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.

Application Note

78K0S/Kx1+

Sample Program (8-bit Timer 80)

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 80. In the sample program, the LEDs are blinked at fixed cycles by using the interval timer function of 8-bit timer 80. 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

CONTENTS

Document No. U18864EJ2V0AN00 (2nd edition)
Date Published September 2008 NS

- **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 interval timer function of 8-bit timer 80 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.

Caution 8-bit timer 80 is not mounted with the 78K0S/KU1+ and 78K0S/KY1 microcontrollers.

1.1 Main Contents of the Initial Settings

The main contents of the initial settings are as follows.

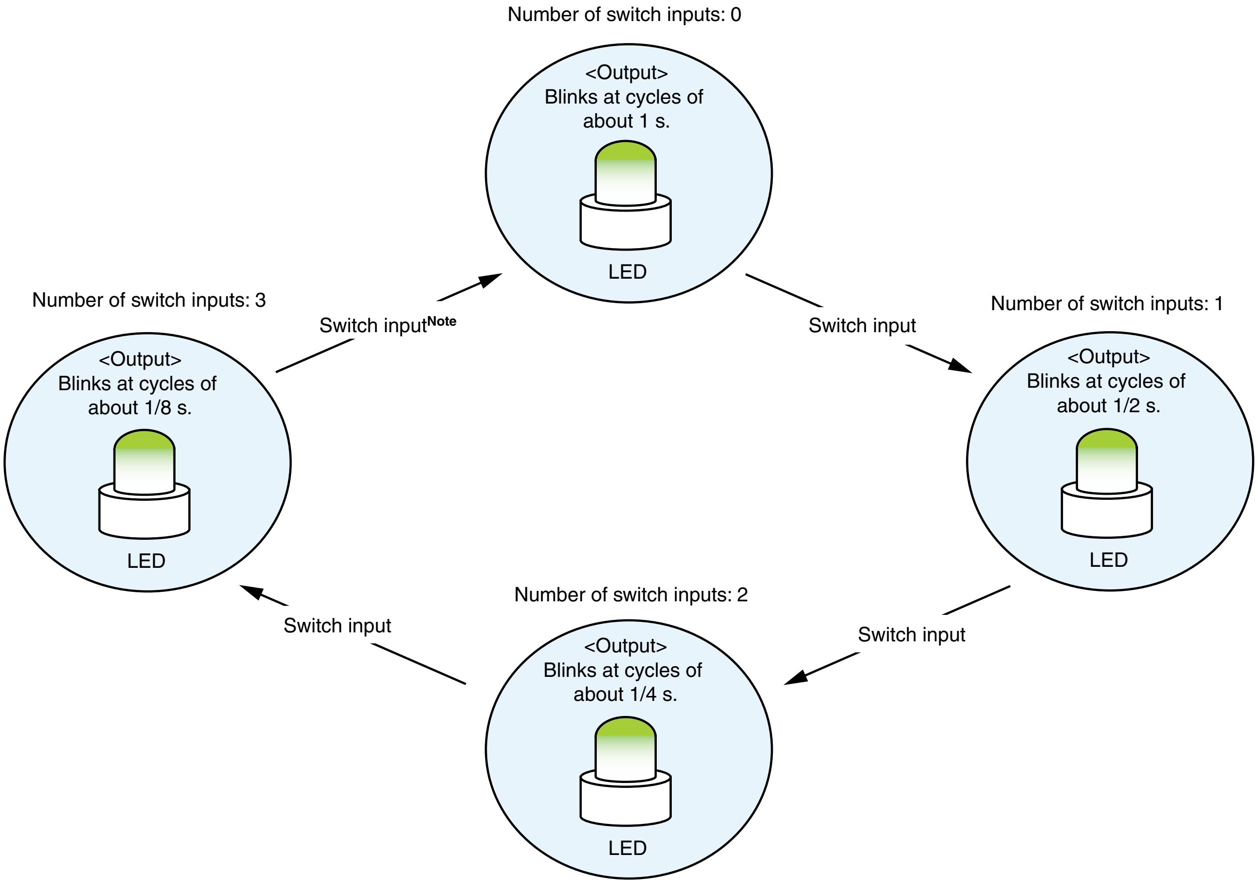
- Selecting the high-speed internal oscillator as the system clock source$^{\text{Note}}$
- Stopping watchdog timer operation
- Setting V_{LVI} (low-voltage detection voltage) to 4.3 V ±0.2 V
- Generating an internal reset (LVI reset) signal when it is detected that V_{DD} is less than V_{LVI}, after 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 8-bit timer 80
    - Setting the count clock to $f_{XP}/2^6$ (125 kHz)
    - Setting the interval cycle to 2 ms (8 μs × 250)
- Setting the valid edge of INTP1 (external interrupt) to the falling edge
- Enabling INTP1 and INTTM80 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 80 interrupt (INTTM80), 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/KA1+, 78K0S/KB1+).

[Column] Chattering

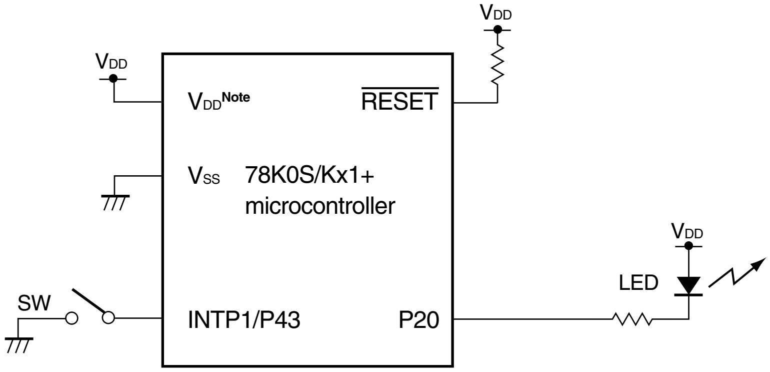
Chattering is a phenomenon in which the electric signal repeats turning on and off due to a mechanical flip-flop of the contacts, immediately after the switch has been pressed.

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.



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

Cautions 1. Connect the AV_{REF} pin directly to V_{DD}.

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 80 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		
		ZIP	PM 32	32
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)	●	●	
tm80.prw	Work space file for integrated development environment PM+		●	
tm80.prj	Project file for integrated development environment PM+		●	
tm80.pri tm80.prs tm80.prm	Project files for system simulator SM+ for 78K0S/Kx1+		●	
tm800.pnl	I/O panel file for system simulator SM+ for 78K0S/Kx1+ (used for checking peripheral hardware operations)		●	●
tm800.wvo	Timing chart file for system simulator SM+ for 78K0S/Kx1+ (used for checking waveforms)			●

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

Remark ZIP : Only the source file is included.

PM 32 : The files to be used with integrated development environment PM+ and 78K0S/Kx1+ system simulator SM+ are included.

32 : 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 80
- $V_{DD} < V_{LVI}$ detection: Low-voltage detector (LVI)
- Switch input: INTP1/P43 (external interrupt)
- LED output: P20 (output port)

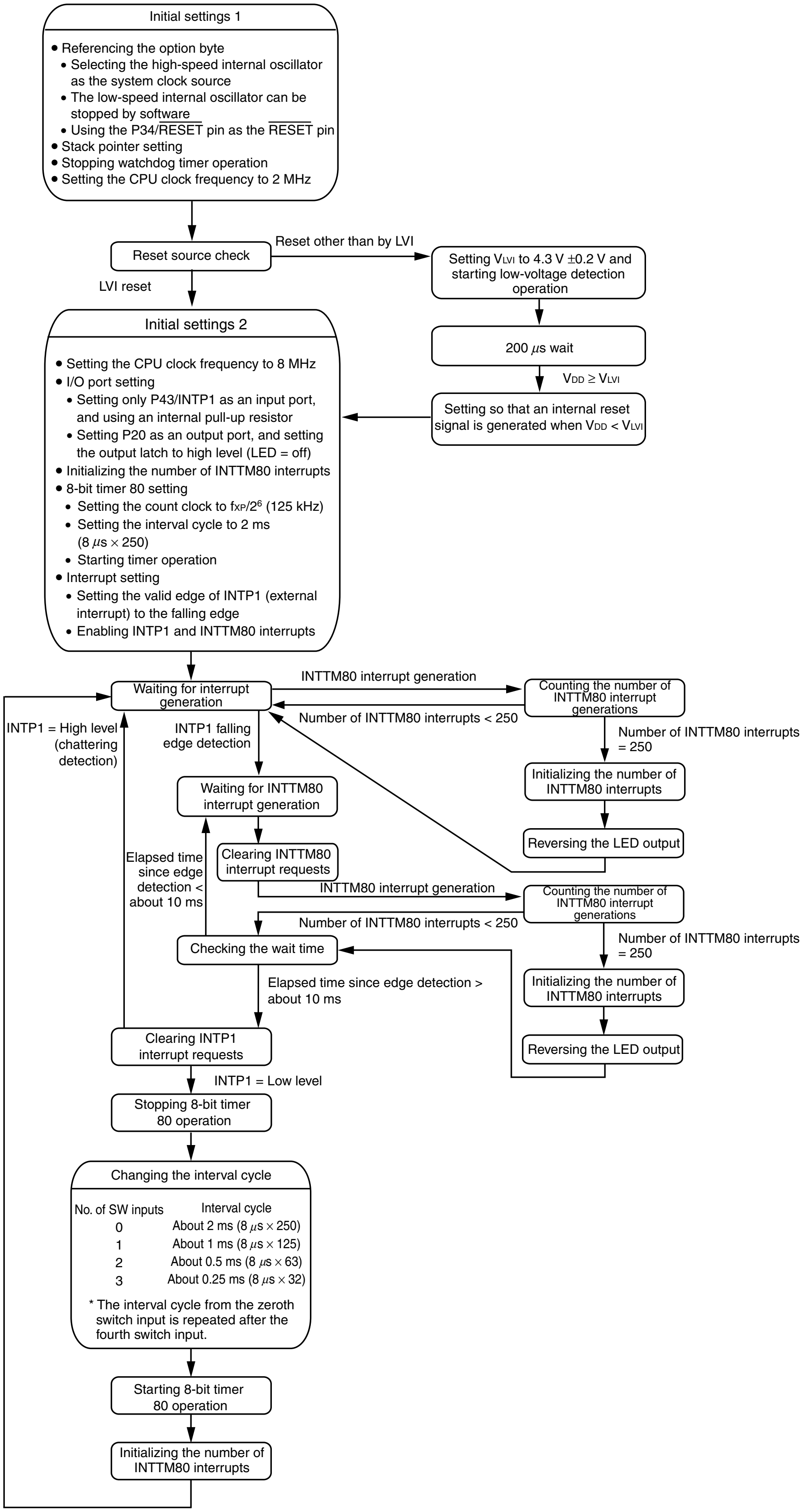
3.3 Initial Settings and Operation Overview

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 80 (interval timer), and setting of interrupts are performed.

The LEDs are blinked at fixed cycles by using the generation of an 8-bit timer 80 interrupt (INTTM80), after completion of the initial settings.

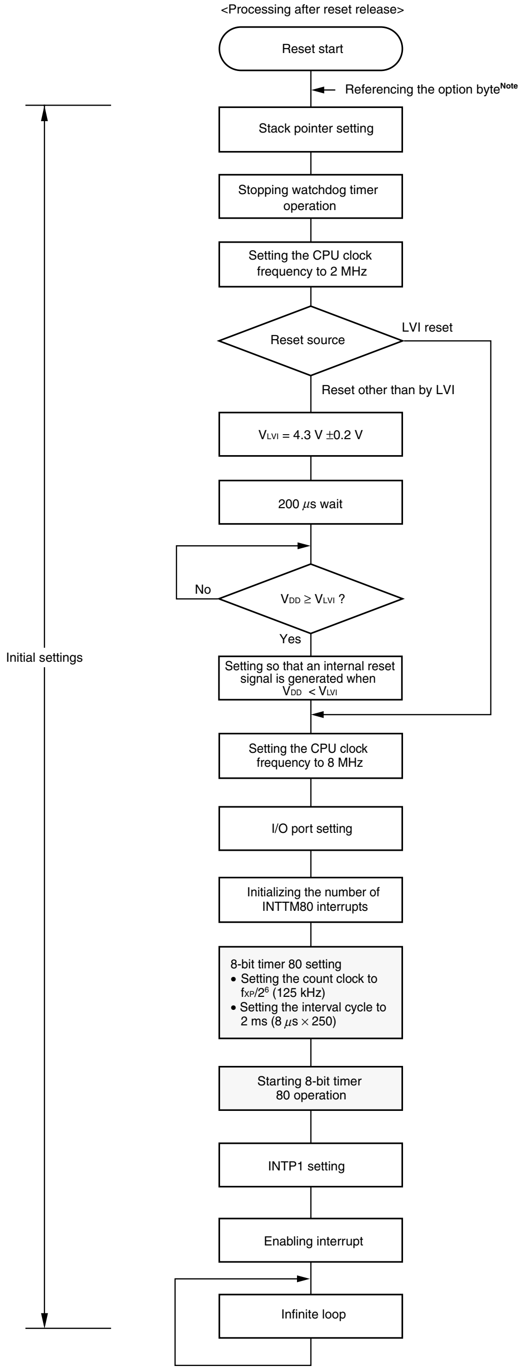
An INTP1 interrupt is serviced when the falling edge of the INTP1 pin, which is generated by switch input, is detected. Chattering is identified when INTP1 is at high level (switch is off), after 10 ms have elapsed since a fall of the INTP1 pin was detected. The blinking cycle of the LEDs is changed 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.

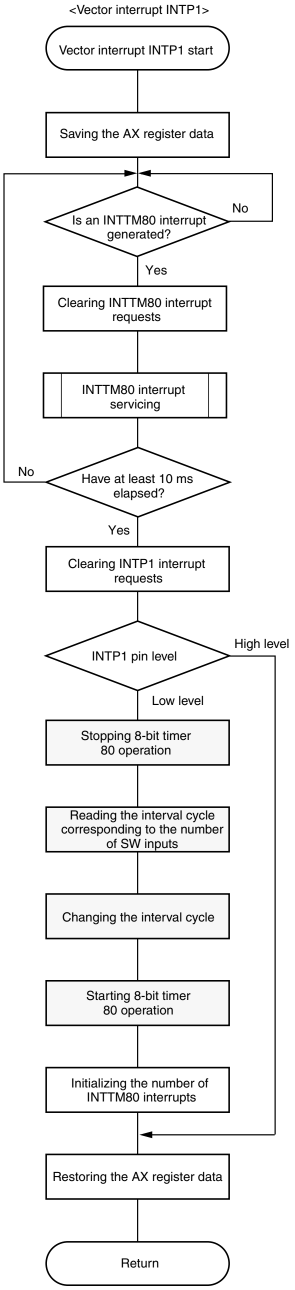
The details are described in the status transition diagram shown below.

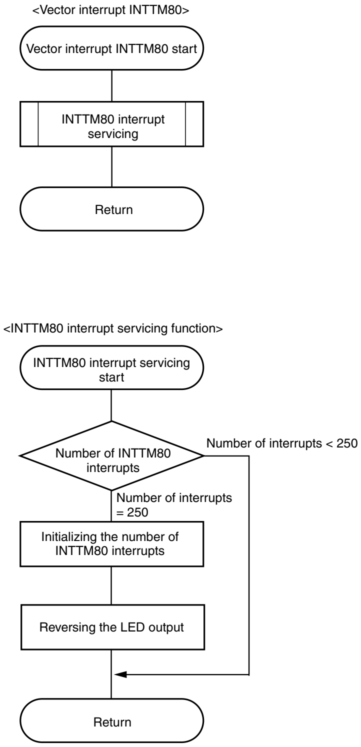


3.4 Flow Charts

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 interval timer function of 8-bit timer 80.

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/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 80

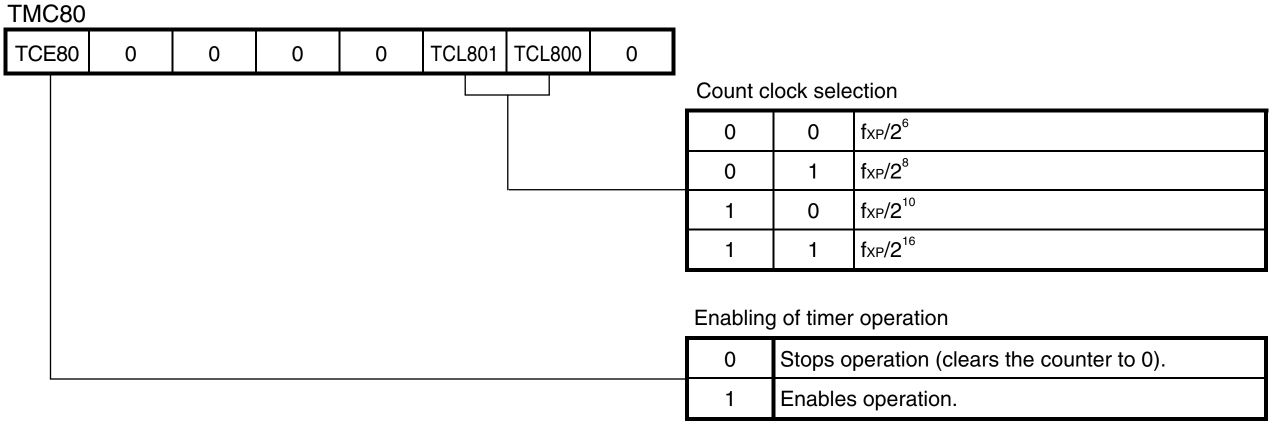
The following two types of registers are set when using 8-bit timer 80.

- 8-bit timer mode control register 80 (TMC80)
- 8-bit compare register 80 (CR80)

(1) Setting regarding the operation mode of 8-bit timer 80

The count clock of 8-bit timer 80 is selected and operation is controlled by using 8-bit timer mode control register 80 (TMC80).

Figure 4-1. Format of 8-bit Timer Mode Control Register 80 (TMC80)



TMC80

TCE80	0	0	0	0	TCL801	TCL800	0

Count clock selection

0	0	$f_{XP}/2^6$
0	1	$f_{XP}/2^8$
1	0	$f_{XP}/2^{10}$
1	1	$f_{XP}/2^{16}$

Enabling of timer operation

0	Stops operation (clears the counter to 0).
1	Enables operation.

Caution Setting the TCL801 bit and TCL800 bit is prohibited when TCE80 is set to 1.

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

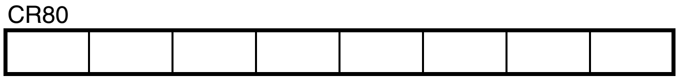
(2) Interval time setting

The interval time is set by using 8-bit compare register 80 (CR80).

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

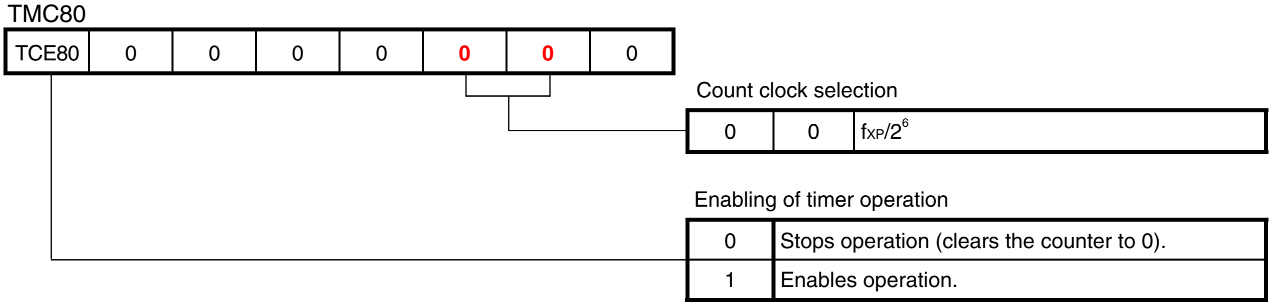
Remark N: CR80 setting value (00H to FFH)
f_{CNT}: Count clock frequency of 8-bit timer 80

Figure 4-2. Format of 8-bit Compare Register 80 (CR80)



Caution Rewriting the CR80 register value during timer count operation is prohibited.

[Example 1]
- Setting the count clock of 8-bit timer 80 to $f_{XP}/2^6$ (f_{XP} = 8 MHz)
- Setting the interval cycle to 2 ms, and starting timer operation
  (Same content as in the sample program)



TMC80

TCE80	0	0	0	0	0	0	0

Count clock selection

0	0	$f_{XP}/2^6$

Enabling of timer operation

0	Stops operation (clears the counter to 0).
1	Enables operation.

CR80 setting value (N): 249
- Count clock f_{CNT} = 8 MHz/2^6 = 0.125 MHz = 125 kHz
- Interval cycle 2 ms = (N + 1)/125 kHz

→ N = 2 ms × 125 kHz − 1 = 249

Timer operation is started by setting 1 to TCE80 after setting “00000000” to TMC80 and “249” to CR80.

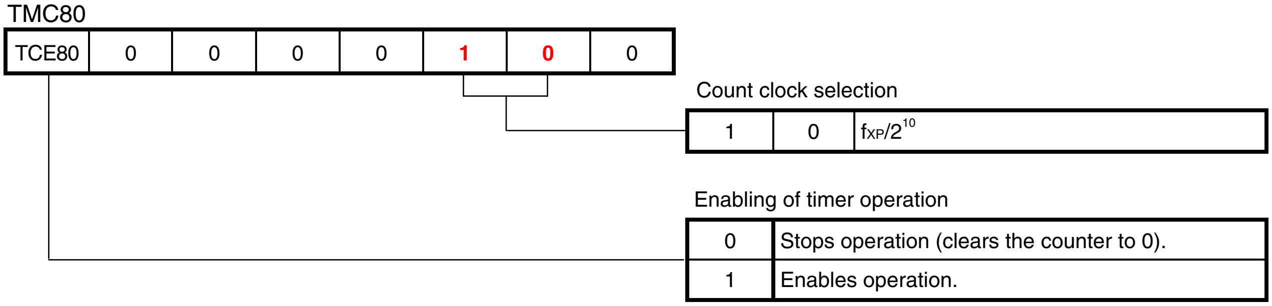
- Assembly language

```
MOV     TMC80, #00000000B
MOV     CR80,  #249
SET1    TCE80
```

- C language

```
TMC80 = 0b00000000;
CR80 = 249;
TCE80 = 1;
```

[Example 2]
- Setting the count clock of 8-bit timer 80 to $f_{XP}/2^{10}$ (f_{XP} = 8 MHz)
- Setting the interval cycle to 32 ms, and starting timer operation



TMC80

TCE80	0	0	0	0	1	0	0

Count clock selection

1	0	$f_{XP}/2^{10}$

Enabling of timer operation

0	Stops operation (clears the counter to 0).
1	Enables operation.

CR80 setting value (N): 249

- Count clock f_{CNT} = 8 MHz/2^{10} = 0.0078125 MHz = 7.8125 kHz
- Interval cycle 32 ms = (N + 1)/7.8125 kHz

→ N = 32 ms × 7.8125 kHz − 1 = 249

Timer operation is started by setting 1 to TCE80 after setting “00000100” to TMC80 and “249” to CR80.

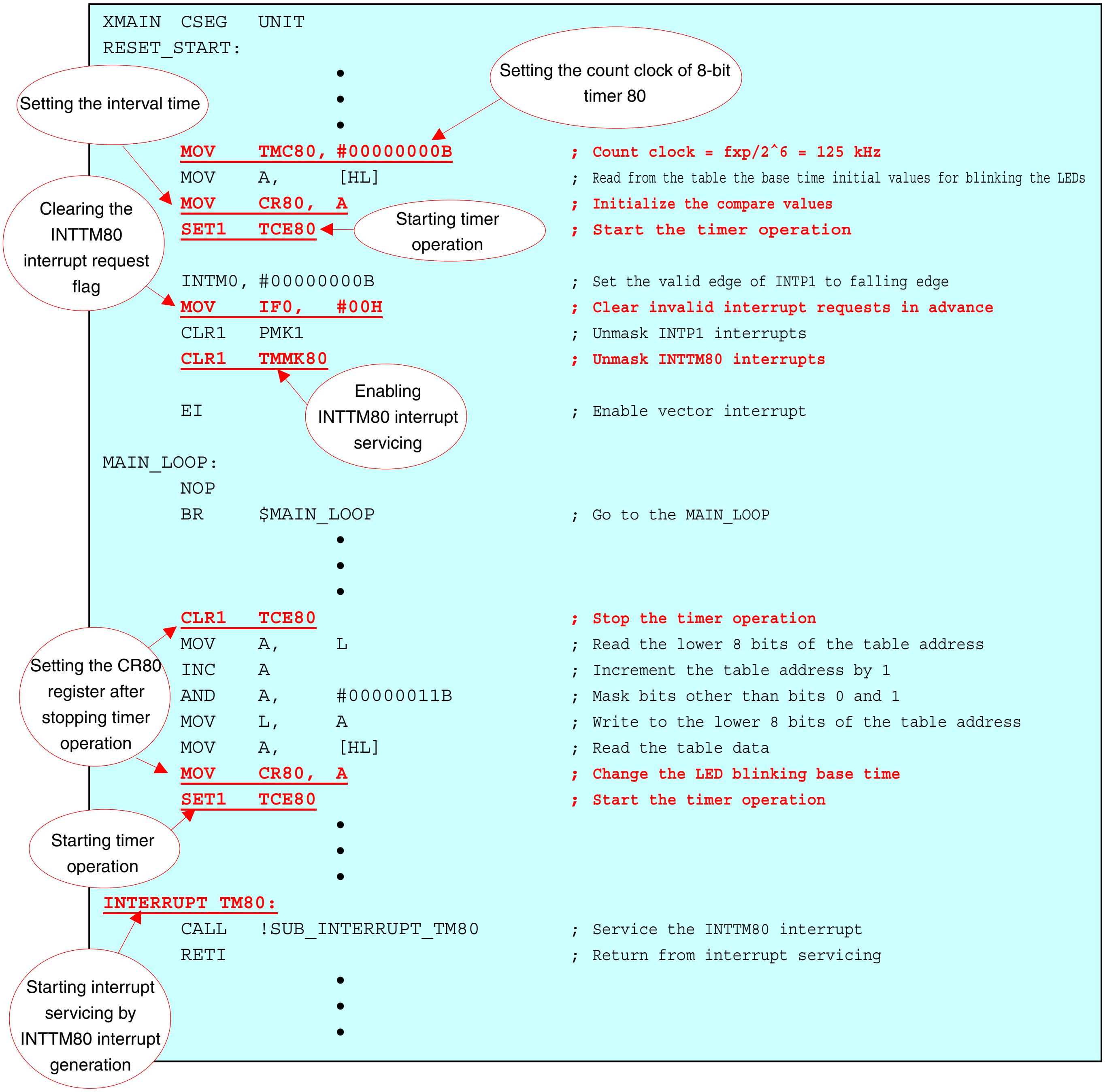
- Assembly language

```
MOV     TMC80, #00000100B
MOV     CR80,  #249
SET1    TCE80
```

- C language

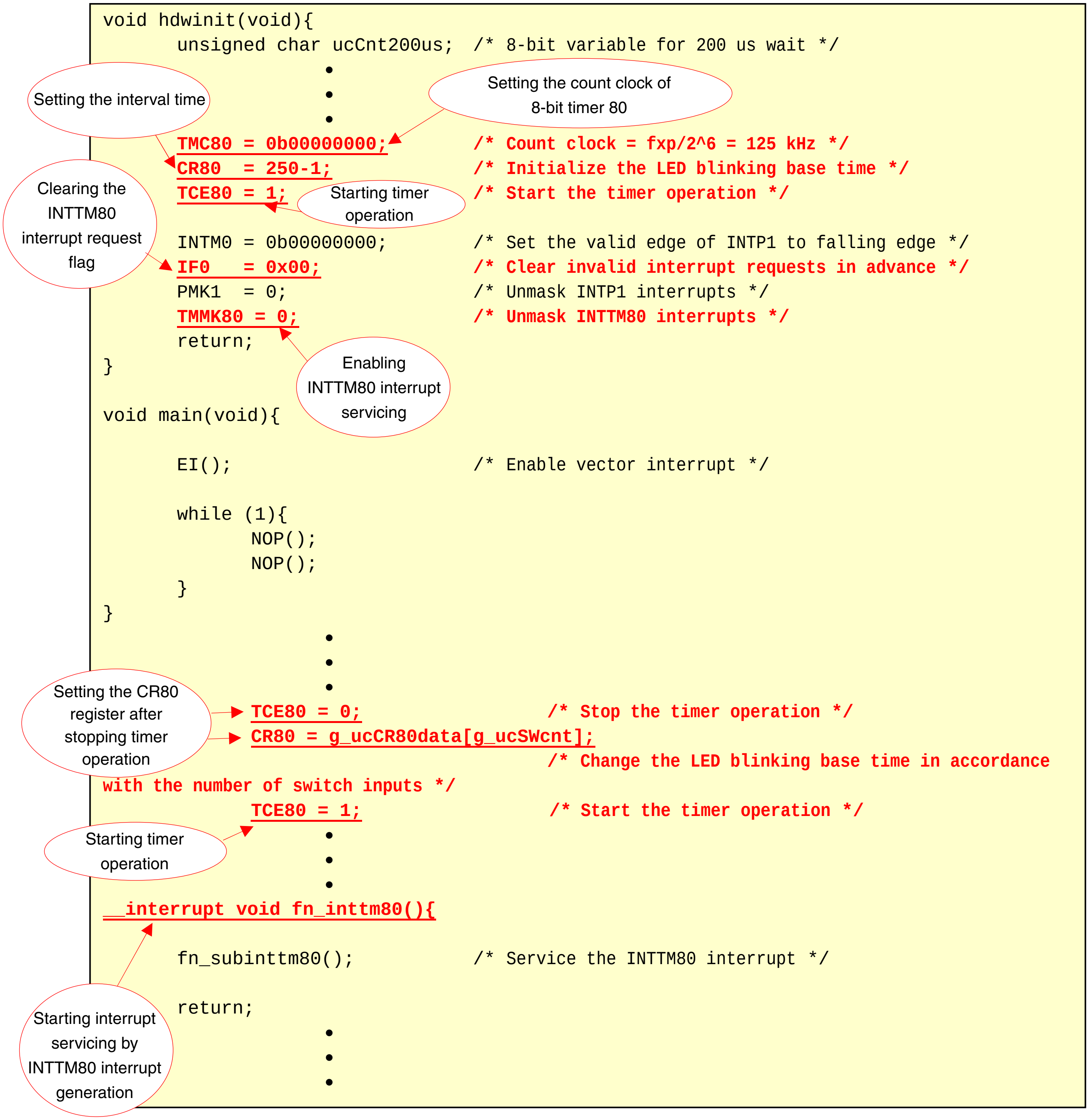
```
TMC80 = 0b00000100;
CR80 = 249;
TCE80 = 1;
```

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



```
XMAIN  CSEG   UNIT
RESET_START:
                    •
                    •
                    •
       MOV    TMC80, #00000000B                ; Count clock = fxp/2^6 = 125 kHz
       MOV    A,      [HL]                     ; Read from the table the base time initial values for blinking the LEDs
       MOV    CR80,  A                         ; Initialize the compare values
       SET1   TCE80                            ; Start the timer operation

       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   TMMK80                           ; Unmask INTTM80 interrupts

       EI                                      ; Enable vector interrupt

MAIN_LOOP:
       NOP
       BR     $MAIN_LOOP                       ; Go to the MAIN_LOOP
                    •
                    •
                    •
       CLR1   TCE80                            ; 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    CR80,  A                         ; Change the LED blinking base time
       SET1   TCE80                            ; Start the timer operation
                    •
                    •
                    •
INTERRUPT_TM80:
       CALL   !SUB_INTERRUPT_TM80              ; Service the INTTM80 interrupt
       RETI                                    ; Return from interrupt servicing
                    •
                    •
                    •
```

- 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 */
                                •
                                •
                                •
        TMC80 = 0b00000000;        /* Count clock = fxp/2^6 = 125 kHz */
        CR80  = 250-1;             /* Initialize the LED blinking base time */
        TCE80 = 1;                 /* 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 */
        TMMK80 = 0;                /* Unmask INTTM80 interrupts */
        return;
}

void main(void){

        EI();                      /* Enable vector interrupt */

        while (1){
                NOP();
                NOP();
        }
}
                                •
                                •
                                •
                TCE80 = 0;                 /* Stop the timer operation */
                CR80 = g_ucCR80data[g_ucSWcnt];
                                           /* Change the LED blinking base time in accordance
with the number of switch inputs */
                TCE80 = 1;                 /* Start the timer operation */
                                •
                                •
                                •
__interrupt void fn_inttm80(){

        fn_subinttm80();           /* Service the INTTM80 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 80 interrupts (INTTM80) in this sample program.

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

Remark N: CR80 register setting value
f_{CNT}: Count clock frequency of 8-bit timer 80

Calculation example: The following values result when the CR80 register setting value is 249 (during operation at f_{CNT} = 125 kHz).

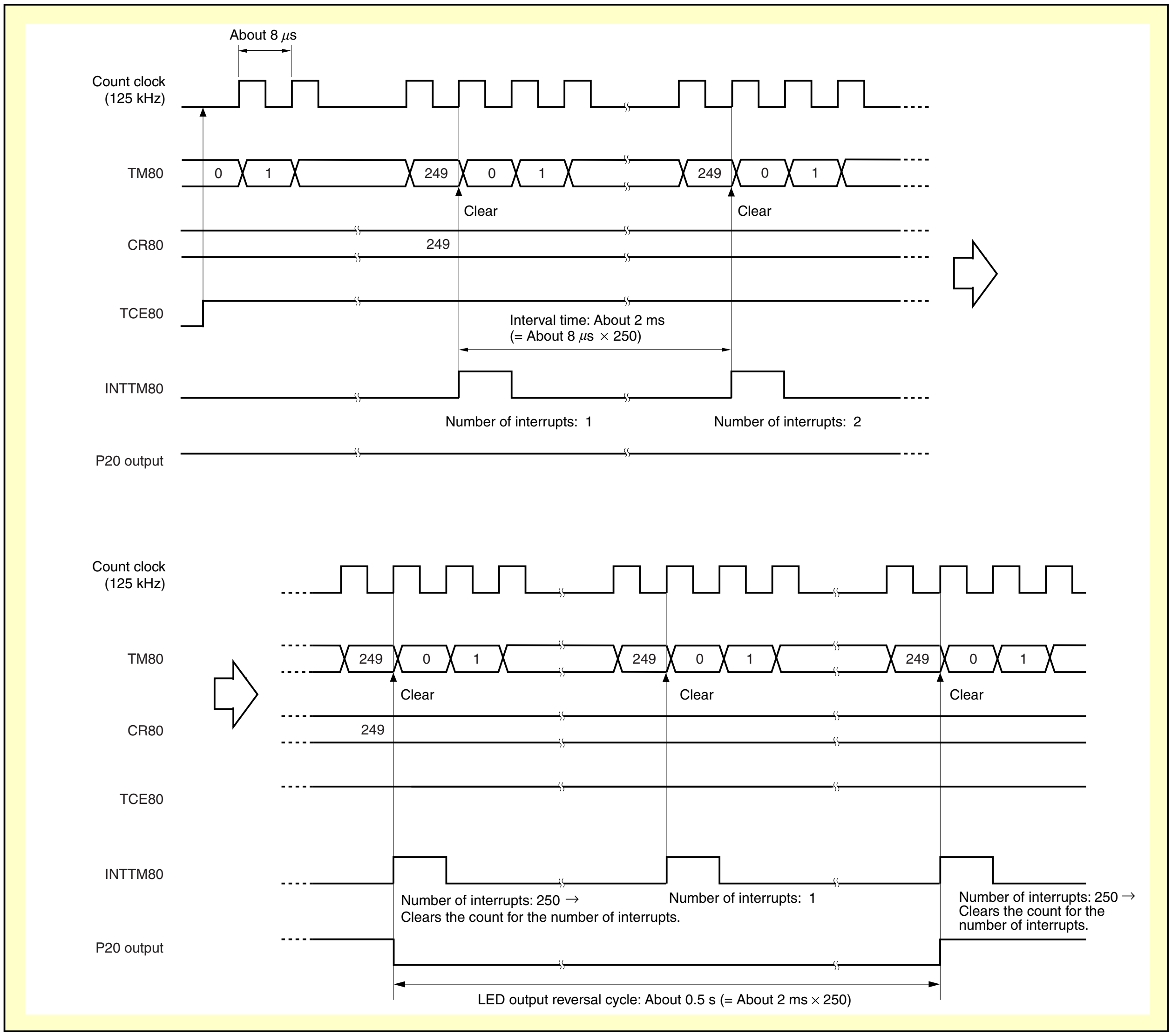
- Interrupt cycle (interval time) = $(N + 1)/f_{CNT}$ = (249 + 1)/125 kHz = 2 ms
- LED output reversal cycle = Interrupt cycle × Number of interrupts = 2 ms × 250 = 500 ms
- LED blinking cycle = LED output reversal cycle × 2 = 500 ms × 2 = 1 s

Furthermore, the CR80 register setting value is changed in accordance with the number of switch inputs, and the LED blinking cycle is changed.

Number of Switch Inputs[Note]	CR80 Register Setting Value	Interrupt Cycle	LED Blinking Cycle
0	249	About 2 ms ((249 + 1)/125 kHz)	About 1 s (about 2 ms × 250 × 2)
1	124	About 1 ms ((124 + 1)/125 kHz)	About 0.5 s (about 1 ms × 250 × 2)
2	62	About 0.504 ms ((62 + 1)/125 kHz)	About 0.252 s (about 504 μs × 250 × 2)
3	31	About 0.256 ms ((31 + 1)/125 kHz)	About 0.128 s (about 256 μs × 250 × 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 CR80 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 80 interrupts (INTTM80) 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.

INTTM80 interrupts can be continuously counted even during chattering detection by using INTTM80 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: INTTM80 interrupt cycle

T': Time from the start of INTP1 edge detection until the first INTTM80 is generated after INTP1 edge detection ($0 < T' \leq T$)

M: Number of INTTM80 interrupts after INTP1 edge detection

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

$T_C = T' + 10$ ms

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

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

↓

Chattering detection time (T_C) > 10 ms

Calculation example: When the interrupt cycle (T) is 2 ms (refer to the calculation example in **(1) Setting the LED blinking cycle**), and the number of INTTM80 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,

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

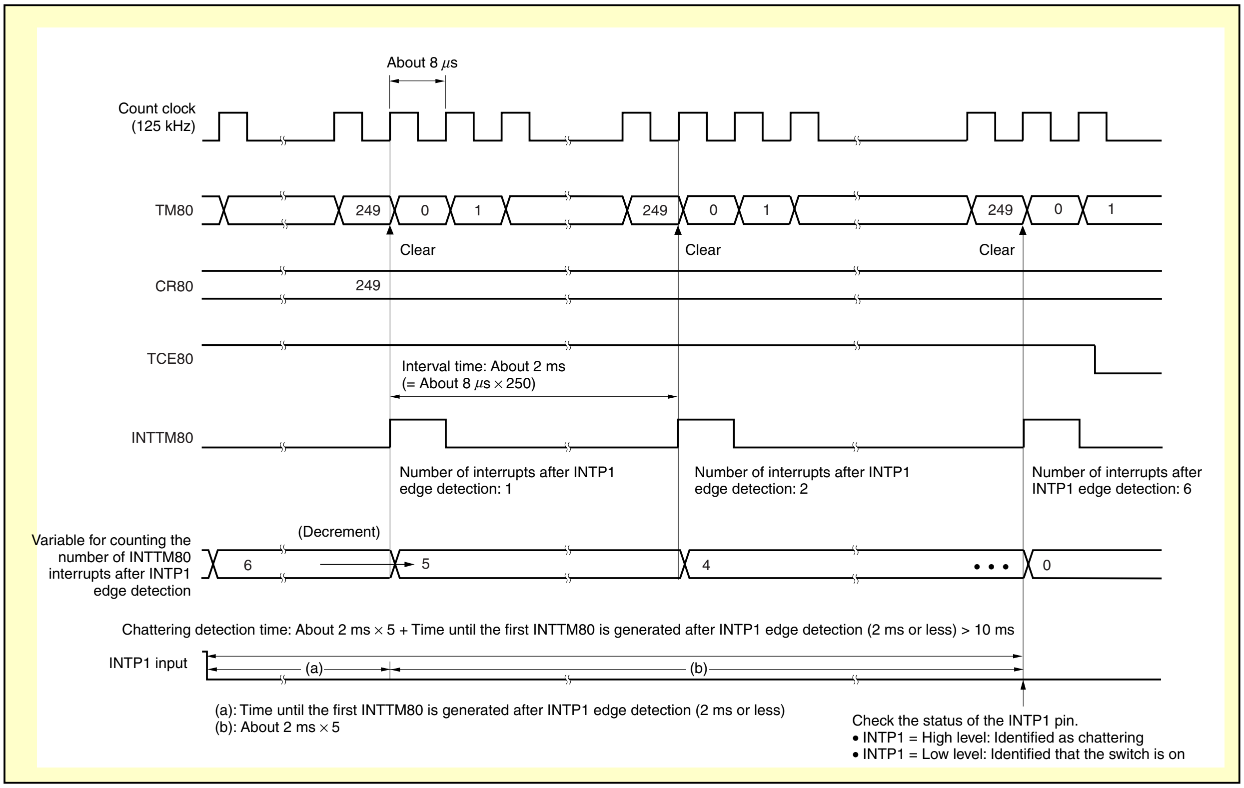
↓

Chattering detection time (T_C) > 10 ms

The following table shows the correspondence between the interrupt cycles during switch input and the number of INTTM80 interrupts after INTP1 edge detection in this sample program.

LED Blinking Cycle	Interrupt Cycle	Number of INTTM80 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 INTTM80 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 PM 32 icon.

<R> 5.1 Building the Sample Program

To check the operation of the sample program by using system simulator SM+ for 78K0S/Kx1+ (hereinafter referred to as “SM+”), SM+ must be started after building the sample program. This section describes how to build a sample program by using the assembly language sample program (source program + project file) downloaded by clicking the PM 32 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.)

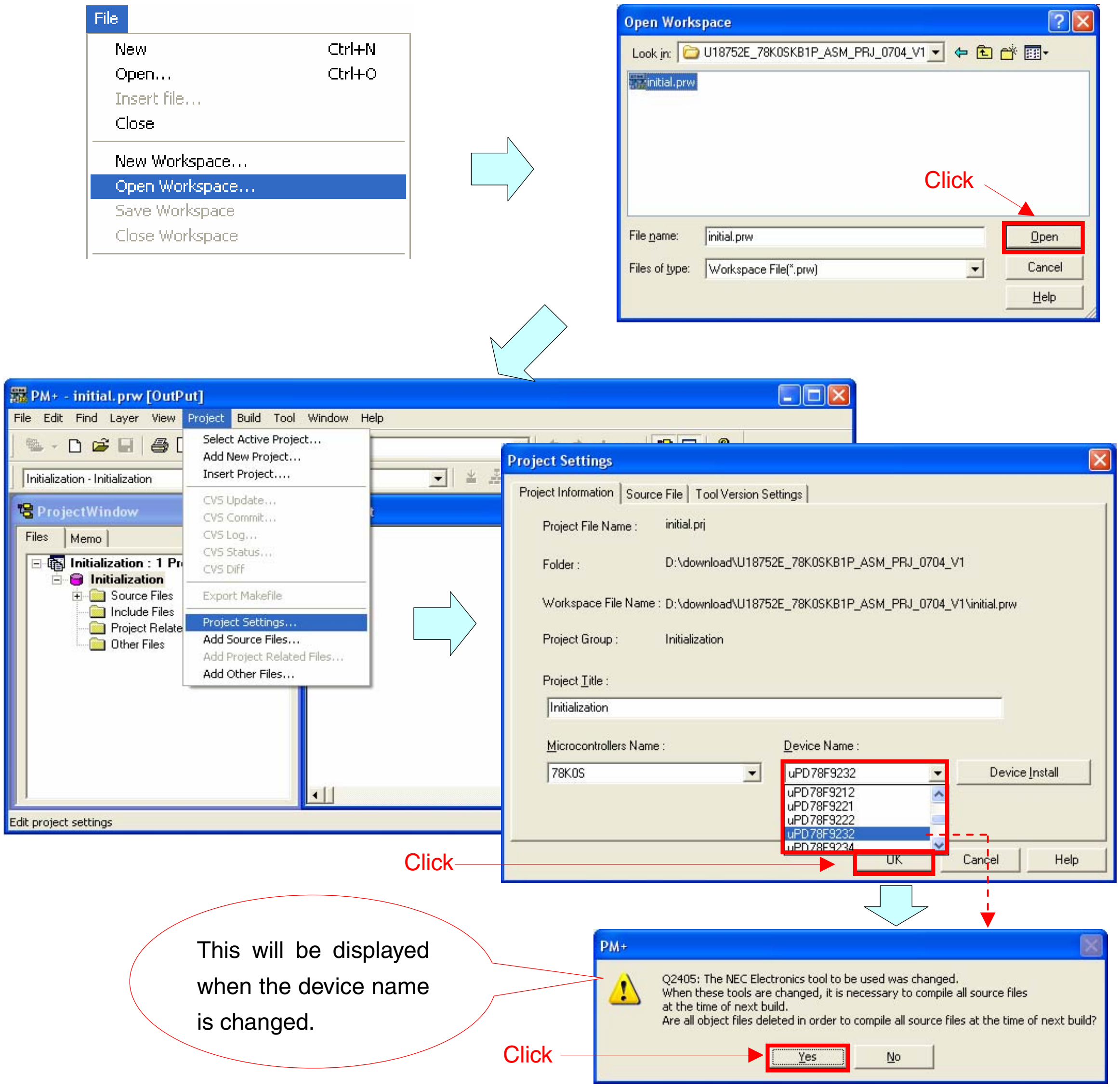
A RAM area of 118 bytes that has been secured as a fixed standard library area will be enabled for use when the [Using Fixed Area of Standard Library] check box is unchecked; however, the standard libraries (such as the getchar function and malloc function) will be disabled for use.
The [Using Fixed Area of Standard Library] check box is unchecked by default when the file that has been downloaded by clicking the PM 32 icon is used in this sample program.

(1) Start PM+.

(2) Select “tm80.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.



File name: initial.prw

Files of type: Workspace File(*.prw)

Project File Name : initial.prj

Folder : D:\download\U18752E_78K0SKB1P_ASM_PRJ_0704_V1

Workspace File Name : D:\download\U18752E_78K0SKB1P_ASM_PRJ_0704_V1\initial.prw

Project Group : Initialization

Project Title : Initialization

Microcontrollers Name : 78K0S

Device Name : uPD78F9232

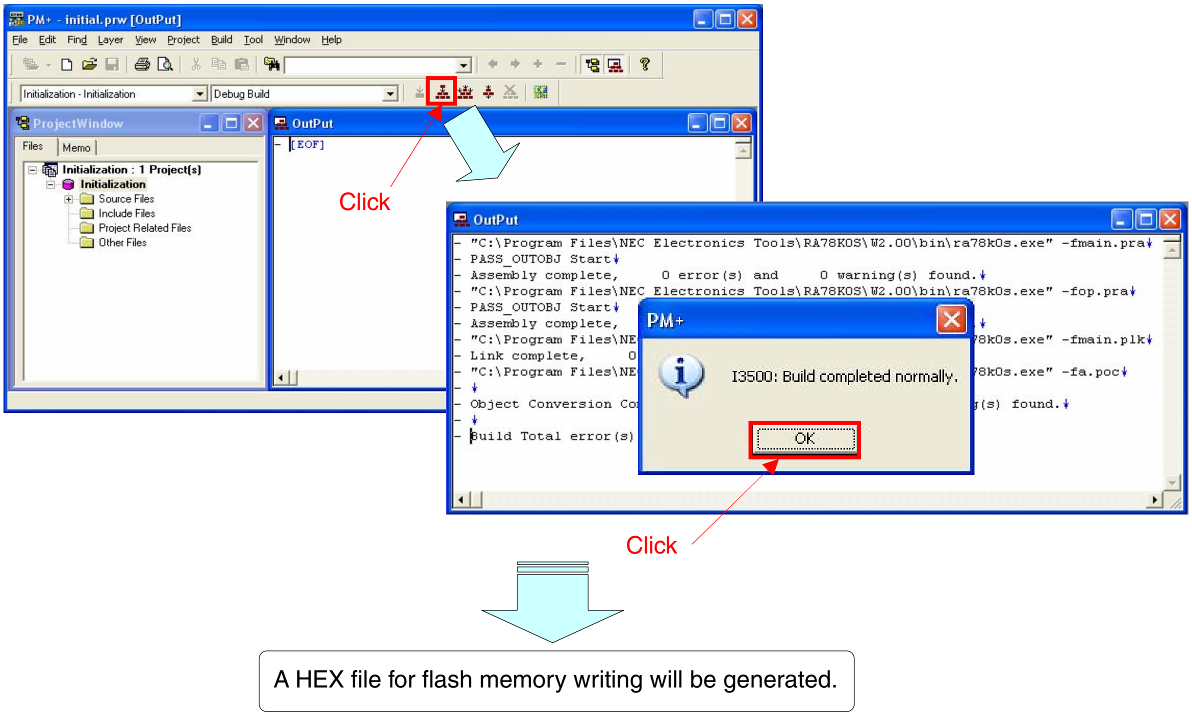
Q2405: The NEC Electronics tool to be used was changed.
When these tools are changed, it is necessary to compile all source files at the time of next build.
Are all object files deleted in order to compile all source files at the time of next build?

This will be displayed when the device name is changed.

(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.



```
- [EOF]
```

```
- "C:\Program Files\NEC Electronics Tools\RA78K0S\W2.00\bin\ra78k0s.exe" -fmain.pra↓
- PASS_OUTOBJ Start↓
- Assembly complete,      0 error(s) and      0 warning(s) found.↓
- "C:\Program Files\NEC Electronics Tools\RA78K0S\W2.00\bin\ra78k0s.exe" -fop.pra↓
- PASS_OUTOBJ Start↓
- Assembly complete,
- "C:\Program Files\NE                              78k0s.exe" -fmain.plk↓
- Link complete,      0
- "C:\Program Files\NE                              78k0s.exe" -fa.poc↓
- ↓
- Object Conversion Co                              g(s)  found.↓
- ↓
- Build Total error(s)
```

I3500: Build completed normally.

A HEX file for flash memory writing will be generated.

5.2 Operation with SM+

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

<R> (1) When SM+ for 78K0S/Kx1+ W1.02 (“SM+” hereafter) is used in the environment of PM+ Ver. 6.30, SM+ cannot be selected as the debugger. In this case, start SM+ via method (a) or (b) described below, while keeping PM+ running after completing building a project.

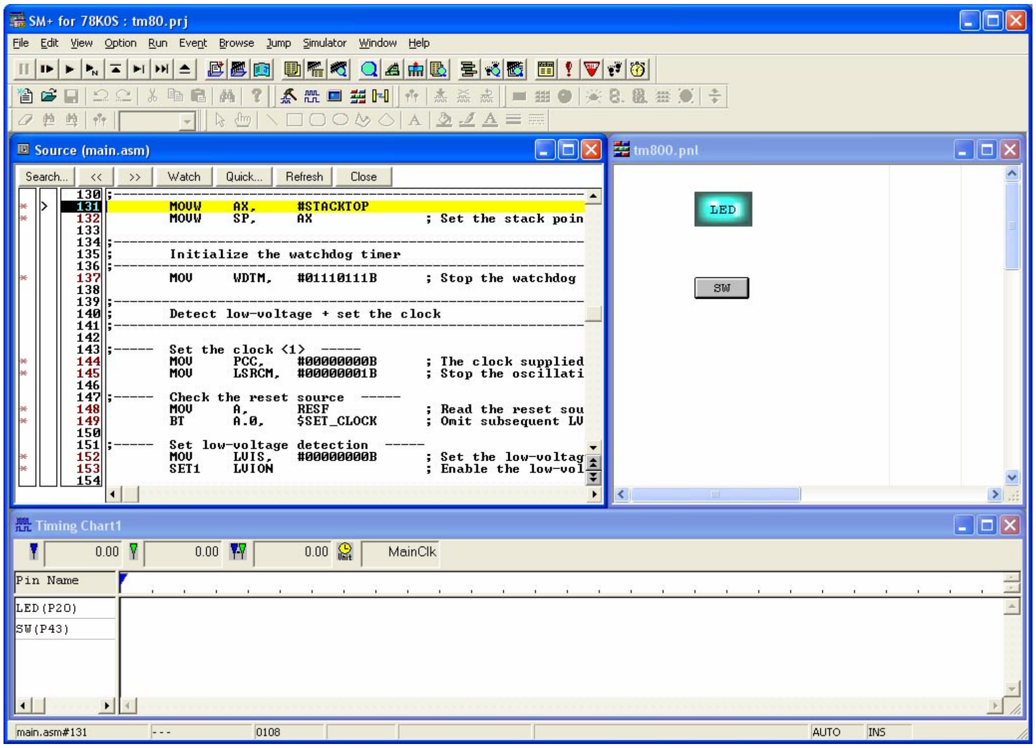
(a) When starting SM+ in PM+

<1> Select [Register Ex-tool] from the [Tool] menu and register “SM+ for 78K0S/Kx1+”.
<2> Select [Ex-tool Bar] from the [View] menu and add the SM+ icon to the PM+ toolbar.
<3> Click the SM+ icon and start SM+.
(See the PM+ help for details on how to register external tools.)

(b) When not starting SM+ in PM+

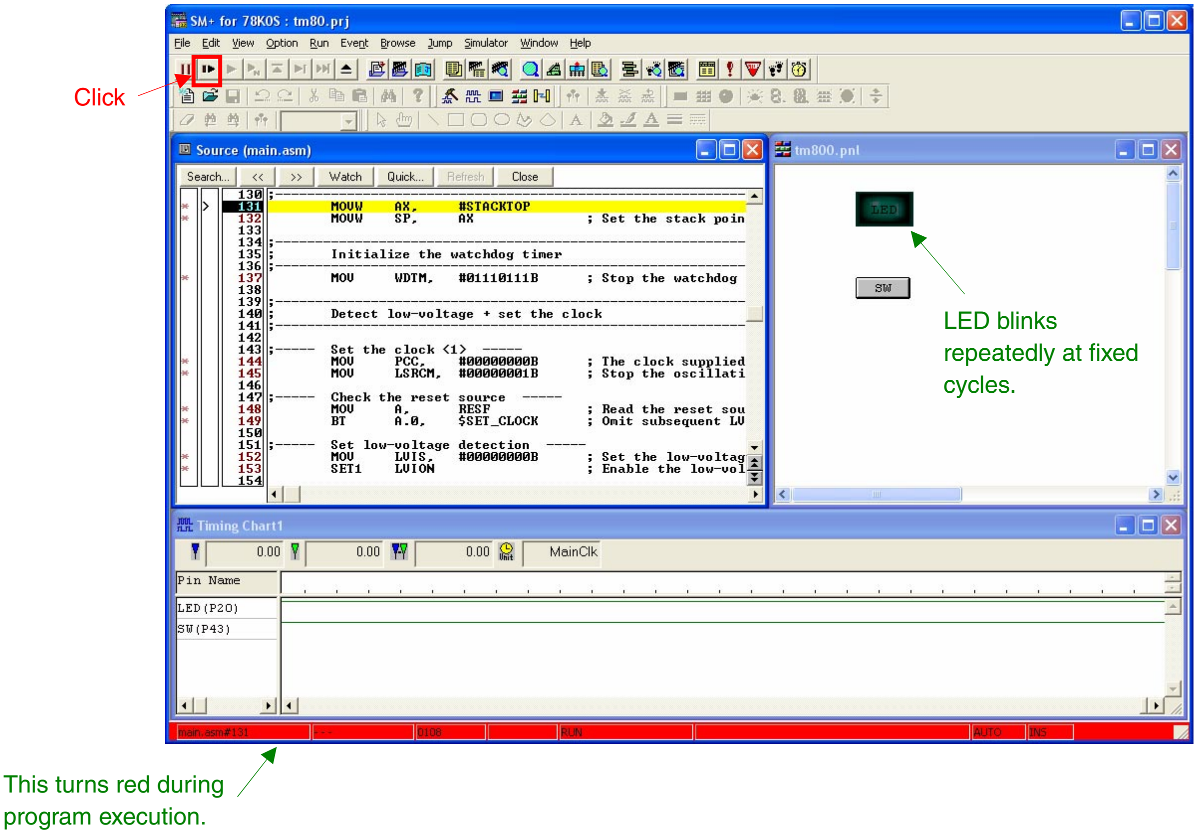
- Start SM+ from the Windows start menu.

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



```
;------------------------------------------------------------
        MOVW    AX,     #STACKTOP
        MOVW    SP,     AX              ; Set the stack poin

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

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

;-----  Set the clock <1>  -----
        MOV     PCC,    #00000000B      ; The clock supplied
        MOV     LSRCM,  #00000001B      ; Stop the oscillati

;-----  Check the reset source  -----
        MOV     A,      RESF            ; Read the reset sou
        BT      A.0,    $SET_CLOCK      ; Omit subsequent LV

;-----  Set low-voltage detection  -----
        MOV     LVIS,   #00000000B      ; Set the low-voltag
        SET1    LVION                   ; Enable the low-vol
```

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



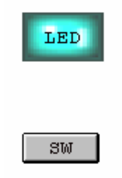
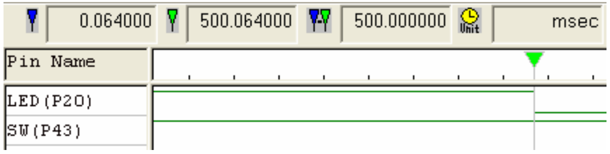
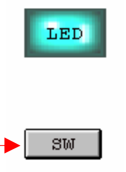
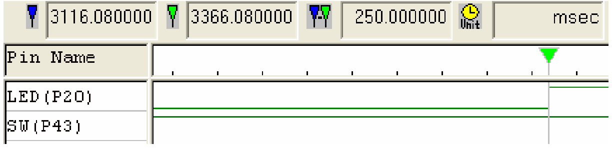
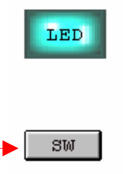
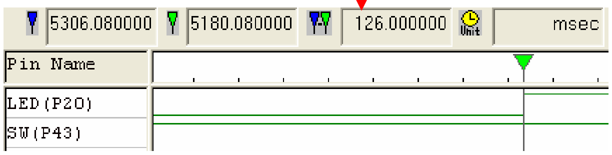
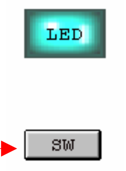
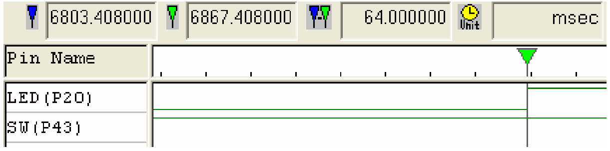
```
;------------------------------------------------------------
        MOVW    AX,     #STACKTOP
        MOVW    SP,     AX              ; Set the stack poin

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

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

;-----  Set the clock <1>  -----
        MOV     PCC,    #00000000B      ; The clock supplied
        MOV     LSRCM,  #00000001B      ; Stop the oscillati

;-----  Check the reset source  -----
        MOV     A,      RESF            ; Read the reset sou
        BT      A.0,    $SET_CLOCK      ; Omit subsequent LV

;-----  Set low-voltage detection  -----
        MOV     LVIS,   #00000000B      ; Set the low-voltag
        SET1    LVION                   ; Enable the low-vol
```

LED blinks repeatedly at fixed cycles.

This turns red during program execution.

(4) 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.

Note 1

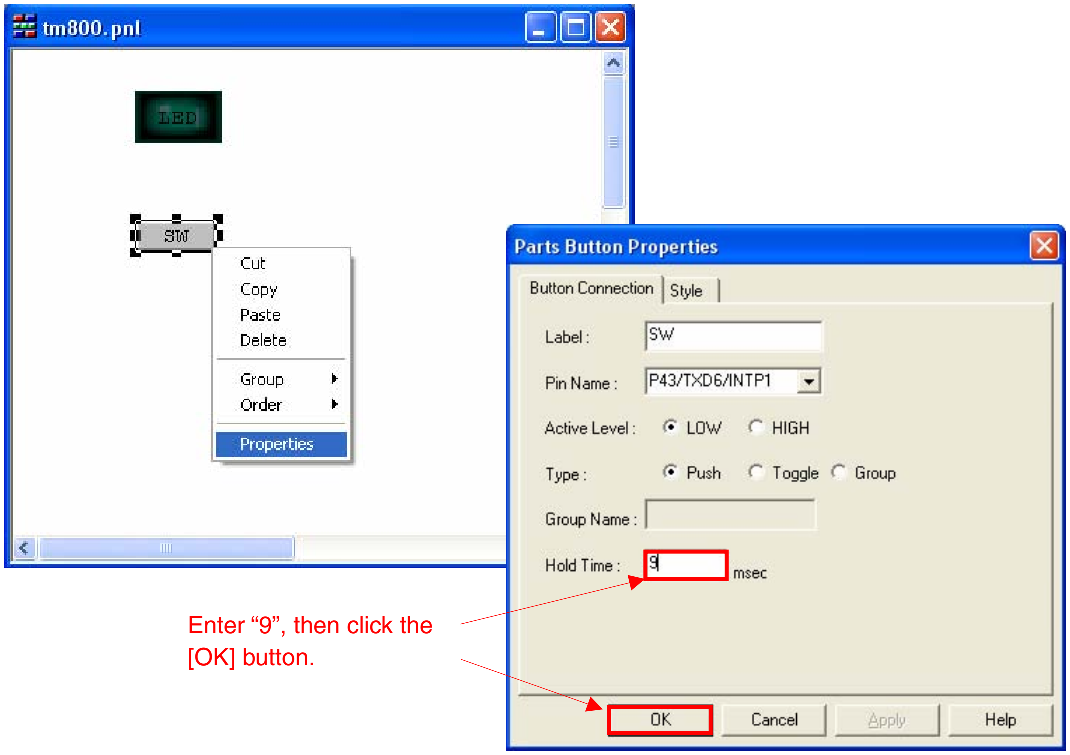
I/O panel window	Timing chart window
Blinks at cycles of about 1 s[Note 2]. Do not click.  	The reversal cycle of the LED (P20) is 500 ms.  0.064000 500.064000 500.000000 msec Pin Name LED(P20) SW(P43)
Blinks at cycles of about 1/2 s[Note 2]. Click once.  	The reversal cycle of the LED (P20) is 250 ms.  3116.080000 3366.080000 250.000000 msec Pin Name LED(P20) SW(P43)
Blinks at cycles of about 1/4 s[Note 2]. Click twice.  	The reversal cycle of the LED (P20) is 126 ms.  5306.080000 5180.080000 126.000000 msec Pin Name LED(P20) SW(P43)
Blinks at cycles of about 1/8 s[Note 2]. Click three times.  	The reversal cycle of the LED (P20) is 64 ms.  6803.408000 6867.408000 64.000000 msec Pin Name LED(P20) SW(P43)

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] 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.



Label : SW
Pin Name : P43/TXD6/INTP1
Active Level : ● LOW ○ HIGH
Type : ● Push ○ Toggle ○ Group
Group Name :
Hold Time : 9 msec

Enter “9”, then 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/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/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

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 80
;*****************************************************************************
;<<History>>
;     2007.7.--   Release
;*****************************************************************************
;
;<<Overview>>
;
;This sample program presents an example of using the interval timer function
;of 8-bit timer 80.  The LEDs are blinked by reversing the P20 pin output
;through the use of 8-bit timer 80 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 80 settings>
;  - Count clock = fxp/2^6 (125 kHz)
;  - Initial value of timer cycle = 2 ms (8[us/clk] 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    RESET_START       ;(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    INTERRUPT_TM80    ;(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 80 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_TM80:   DS    1           ; For counting INTTM80 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/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,   #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_TM80, #250    ; Initialize the number of INTTM80 interrupts
```

```
      MOVW  HL,   #0100H      ; Specify the table address to HL (used for
INTP1 interrupt)

;------------------------------------------------------------------------------
;     Set 8-bit timer 80
;------------------------------------------------------------------------------
      MOV   TMC80,      #00000000B  ; Count clock = fxp/2^6 = 125 kHz
      MOV   A,    [HL]        ; Read from the table the base time initial
values for blinking the LEDs
      MOV   CR80, A           ; Initialize the compare values
      SET1  TCE80             ; 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  TMMK80            ; Unmask INTTM80 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 80 cycle
WAIT_CHAT:
      NOP
      BF    TMIF80,     $WAIT_CHAT  ; Wait for the INTTM80 interrupt
      CLR1  TMIF80            ; Clear the INTTM80 interrupt request flag
      CALL  !SUB_INTERRUPT_TM80     ; Service the INTTM80 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 TM80 interval cycle  -----
      CLR1  TCE80             ; 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   CR80, A           ; Change the LED blinking base time

      SET1  TCE80             ; Start the timer operation

      MOV   CNT_TM80, #250    ; Initialize the number of INTTM80 interrupts

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

;*****************************************************************************
;
;     Interrupt INTTM80
;
;*****************************************************************************
INTERRUPT_TM80:
      CALL  !SUB_INTERRUPT_TM80     ; Service the INTTM80 interrupt
      RETI                    ; Return from interrupt servicing

;-----------------------------------------------------------------------------
;     Subroutine for measuring the number of INTTM80 interrupts
;-----------------------------------------------------------------------------
SUB_INTERRUPT_TM80:
      DBNZ  CNT_TM80, $END_INTTM80  ; Branch if the number of INTTM80
interrupts < 250
      MOV   CNT_TM80, #250    ; Initialize the number of INTTM80 interrupts

      XOR   P2,   #00000001B  ; Reverse the LED output

END_INTTM80:
      RET                     ; Return from the subroutine

end
```

- main.c (C language version)

```
/*******************************************************************************

      NEC Electronics     78K0S/KB1+

********************************************************************************
      78K0S/KB1+  Sample program
********************************************************************************
      8-bit timer 80
********************************************************************************
<<History>>
      2007.7.--   Release
********************************************************************************

<<Overview>>

This sample program presents an example of using the interval timer function
of 8-bit timer 80.  The LEDs are blinked by reversing the P20 pin output
through the use of 8-bit timer 80 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: INTTM80 -> fn_inttm80()
  - 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 80 settings>
  - Count clock = fxp/2^6 (125 kHz)
  - Initial value of timer cycle = 2 ms (8[us/clk] 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.

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


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

      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 INTTM80 fn_inttm80       /* Interrupt function
declaration:INTTM80 */

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

      Declare the function prototype

=============================================================================*/
void fn_subinttm80();                 /* INTTM80 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_ucTM80cnt = 0; /* 8-bit variable for counting the number
of INTTM80 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_ucCR80data[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 80
-----------------------------------------------------------------------------*/
      TMC80 = 0b00000000;           /* Count clock = fxp/2^6 = 125 kHz */
      CR80  = 250-1;                /* Initialize the LED blinking base time
*/
      TCE80 = 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 */
      TMMK80 = 0;                   /* Unmask INTTM80 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 (!TMIF80){  /* Wait for the INTTM80 interrupt request */
                  NOP();
            }

            TMIF80 = 0;             /* Clear the INTTM80 interrupt request
flag */
            fn_subinttm80();        /* Service the INTTM80 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 */

            TCE80 = 0;              /* Stop the timer operation */
            CR80 = g_ucCR80data[g_ucSWcnt];     /* Change the LED blinking
base time in accordance with the number of switch inputs */
            TCE80 = 1;              /* Start the timer operation */

            g_ucTM80cnt = 0;        /* Clear the number of INTTM80 interrupts
*/
      }

      return;
}

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

      Interrupt INTTM80

******************************************************************************/
__interrupt void fn_inttm80(){

      fn_subinttm80();        /* Service the INTTM80 interrupt */

      return;
}

/*-----------------------------------------------------------------------------
      Subroutine for measuring the number of INTTM80 interrupts
-----------------------------------------------------------------------------*/
void fn_subinttm80(){

      if (++g_ucTM80cnt == 250){    /* Processing when the number of INTTM80
interrupts is 250 */
            g_ucTM80cnt = 0;  /* Clear the number of INTTM80 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

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	October 2007	–	–
2nd edition	September 2008	pp.20 to 22	Modification of 5.1 Building the Sample Program
		p.22	5.2 Operation with SM+ • Addition of (1)
		p.26	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