */
        while (!TMIFH1){ /* Wait for the INTTMH1 interrupt request */
            NOP();
        }

        TMIFH1 = 0;          /* Clear the INTTMH1 interrupt request flag */
        fn_subinttmH1(); /* Service the INTTMH1 interrupt */
    }

    PIF1 = 0;                /* Clear the INTP1 interrupt request */

    if (!P4.3){              /* Processing performed if SW is on for 10 ms or more
*/
        g_ucSWcnt = (g_ucSWcnt + 1) & 0b00000011; /* Increment the number
of switch inputs by 1 */

        TMHE1 = 0;          /* Stop the timer operation */
        CMP01 = g_ucCMPdata[g_ucSWcnt]; /* Change the LED blinking
base time in accordance with the number of switch inputs */
        TMHE1 = 1;          /* Start the timer operation */

        g_ucTMH1cnt = 0;    /* Clear the number of INTTMH1 interrupts
*/
    }

    return;
}

/*****

Interrupt INTTMH1

*****/
__interrupt void fn_inttmH1(){

    fn_subinttmH1();        /* Service the INTTMH1 interrupt */

    return;
}

/*-----
Subroutine for measuring the number of INTTMH1 interrupts
-----*/
void fn_subinttmH1(){

```

```

    if (++g_ucTMH1cnt == 250){ /* Processing when the number of INTTMH1
interrupts is 250 */
        g_ucTMH1cnt = 0; /* Clear the number of INTTMH1 interrupts */
        P2 ^= 0b00000001; /* Reverse the LED output */
    }

    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

Edition	Date Published	Page	Revision
1st edition	November 2007	–	–

*For further information,  
please contact:*

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

**[America]**

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

**[Europe]**

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

**Hanover Office**

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

**Munich Office**

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

**Stuttgart Office**

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

**United Kingdom Branch**

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

**Succursale Française**

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

**Sucursal en España**

Juan Esplandiú, 15  
28007 Madrid, Spain  
Tel: 091-504-2787

**Tyskland Filial**

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

**Filiale Italiana**

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

**Branch The Netherlands**

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

**[Asia & Oceania]**

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

**Shanghai Branch**

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

**Shenzhen Branch**

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

**NEC Electronics Hong Kong Ltd.**

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

**NEC Electronics Taiwan Ltd.**

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

**NEC Electronics Singapore Pte. Ltd.**

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

**NEC Electronics Korea Ltd.**

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