Application Note

78K0S/Kx1+

Sample Program (Low-Voltage Detection)

Reset Generation During Detection at Less than 2.7 V

This document describes an operation overview of the sample program, as well as how to use the sample program and how to set and use the low-voltage detection function. In the sample program, an internal reset (LVI reset) signal is generated by detecting that V_{DD} is less than V_{LVI} (V_{LVI} = 2.85 V ±0.15 V). With an LVI reset, the LED display data immediately before the reset is retained and LED output is restored after reset release.

Target devices

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

CONTENTS

Document No. U18821EJ1V0AN00 (1st edition)
Date Published October 2007 N

- **The information in this document is current as of June, 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).

M8E 02. 11-1

CHAPTER 1 OVERVIEW

In this sample program, an example of using the low-voltage detection (LVI) function is presented.

An LVI circuit is set to generate an internal reset (LVI reset) signal by detecting that V_{DD} is less than V_{LVI} (V_{LVI} = 2.85 V ±0.15 V).

After completion of the initial settings, an LED lighting pattern is displayed according to the number of switch inputs, by detecting the falling edge of the switch input and performing interrupt servicing. (This processing is the same as that described in the Sample Program (Interrupt).)

When a reset is generated by other than LVI, the program is used to initialize the number of switch inputs. When an LVI reset is generated, the number of switch inputs immediately before reset generation is restored and an LED lighting pattern is displayed accordingly after LVI reset release, because RAM retains the data immediately before the reset, unless it falls below the POC voltage.

1.1 Main Contents of Initial Settings

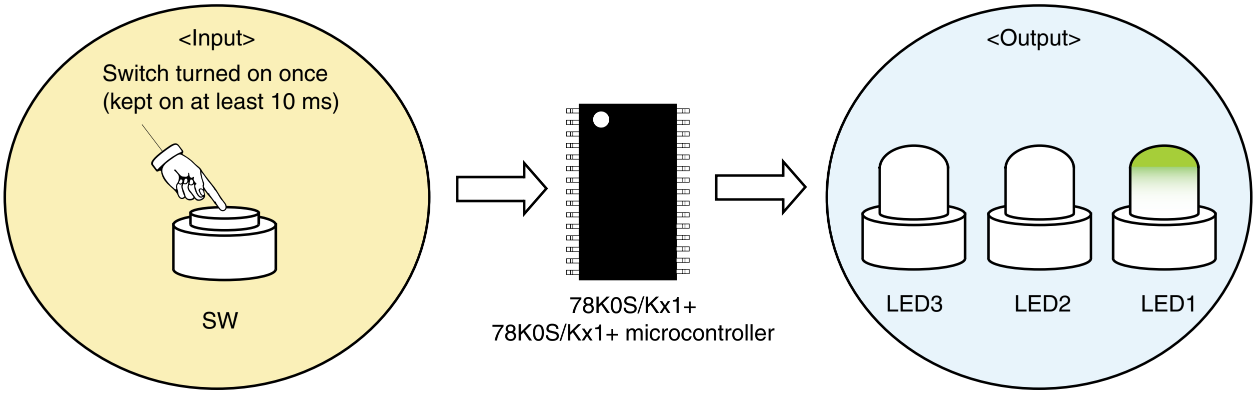
The contents of the initial settings are as follows.

- Selecting the high-speed internal oscillator as the system clock source[Note]
- Stopping watchdog timer operation
- Setting V_{LVI} (low-voltage detection voltage) to 2.85 V ±0.15 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 4 MHz
- Setting I/O ports
- Setting the valid edge of INTP1 (external interrupt) to the falling edge
- Enabling interrupt

Note This is set by using the option byte.

1.2 Contents Following the Main Loop

Interrupt servicing is performed by detecting the falling edge of the INTP1 pin, caused by switch input. In interrupt servicing, the LED lighting pattern is changed by confirming that the switch is on, after about 10 ms have elapsed after the falling edge of the INTP1 pin was detected. If the switch is off, after about 10 ms have elapsed, processing is identified as chattering and the LED lighting pattern is not changed.



Number of Switch Inputs[Note]	LED Output		
	LED3	LED2	LED1
0	OFF	OFF	OFF
1	OFF	OFF	ON
2	OFF	ON	OFF
3	OFF	ON	ON
4	ON	OFF	OFF
5	ON	OFF	ON
6	ON	ON	OFF
7	ON	ON	ON

Note The lighting patterns from the zeroth switch input are repeated after the eighth switch input.

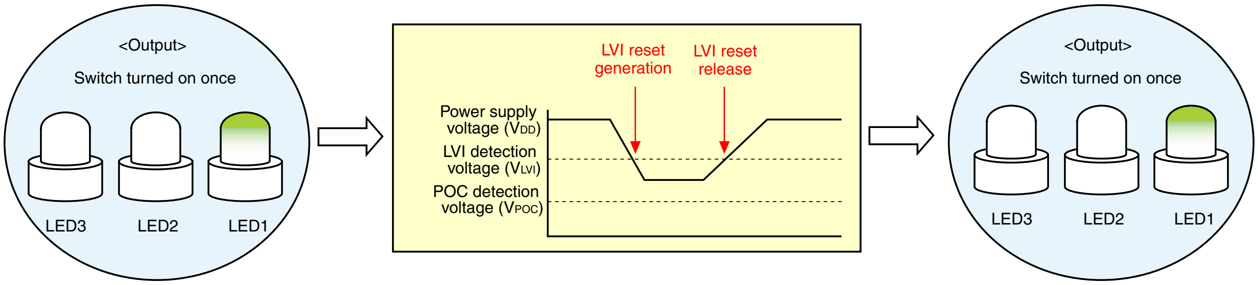
Caution For cautions when using the device, refer to the user's manual of each product (78K0S/KU1+, 78K0S/KY1+, 78K0S/KA1+, 78K0S/KB1+).

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

1.3 Content of the Low-Voltage Detection (LVI) Function

In this sample program, an internal reset (LVI reset) is generated by the low-voltage detection (LVI) function when V_{DD} becomes less than V_{LVI}. At this time, register values are initialized, but RAM retains the data immediately before the reset, unless it falls below the POC voltage. The number of switch inputs immediately before the reset is retained, the number of switch inputs is restored after reset release, and an LED lighting pattern can therefore be displayed accordingly when an LVI reset is generated[Note]. When a reset is generated by other than LVI, the program is used to initialize the number of switch inputs and all LEDs are turned off.

Note As mentioned in [Column] below, when a standard startup routine is used in a C language program, RAM data is initialized before the main function. To avoid this, a section of the standard startup routine is commented out in this C language version sample program, so that RAM data without initial values is not initialized (cleared to 0).



[Column] Processing of the startup routine

A standard startup routine mainly performs the following processing.

- Stack pointer setting
- Hardware initialization (needed to be performed at an early stage)
- Initialization of variables to be used with a library
- Transferring from ROM to RAM the initial values of external variables with initial values, and sreg variables
- Assigning 0 to RAM of external variables without initial values, and sreg variables[Note]

  Note This processing is commented out in the source file (cstart.asm) of the startup routine included in this C language version sample program.

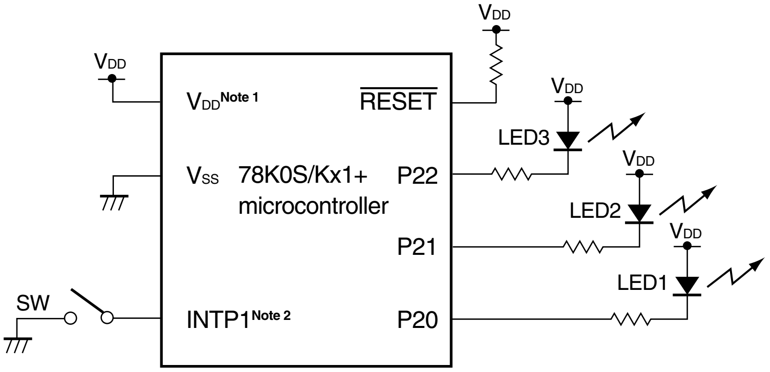
For details, refer to the chapter regarding the startup routine of the **CC78K0S C Compiler Operation User's Manual**.

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 3.0 V ≤ V_{DD} ≤ 5.5 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) LEDs (LED1, LED2, LED3)

The LEDs are used as outputs corresponding to 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.

(1) Assembly language version

File Name	Description	Compressed (*.zip) File Included	
		ZIP	PM 32
main.asm	Source file for hardware initialization processing and main processing of microcontroller	●	●
op.asm	Assembler source file for setting the option byte (sets the system clock source)	●	●
lvi.prw	Work space file for integrated development environment PM+		●
lvi.prj	Project file for integrated development environment PM+		●

(2) C language version

File Name	Description	Compressed (*.zip) File Included	
		ZIP	PM 32
main.c	Source file for hardware initialization processing and main processing of microcontroller	●	●
op.asm	Assembler source file for setting the option byte (sets the system clock source)	●	●
cstart.asm	Startup routine source file (comments out a section of ROM processing)	●	●
def.inc	Library type setting file (include file of “cstart.asm”)	●	●
macro.inc	Macro definition file regarding various template patterns (include file of “cstart.asm”)	●	●
lvi.prw	Work space file for integrated development environment PM+		●
lvi.prj	Project file for integrated development environment PM+		●

Remark ZIP : Only the source files are included.

PM 32 : The files to be used with integrated development environment PM+ are included.

3.2 Internal Peripheral Functions to Be Used

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

• $V_{DD} < V_{LVI}$ detection:	Low-voltage detector (LVI)
• Switch input:	INTP1[Note] (external interrupt)
• LED outputs:	P20, P21, P22 (output ports)

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

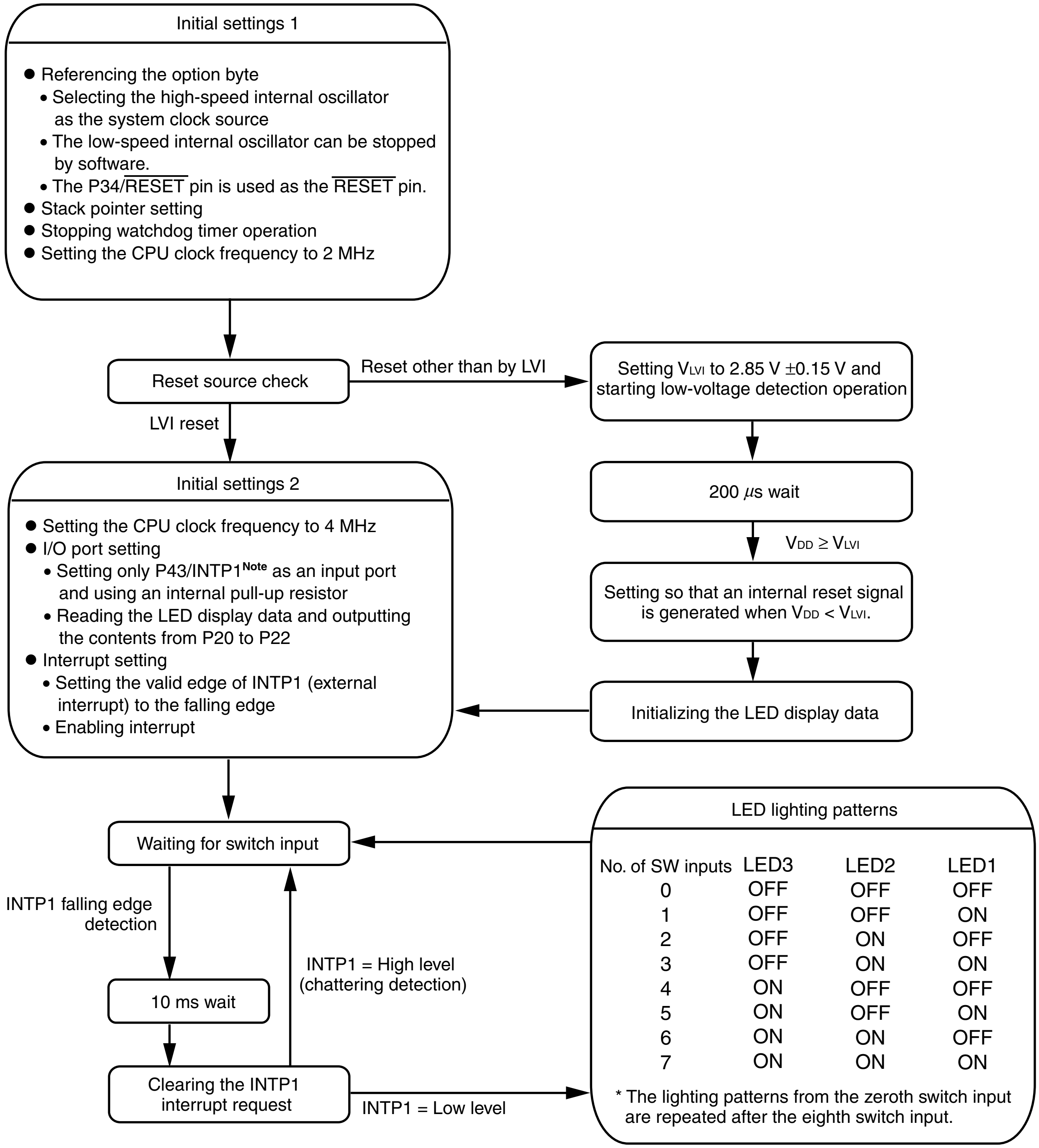
3.3 Initial Settings and Operation Overview

In this sample program, the selection of the clock frequency, setting of the I/O ports, setting of interrupt, setting of LVI, and the like are performed in the initial settings.

After completion of the initial settings, interrupt servicing is performed by detecting the falling edge of the switch input (SW) and the lighting of the three LEDs (LED1, LED2, and LED3) is controlled according to the number of switch inputs. (This processing is the same as that described in the Sample Program (Interrupt).)

When a reset is generated by other than LVI, the program is used to initialize the number of switch inputs. When an LVI reset is generated, the number of switch inputs immediately before reset generation is restored and an LED lighting pattern is displayed accordingly after LVI reset release, because RAM retains the data immediately before the reset, unless it falls below the POC voltage.

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



LED lighting patterns

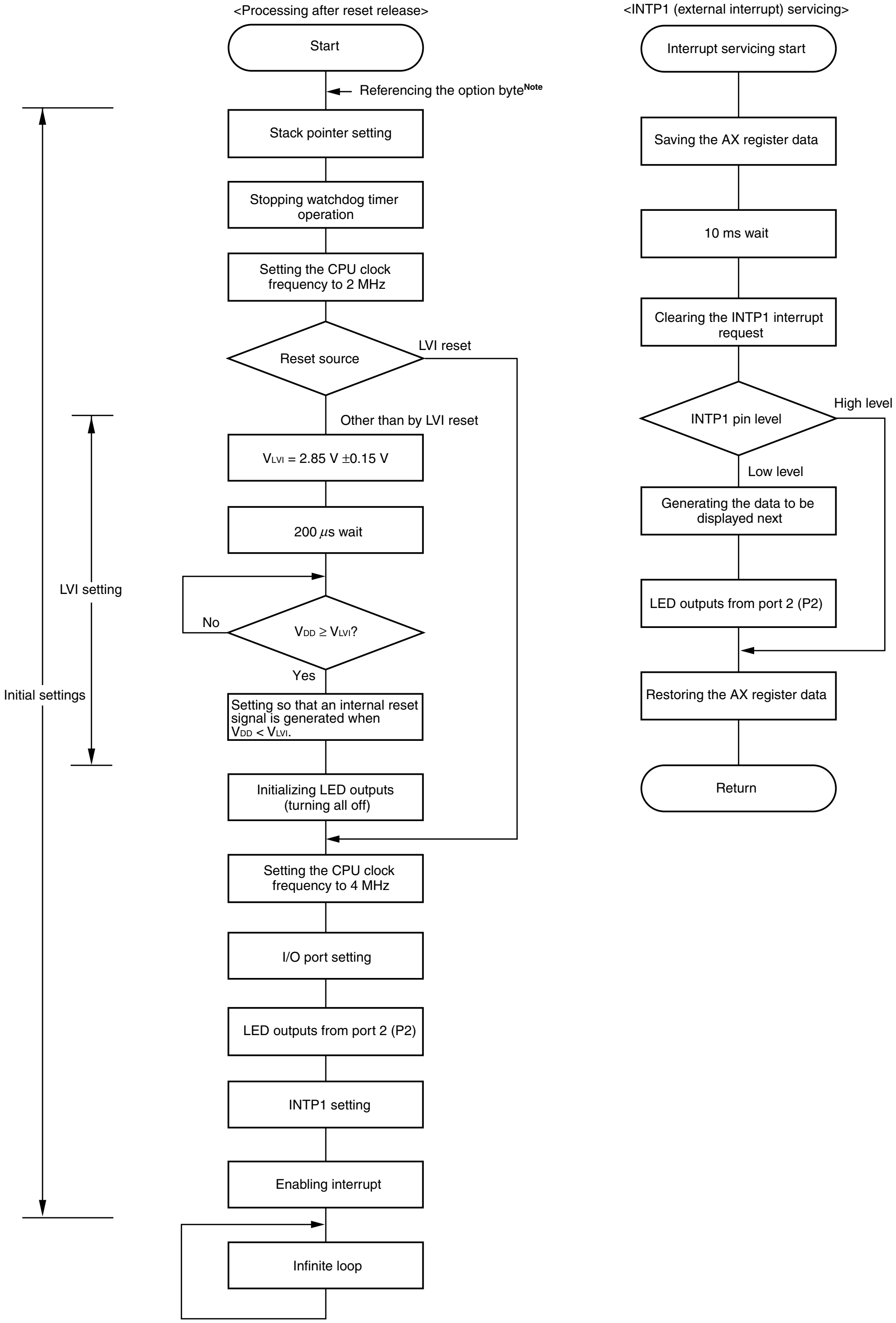
No. of SW inputs	LED3	LED2	LED1
0	OFF	OFF	OFF
1	OFF	OFF	ON
2	OFF	ON	OFF
3	OFF	ON	ON
4	ON	OFF	OFF
5	ON	OFF	ON
6	ON	ON	OFF
7	ON	ON	ON

* The lighting patterns from the zeroth switch input are repeated after the eighth switch input.

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

3.4 Flow Chart

A flow chart for the sample program is 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 low-voltage detection function.

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 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 Low-Voltage Detection (LVI) Function Setting

The low-voltage detection function has the following two types of operation modes.

- Using it as a reset (see **[Example 1]**)
- Using it as an interrupt (see **[Example 2]**)

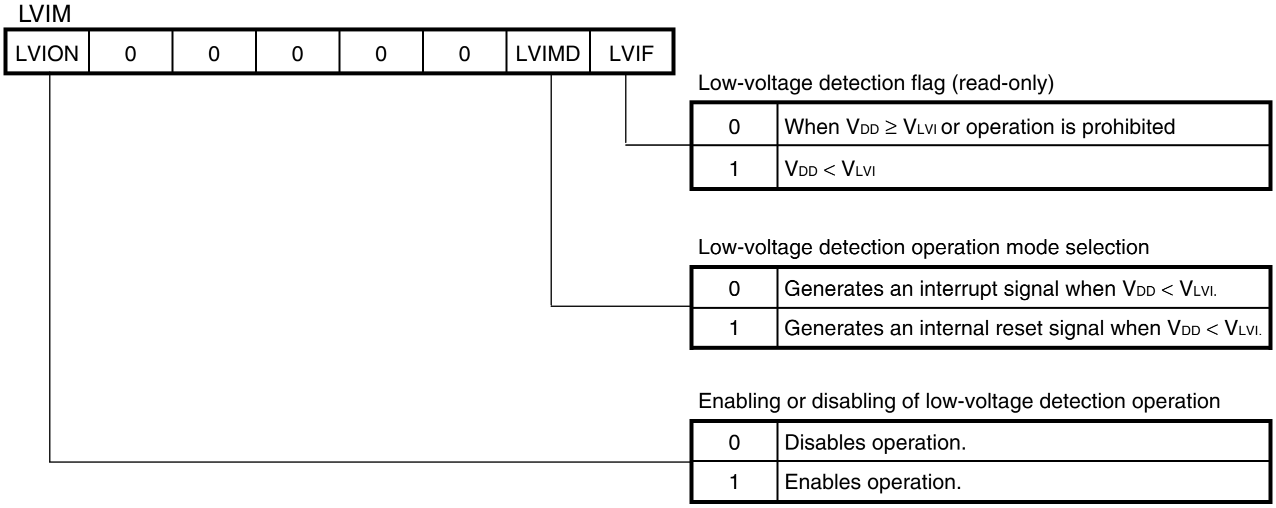
The low-voltage detection function is mainly controlled by the following two types of registers.

- Low-voltage detection register (LVIM)
- Low-voltage detection level select register (LVIS)

(1) Settings regarding low-voltage detection operation

The low-voltage detection register (LVIM) is used to set the low-voltage detection operation mode and control the operation.

Figure 4-1. Format of Low-Voltage Detection Register (LVIM)



LVIM

LVION	0	0	0	0	0	LVIMD	LVIF

Low-voltage detection flag (read-only)

0	When $V_{DD} \geq V_{LVI}$ or operation is prohibited
1	$V_{DD} < V_{LVI}$

Low-voltage detection operation mode selection

0	Generates an interrupt signal when $V_{DD} < V_{LVI}$.
1	Generates an internal reset signal when $V_{DD} < V_{LVI}$.

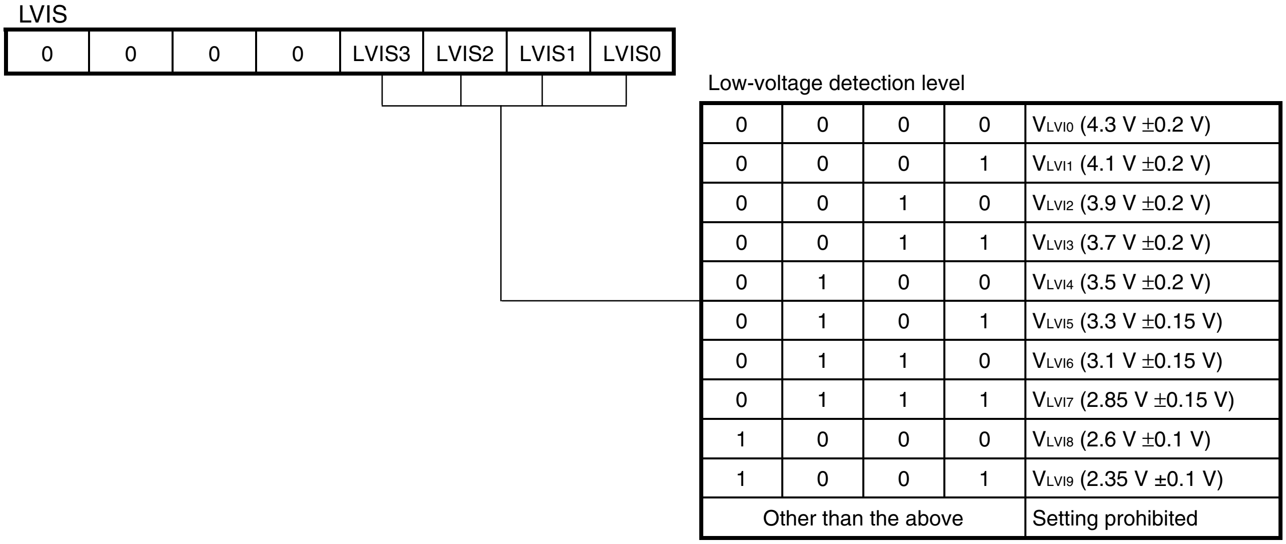
Enabling or disabling of low-voltage detection operation

0	Disables operation.
1	Enables operation.

(2) Settings regarding the low-voltage detection level

The low-voltage detection level select register (LVIS) is used to set the low-voltage detection level.

Figure 4-2. Format of Low-Voltage Detection Level Select Register (LVIS)



LVIS

0	0	0	0	LVIS3	LVIS2	LVIS1	LVIS0

Low-voltage detection level

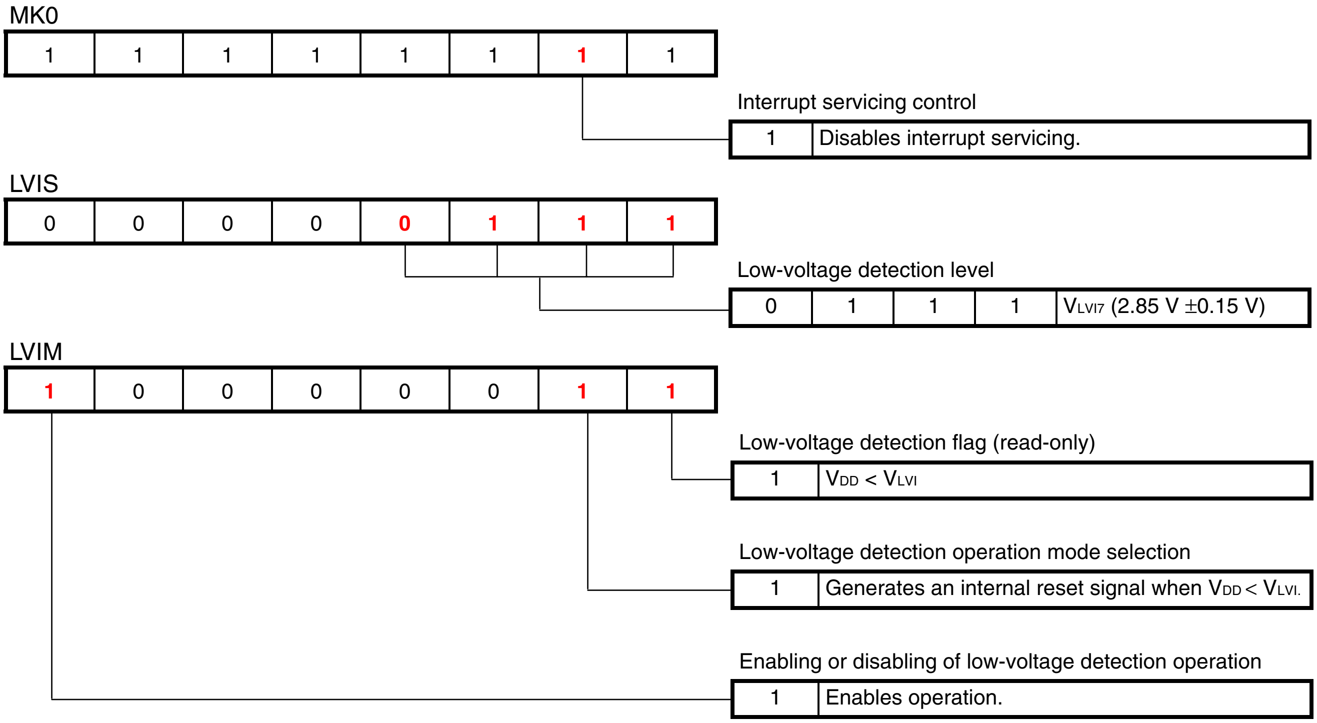
0	0	0	0	V_{LVI0} (4.3 V ±0.2 V)
0	0	0	1	V_{LVI1} (4.1 V ±0.2 V)
0	0	1	0	V_{LVI2} (3.9 V ±0.2 V)
0	0	1	1	V_{LVI3} (3.7 V ±0.2 V)
0	1	0	0	V_{LVI4} (3.5 V ±0.2 V)
0	1	0	1	V_{LVI5} (3.3 V ±0.15 V)
0	1	1	0	V_{LVI6} (3.1 V ±0.15 V)
0	1	1	1	V_{LVI7} (2.85 V ±0.15 V)
1	0	0	0	V_{LVI8} (2.6 V ±0.1 V)
1	0	0	1	V_{LVI9} (2.35 V ±0.1 V)
Other than the above				Setting prohibited

[Example 1] Using the low-voltage detection function as a reset by setting the low-voltage detection level (V_{LVI}) to 2.85 V ±0.15 V (same content as the sample program)

- Setting procedure
  - <1> Mask the LVI interrupt (LVIMK = 1)[Note].
  - <2> Set the detection level by using bits 3 to 0 (LVIS3 to LVIS0) of the LVIS register.
  - <3> Enable LVI operation by setting bit 7 (LVION) of the LVIM register to 1.
  - <4> Use software to wait at least 200 μs.
  - <5> Use bit 0 (LVIF) of the LVIM register to wait until "$V_{DD} \geq V_{LVI}$ (LVIF = 0)" can be confirmed.
  - <6> Set bit 1 (LVIMD) of the LVIM register to 1, so that an internal reset signal is generated when LVI is detected.

Note The settings for the sample program and the program example on the next page are omitted, because the LVI interrupt is masked after the reset.

Remark <2> to <6> mentioned above correspond to <2> to <6> on the next page.



MK0

1	1	1	1	1	1	1	1

Interrupt servicing control

1	Disables interrupt servicing.

LVIS

0	0	0	0	0	1	1	1

Low-voltage detection level

0	1	1	1	V_{LVI7} (2.85 V ±0.15 V)

LVIM

1	0	0	0	0	0	1	1

Low-voltage detection flag (read-only)

1	$V_{DD} < V_{LVI}$

Low-voltage detection operation mode selection

1	Generates an internal reset signal when $V_{DD} < V_{LVI}$.

Enabling or disabling of low-voltage detection operation

1	Enables operation.

- Assembly language program example (same content as the sample program)

```
      XMAIN  CSEG   UNIT
      RESET_START:
<2>          MOV    LVIS,  #00000111B     ; Set the low-voltage detection level (VLVI) to 2.85 V +-0.15 V
<3>          SET1   LVION                 ; Enable the low-voltage detector operation

             MOV    A,     #40            ; Assign the 200 us wait count value
<4>   WAIT_200US:
             DEC    A
             BNZ    $WAIT_200US           ; 0.5[us/clk] × 10[clk] × 40[count] = 200[us]
      WAIT_LVI:
             NOP
<5>          BT     LVIF,  $WAIT_LVI      ; Branch if VDD < VLVI

<6>          SET1   LVIMD                 ; Set so that an internal reset signal is generated when VDD < VLVI
```

- C language program example (same content as the sample program)

```
      void hdwinit(void){
             unsigned char ucCnt200us;  /* 8-bit variable for 200 us wait */
<2>                 LVIS   = 0b00000111; /* Set the low-voltage detection level (VLVI) to 2.85 V +-0.15 V
      */
<3>                 LVION = 1;           /* Enable the low-voltage detector operation */

<4>                 for (ucCnt200us = 0; ucCnt200us < 9; ucCnt200us++){ /* Wait of about 200 us */
                           NOP();
                    }

<5>                 while (LVIF){        /* Wait for VDD >= VLVI */
                           NOP();
                    }

<6>                 LVIMD = 1;           /* Set so that an internal reset signal is generated when VDD < VLVI */
      }
```

Remarks 1. As in the sample program, the above-mentioned wait time (200 μs) is calculated with f_{CPU} (CPU clock frequency) being 2 MHz.

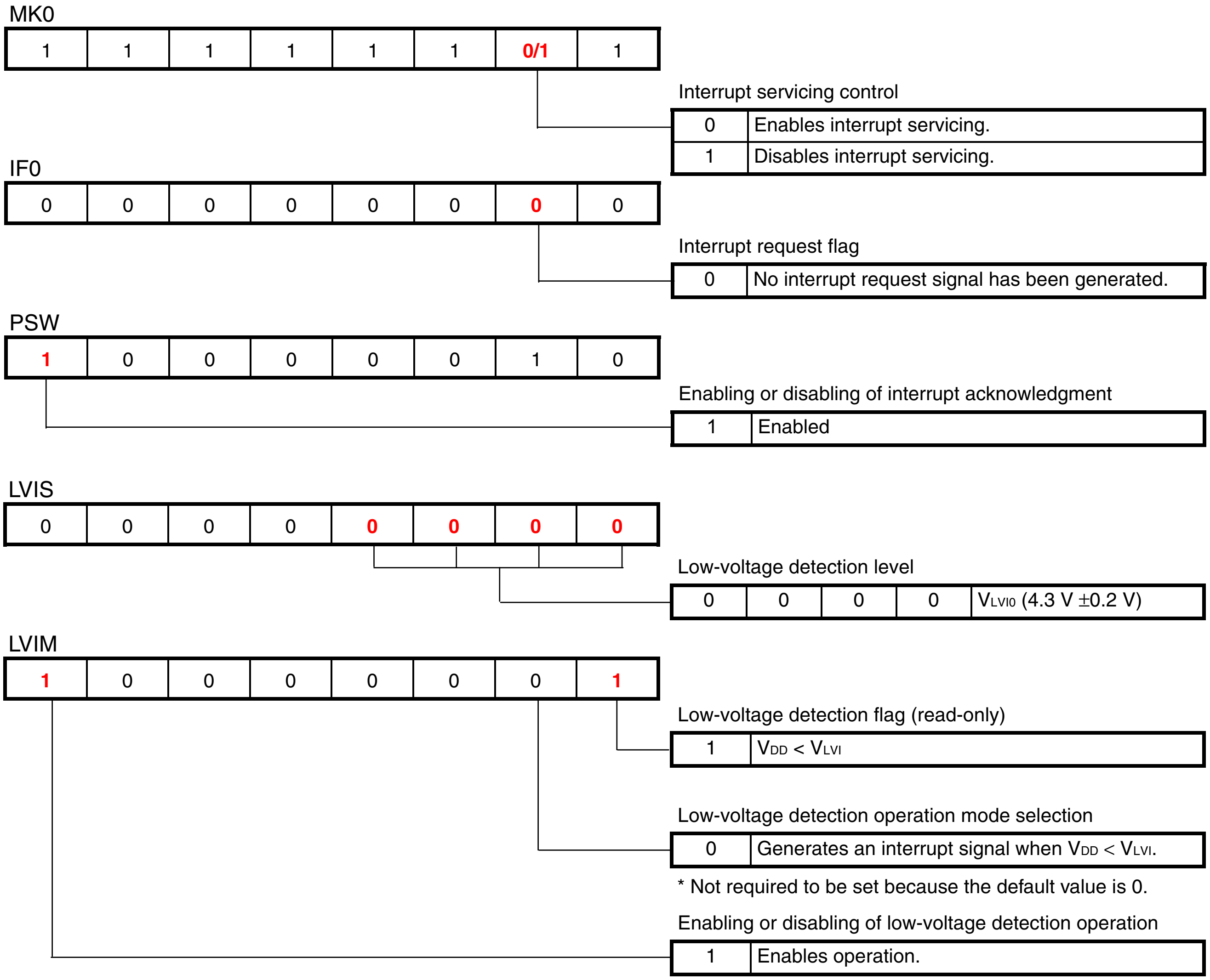
2. <2> to <6> mentioned above correspond to <2> to <6> on the previous page.

[Example 2] Using the low-voltage detection function as an interrupt by setting the low-voltage detection level (V_{LVI}) to 4.3 V ±0.2 V.

- Setting procedure
  - <1> Mask the LVI interrupt (LVIMK = 1)[Note].
  - <2> Set the detection level by using bits 3 to 0 (LVIS3 to LVIS0) of the LVIS register.
  - <3> Enable LVI operation by setting bit 7 (LVION) of the LVIM register to 1.
  - <4> Use software to wait at least 200 µs.
  - <5> Use bit 0 (LVIF) of the LVIM register to wait until "$V_{DD} \geq V_{LVI}$ (LVIF = 0)" can be confirmed.
  - <6> Clear the interrupt request flag of LVI (LVIIF = 0).
  - <7> Release the interrupt mask flag of LVI (LVIMK = 0).
  - <8> Execute the EI instruction (when using vector interrupts).

Note The settings for the program example shown below are omitted, because the LVI interrupt is masked after the reset.

Remark <2> to <8> mentioned above correspond to <2> to <8> on the next page and the page after next.



MK0

1	1	1	1	1	1	0/1	1

Interrupt servicing control

0	Enables interrupt servicing.
1	Disables interrupt servicing.

IF0

0	0	0	0	0	0	0	0

Interrupt request flag

0	No interrupt request signal has been generated.

PSW

1	0	0	0	0	0	1	0

Enabling or disabling of interrupt acknowledgment

1	Enabled

LVIS

0	0	0	0	0	0	0	0

Low-voltage detection level

0	0	0	0	V_{LVI0} (4.3 V ±0.2 V)

LVIM

1	0	0	0	0	0	0	1

Low-voltage detection flag (read-only)

1	$V_{DD} < V_{LVI}$

Low-voltage detection operation mode selection

0	Generates an interrupt signal when $V_{DD} < V_{LVI}$.

* Not required to be set because the default value is 0.

Enabling or disabling of low-voltage detection operation

1	Enables operation.

- Assembly language program example (setting the CPU clock frequency to low speed by using a low-voltage detection interrupt)

```
      HIGH_SPEED   EQU    00H
      LOW_SPEED    EQU    02H

      XVCT   CSEG   AT     0000H
             DW     RESET_START   ;(00)  RESET
             DW     RESET_START   ;(02)  --
             DW     RESET_START   ;(04)  --
             DW     INTERRUPT_LVI ;(06)  INTLVI

      XMAIN  CSEG   UNIT
      RESET_START:
             MOV    PCC,   #00000000B     ; The clock supplied to the CPU (fcpu) = fxp (= fx/4 = 2 MHz)

<2>          MOV    LVIS,  #00000000B     ; Set the low-voltage detection level (VLVI) to 4.3 V +-0.2 V
<3>          SET1   LVION                 ; Enable the low-voltage detector operation

             MOV    A,     #40            ; Assign the 200 us wait count value
<4>   WAIT_200US:
             DEC    A
             BNZ    $WAIT_200US           ; 0.5[us/clk] x 10[clk] x 40[count] = 200[us]

      WAIT_LVI1:
             NOP
<5>          BT     LVIF,  $WAIT_LVI1     ; Branch if VDD < VLVI
             MOV    PPCC,  #HIGH_SPEED    ; The clock supplied to the peripheral hardware (fxp) = fx (= 8 MHz)
                                          ; -> The clock supplied to the CPU (fcpu) = fxp = 8 MHz

<6>          CLR1   LVIIF                 ; Clear the LVI interrupt request flag
<7>          CLR1   LVIMK                 ; Release the LVI interrupt request mask flag
<8>          EI                           ; Enable vector interrupt

      MAIN_LOOP:
             MOV    A,     PPCC
             CMP    A,     #HIGH_SPEED
             BZ     $MAIN_LOOP            ; Go to the MAIN_LOOP if the CPU clock (fcpu) is 8 MHz

      WAIT_LVI2:
             NOP
             BT     LVIF,  $WAIT_LVI2     ; Branch if VDD < VLVI
             MOV    PPCC,  #HIGH_SPEED    ; The clock supplied to the peripheral hardware (fxp) = fx (= 8 MHz)
                                          ; -> The clock supplied to the CPU (fcpu) = fxp = 8 MHz
             BR     $MAIN_LOOP            ; Go to the MAIN_LOOP

      INTERRUPT_LVI:
             MOV    PPCC,  #LOW_SPEED     ; The clock supplied to the peripheral hardware (fxp) = fx/4 (= 2 MHz)
                                          ; -> The clock supplied to the CPU (fcpu) = fxp = 2 MHz
             RETI                         ; Return from interrupt servicing
```

Interrupt servicing is started from the “INTERRUPT_LVI” symbol when an interrupt is generated by LVI detection, by describing the initial value (“INTERRUPT_LVI”) using the DW pseudo instruction.

The CPU clock frequency is set.

The CPU clock frequency is set to high speed when $V_{DD} \geq V_{LVI}$ after booting the power supply.

The CPU clock frequency is set to high speed when $V_{DD} \geq V_{LVI}$ after LVI detection.

Interrupt servicing is started from “INTERRUPT_LVI” by an interrupt generated by LVI detection.

The CPU clock frequency is set to low speed when $V_{DD} < V_{LVI}$.

Remarks 1. The above-mentioned wait time (200 μs) and CPU clock frequency are calculated with fx (system clock oscillation frequency) being 8 MHz.

2. <2> to <8> mentioned above correspond to <2> to <8> on the previous page.

- C language program example (setting the CPU clock frequency to low speed by using a low-voltage detection interrupt)

```
#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 INTLVI fn_intlvi            /* Interrupt function declaration: INTLVI  */

#define      HighSpeed 0x00
#define      LowSpeed 0x02

void hdwinit(void){
        PCC   = 0b00000000;  /* The clock supplied to the CPU (fcpu) = fxp (= fx/4 = 2 MHz)  */

<2>     LVIS  = 0b00000000;  /* Set the low-voltage detection level (VLVI) to 4.3 V +-0.2 V  */
<3>     LVION = 1;           /* Enable the low-voltage detector operation  */

<4>     for (ucCnt200us = 0; ucCnt200us < 9; ucCnt200us++){   /* Wait of about 200 us  */
                NOP();
        }

<5>     while (LVIF){        /* Wait for VDD >= VLVI  */
                NOP();
        }

        PPCC  = HighSpeed;   /* The clock supplied to the peripheral hardware (fxp) = fx (= 8 MHz)
                                -> The clock supplied to the CPU (fcpu) = fxp = 8 MHz  */

<6>     LVIIF = 0;           /* Clear the LVI interrupt request flag  */
<7>     LVIMK = 0;           /* Release the LVI interrupt request mask flag  */
<8>     EI();                /* Enable vector interrupt  */

        return;
}

void main(void){
        while (1){
                while (PPCC == HighSpeed){ /* Wait for LVI interrupt  */
                        NOP();
                }

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

                PPCC  = HighSpeed;   /* The clock supplied to the peripheral hardware (fxp) = fx (= 8 MHz)
                                        -> The clock supplied to the CPU (fcpu) = fxp = 8 MHz  */
        }
}

__interrupt void fn_intlvi(){
        PPCC  = LowSpeed;            /* The clock supplied to the peripheral hardware (fxp) = fx/4 (= 2 MHz)
                                        -> The clock supplied to the CPU (fcpu) = fxp = 2 MHz  */

        return;
}
```

Interrupt servicing is started from the interrupt function declared with the _interrupt modifier when an interrupt is generated, by declaring the INTLVI interrupt function (“fn_intlvi” in this example) in the preprocessing directive (#pragma directive) and declaring that interrupt function using the _interrupt modifier.

The CPU clock frequency is set.

The CPU clock frequency is set to high speed when $V_{DD} \geq V_{LVI}$ after booting the power supply.

The CPU clock frequency is set to high speed when $V_{DD} \geq V_{LVI}$ after LVI detection.

Interrupt servicing is started from “fn_intlvi” by an interrupt generated by LVI detection.

The CPU clock frequency is set to low speed when $V_{DD} < V_{LVI}$.

Remarks 1. The above-mentioned wait time (200 μs) and CPU clock frequency are calculated with fx (system clock oscillation frequency) being 8 MHz.

2. <2> to <8> mentioned above correspond to <2> to <8> on page 15.

CHAPTER 5 OPERATION CHECK USING THE DEVICE

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

5.1 Building the Sample Program

This section describes how to build sample programs, using the assembly language version sample program (source files + project file) downloaded by clicking the icon and the C language version sample program (only source files) downloaded by clicking the icon. 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**.

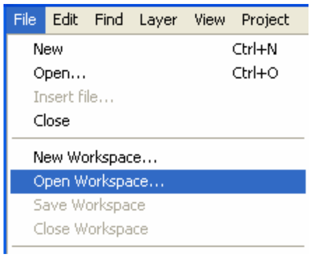
For the details of how to operate PM+, refer to the **PM+ Project Manager User's Manual**.

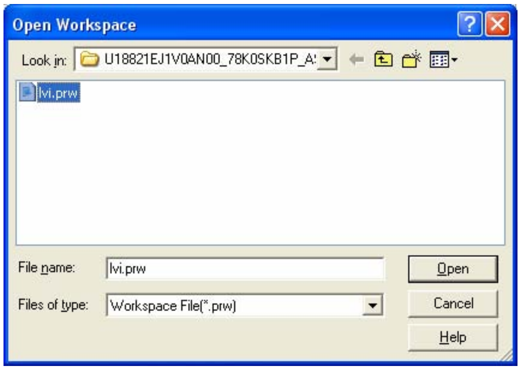
5.1.1 Assembly language version (source files + project file)

This section describes how to build a sample program, using the assembly language version file of the 78K0S/KB1+ microcontroller sample program (low-voltage detection) downloaded by clicking the icon.

(1) Start PM+.

(2) Select "lvi.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.





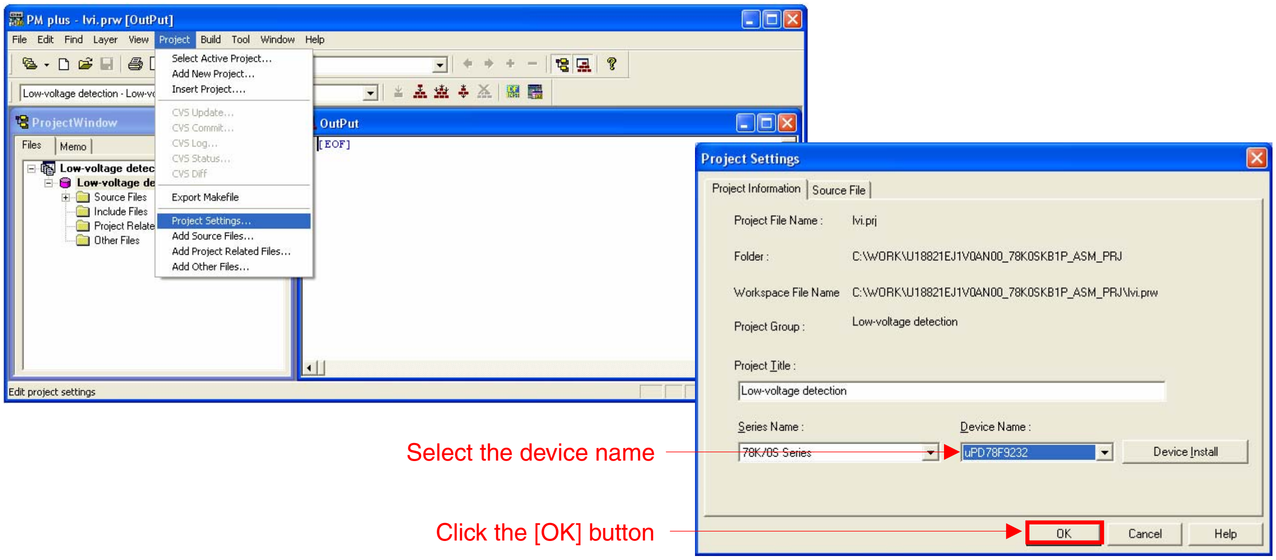
Look in: U18821EJ1V0AN00_78K0SKB1P_A

lvi.prw

File name: lvi.prw

Files of type: Workspace File(*.prw)

(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].



Project File Name : lvi.prj

Folder : C:\WORK\U18821EJ1V0AN00_78K0SKB1P_ASM_PRJ

Workspace File Name C:\WORK\U18821EJ1V0AN00_78K0SKB1P_ASM_PRJ\lvi.prw

Project Group : Low-voltage detection

Project Title : Low-voltage detection

Series Name : 78K/0S Series

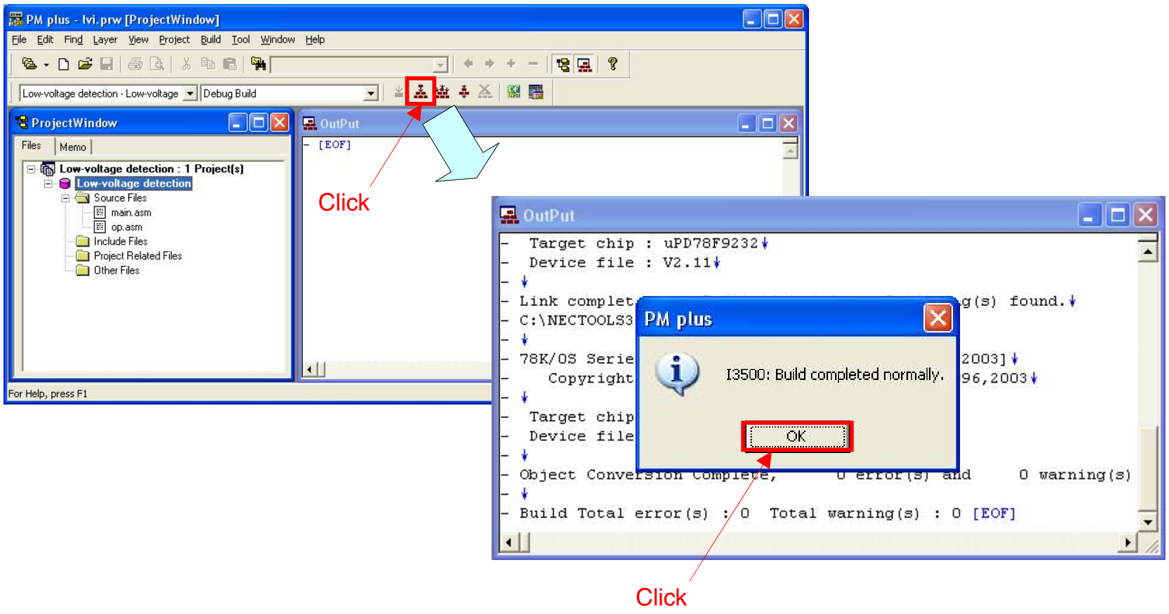
Device Name : uPD78F9232

Select the device name

Click the [OK] button

(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 window. A HEX file for flash memory writing will be created.



```
[EOF]
```

```
- Target chip : uPD78F9232
- Device file : V2.11
-
- Link complet                     g(s) found.
- C:\NECTOOLS3
-
- 78K/0S Serie                     2003]
-    Copyright                     96,2003
-
- Target chip
- Device file
-
- Object Conversion Complete,    0 error(s) and    0 warning(s)
-
- Build Total error(s) : 0  Total warning(s) : 0 [EOF]
```

I3500: Build completed normally.

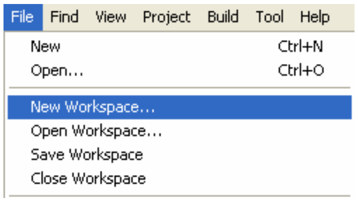
A HEX file for flash memory writing will be generated. → Go to 5.2.

5.1.2 C language version (only source files)

This section describes how to build a sample program, using the C language version file of the 78K0S/KB1+ microcontroller sample program (low-voltage detection) downloaded by clicking the icon.

The C language version of this sample program includes the startup routine source file “cstart.asm”, library type setting file “def.inc”, and macro definition file “macro.inc”, besides “main.c” and “op.asm”. These files correspond to sections commented out from the standard startup routine. To create a PM+ project by applying these files, operations different from those for other C language version sample programs are required. The procedure is described below.

- Project registration
  1. (1) Start PM+.
  2. (2) Select [New Workspace] from the [File] menu.



(3) The [New WorkSpace - Step 1/8 [Workspace Information]] dialog box will be displayed. Set the following items.

(a) Workspace File Name (“lvi” is entered as the file name in this example.)

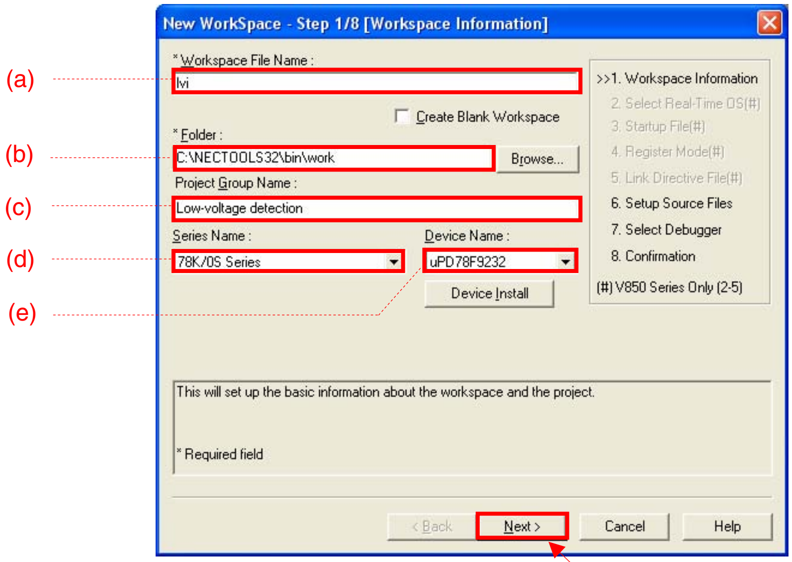
(b) Folder (An arbitrarily created “work” folder located under the default folder (“bin” folder in which PM+ exists) is specified in this example.)

(c) Project Group Name (“Low-voltage detection” is entered as the group name in this example.)

(d) Series Name (“78K/0S Series” is selected as the series name in this example.)

(e) Device Name (The 78K0S/KB1+ microcontroller product “uPD78F9232” is set in this example.)

After setting items (a) to (e), click the [Next] button.



New WorkSpace - Step 1/8 [Workspace Information]

(a) * Workspace File Name : lvi

☐ Create Blank Workspace

(b) * Folder : C:\NECTOOLS32\bin\work

(c) Project Group Name : Low-voltage detection

(d) Series Name : 78K/0S Series

(e) Device Name : uPD78F9232

This will set up the basic information about the workspace and the project.

* Required field

Click after setting (a) to (e)

(4) The [New WorkSpace - Step 6/8 [Setup Source Files]] dialog box will be displayed. Set the source files in the order of the following procedure.

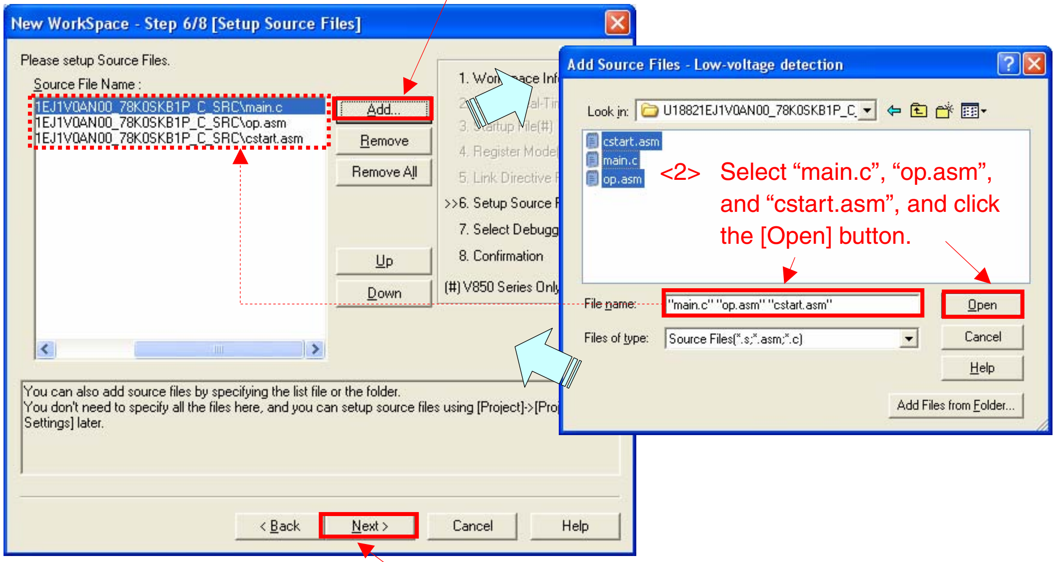
<1> Click the [Add] button.

<2> The [Add Source Files] dialog box will be displayed. Select the following source files and click the [Open] button.

- main.c
- op.asm
- cstart.asm

<3> The source files selected in step <2> will be set. Click the [Next] button.

<1> Click the [Add] button.



New WorkSpace - Step 6/8 [Setup Source Files]

Please setup Source Files.

Source File Name :

1EJ1V0AN00_78K0SKB1P_C_SRC\main.c
1EJ1V0AN00_78K0SKB1P_C_SRC\op.asm
1EJ1V0AN00_78K0SKB1P_C_SRC\cstart.asm

You can also add source files by specifying the list file or the folder.
You don't need to specify all the files here, and you can setup source files using [Project]->[Pro
Settings] later.

Add Source Files - Low-voltage detection

Look in: U18821EJ1V0AN00_78K0SKB1P_C_

cstart.asm
main.c
op.asm

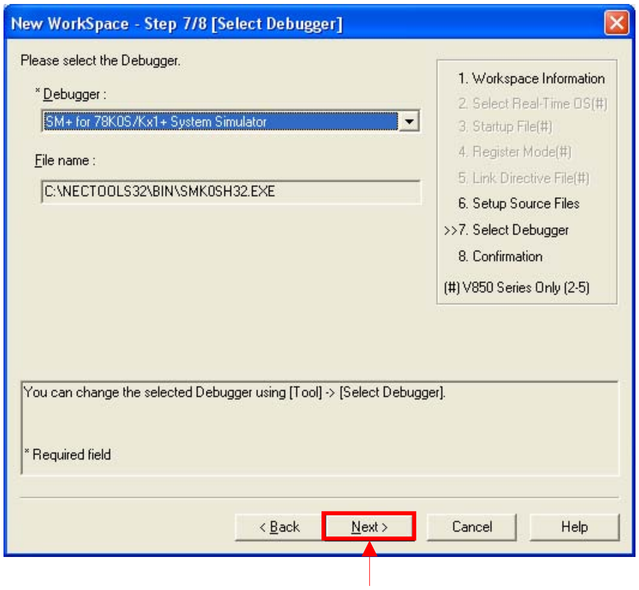
File name: "main.c" "op.asm" "cstart.asm"

Files of type: Source Files(*.s;*.asm;*.c)

<2> Select "main.c", "op.asm", and "cstart.asm", and click the [Open] button.

<3> Click the [Next] button.

(5) The [New WorkSpace - Step 7/8 [Select Debugger]] dialog box will be displayed. Click the [Next] button.



New WorkSpace - Step 7/8 [Select Debugger]

Please select the Debugger.

* Debugger : SM+ for 78K0S/Kx1+ System Simulator

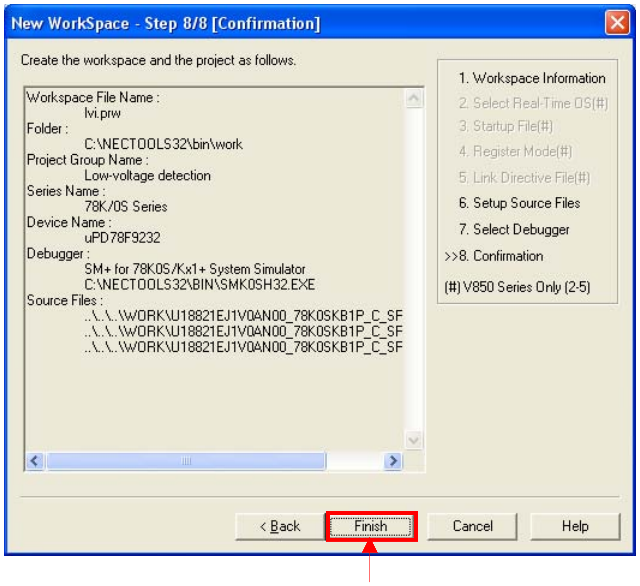
File name : C:\NECTOOLS32\BIN\SMK0SH32.EXE

You can change the selected Debugger using [Tool] -> [Select Debugger].

* Required field

Click

(6) The [New WorkSpace - Step 8/8 [Confirmation]] dialog box will be displayed. Confirm the settings and click the [Finish] button.



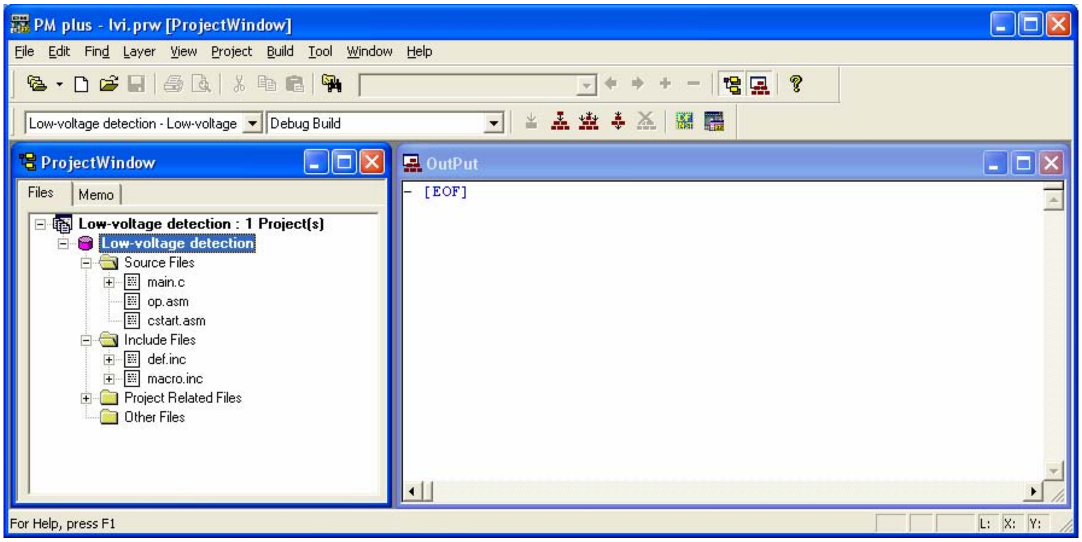
New WorkSpace - Step 8/8 [Confirmation]

Create the workspace and the project as follows.

```
Workspace File Name :
          lvi.prw
Folder :
          C:\NECTOOLS32\bin\work
Project Group Name :
          Low-voltage detection
Series Name :
          78K/0S Series
Device Name :
          uPD78F9232
Debugger :
          SM+ for 78K0S/Kx1+ System Simulator
          C:\NECTOOLS32\BIN\SMK0SH32.EXE
Source Files :
          ..\..\..\WORK\U18821EJ1V0AN00_78K0SKB1P_C_SF
          ..\..\..\WORK\U18821EJ1V0AN00_78K0SKB1P_C_SF
          ..\..\..\WORK\U18821EJ1V0AN00_78K0SKB1P_C_SF
```

Click

(7) A workspace will be created and the project will be registered.

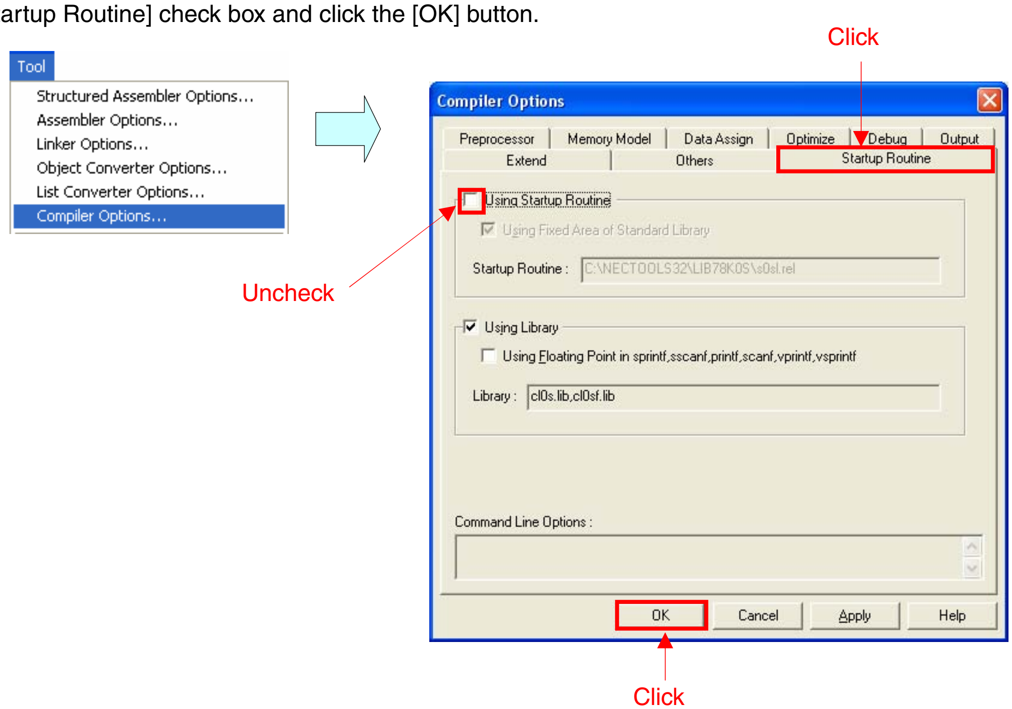


```
- [EOF]
```

- Compiler option settings

(8) Select [Compiler Options] from the [Tool] menu.

(9) The [Compiler Options] dialog box will be displayed. Click the [Startup Routine] tab, uncheck the [Using Startup Routine] check box and click the [OK] button.



☐ Using Startup Routine

☑ Using Fixed Area of Standard Library

Startup Routine : C:\NECTOOLS32\LIB78K0S\s0sl.rel

☑ Using Library

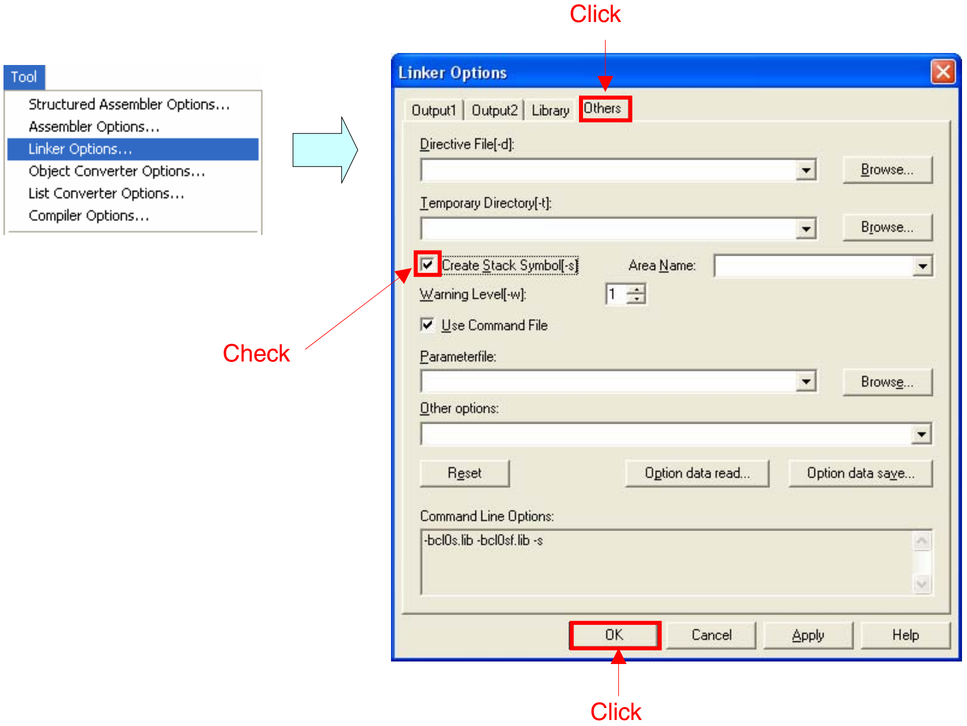
☐ Using Floating Point in sprintf,sscanf,printf,scanf,vprintf,vsprintf

Library : cl0s.lib,cl0sf.lib

- Linker option settings

(10) Select [Linker Options] from the [Tool] menu.

(11) The [Linker Options] dialog box will be displayed. Click the [Others] tab, check the [Create Stack Symbol[-s]] check box, and click the [OK] button.



☑ Create Stack Symbol[-s]

Warning Level[-w]: 1

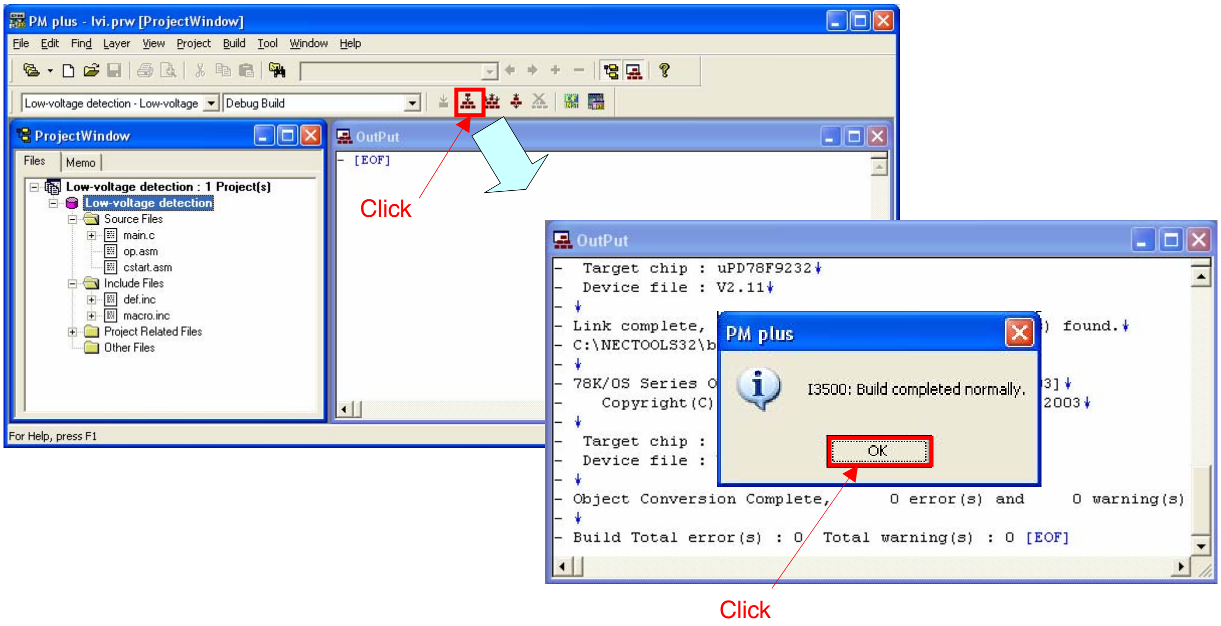
☑ Use Command File

Command Line Options:
-bcl0s.lib -bcl0sf.lib -s

- Build execution

(12) Click ([Build] button). When the source files are built normally, the message “I3500: Build completed normally.” will be displayed.

(13) Click the [OK] button in the message window. A HEX file for flash memory writing will be created.



```
- Target chip : uPD78F9232
- Device file : V2.11
-
- Link complete,                              ) found.
- C:\NECTOOLS32\b
-
- 78K/OS Series O                             3]
-    Copyright(C)                             2003
-
- Target chip :
- Device file :
-
- Object Conversion Complete,      0 error(s) and     0 warning(s)
-
- Build Total error(s) : 0  Total warning(s) : 0 [EOF]
```

I3500: Build completed normally.

A HEX file for flash memory writing will be generated. → Go to 5.2.

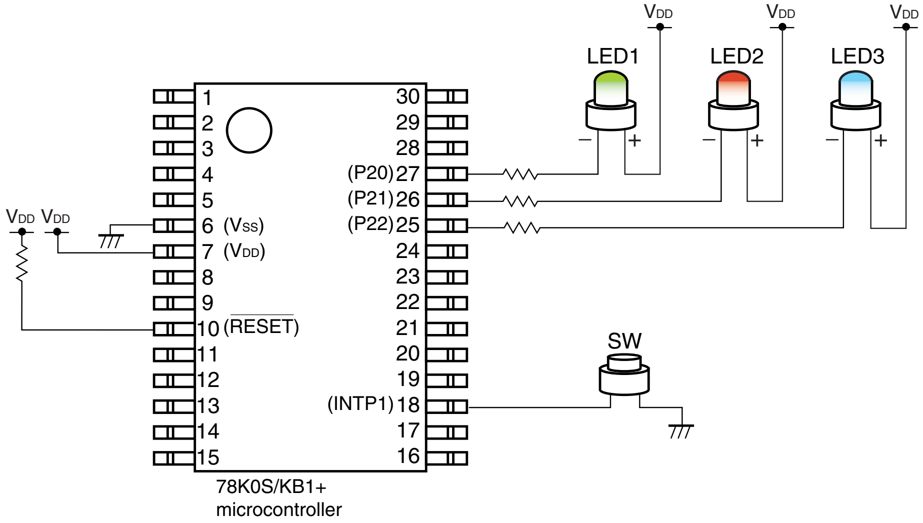
5.2 Operation with the Device

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

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

For how to write to the flash memory of the device, refer to the **78K0S/Kx1+ Simplified Flash Writing Manual Information** (being prepared).

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



An operation example of the device to which this sample program has been written is described below.

(1) When $V_{DD} \geq 3.0$ V (in normal operation)[Note]

The LED lighting pattern changes depending on the number of switch inputs.

Note Perform switch input interrupt servicing operation at $V_{DD} \geq 3.0$ V, because the low-voltage detection voltage (V_{LVI}) is set to 2.85 V ±0.15 V.

Number of Switch Inputs[Note]	LED Output		
	LED3	LED2	LED1
0	OFF	OFF	OFF
1	OFF	OFF	ON
2	OFF	ON	OFF
3	OFF	ON	ON
4	ON	OFF	OFF
5	ON	OFF	ON
6	ON	ON	OFF
7	ON	ON	ON

Note The lighting patterns from the zeroth switch input are repeated after the eighth switch input.

If the switch is pressed for less than 10 ms, the switch input is identified as chattering and the LED display pattern will not be changed (remains the same as before pressing the switch).

(2) $V_{DD} \geq 3.0$ V → $V_{DD} = 2.5$ V (low-voltage detection) → $V_{DD} \geq 3.0$ V

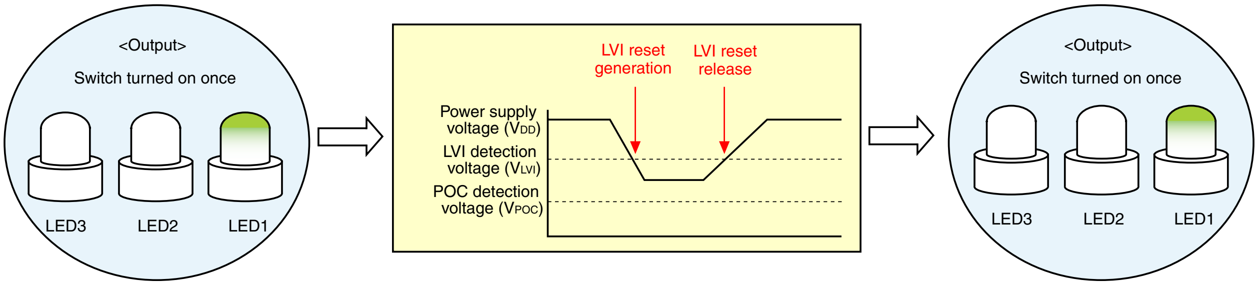
The lighting of the LEDs changes as follows in accordance with the change in power supply voltage.

<1> $V_{DD} \geq 3.0$ V → $V_{DD} = 2.5$ V

An LVI reset will be generated, because the low-voltage detection level (V_{LVI}) is set to 2.85 V ±0.15 V and the low-voltage detection function is set to be used for reset. At this time, all LEDs will be turned off, but RAM retains the LED display data immediately before the reset.

<2> $V_{DD} = 2.5$ V → $V_{DD} \geq 3.0$ V

Operation is returned to normal mode. At this time, the lighting pattern immediately before the reset will be restored, because the reset source is confirmed to be an LVI reset and the LED display data retained in RAM is read.



(3) $V_{DD} \geq 3.0$ V → 2.0 V > V_{DD} (less than data retention power supply voltage) → $V_{DD} \geq 3.0$ V

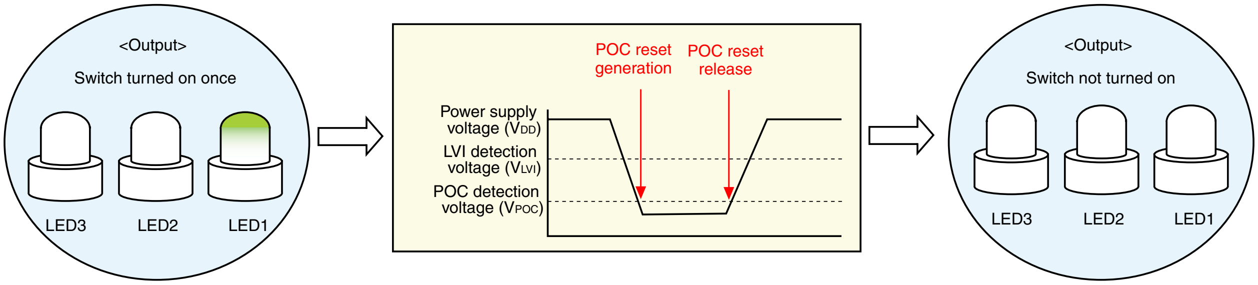
The lighting of the LEDs changes as follows in accordance with the change in power supply voltage.

<1> $V_{DD} \geq 3.0$ V → 2.0 V > V_{DD}

A reset caused by power-on-clear (POC) will be generated. At this time, all LEDs will be turned off and RAM data becomes undefined.

<2> 2.0 V > V_{DD} → $V_{DD} \geq 3.0$ V

Operation is returned to normal mode. At this time, all LEDs will be turned off (number of switch inputs = 0), because the reset source is confirmed to be other than an LVI reset and the LED lighting pattern of RAM is initialized.



CHAPTER 6 RELATED DOCUMENTS

Document Name		Japanese/English
78K0S/KU1+ User’s Manual		PDF
78K0S/KY1+ User’s Manual		PDF
78K0S/KA1+ User’s Manual		PDF
78K0S/KB1+ User’s Manual		PDF
78K/0S Series Instructions User’s Manual		PDF
RA78K0S Assembler Package User’s Manual	Language	PDF
	Operation	PDF
CC78K0S C Compiler User’s Manual	Language	PDF
	Operation	PDF
PM+ Project Manager User’s Manual		PDF
SM+ System Simulator Operation User’s Manual		PDF
78K0S/KA1+ Simplified Flash Writing Manual MINICUBE2 Information		PDF
78K0S/Kx1+ Sample Program (Initial Settings) LED Lighting Switch Control Application Note		PDF
78K0S/Kx1+ Sample Program Startup Guide Application Note		PDF
78K0S/Kx1+ Sample Program (Interrupt) External Interrupt Generated by Switch Input Application Note		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
;*****************************************************************************
;     Low-voltage detection
;*****************************************************************************
;<<History>>
;     2007.6.--   Release
;*****************************************************************************
;
;<<Overview>>
;
;This sample program presents an example of using the low-voltage detection
;(LVI) function.
;A low-voltage detector (LVI) is used to set so that an internal reset signal
;is generated when VDD is less than VLVI (2.85 V +-0.15 V).
;After completion of the initial settings, the LED lighting pattern changes,
;depending on the number of switch inputs.
;(This is the same processing described in Sample Program Interrupt.)
;Here, the number of switch inputs is initialized when a reset is generated
;by other than LVI, but when a reset is generated by LVI, the number of switch
;inputs before the reset is restored and an LED lighting pattern is displayed
;accordingly, because RAM data is retained.
;
;
;  <Principal setting contents>
;
;  - Stop the watchdog timer operation
;  - Set the low-voltage detection voltage (VLVI) to 2.85 V +-0.15 V
;  - Generate an internal reset signal (low-voltage detector) when VDD < VLVI
after VDD >= VLVI
;  - Set the CPU clock frequency to 4 MHz
;  - Set the valid edge of external interrupt INTP1 to falling edge
;  - Set the chattering removal time during switch input to 10 ms
;
;
;  <LED lighting pattern after low-voltage detection and reset release>
;
;  - Reset generated by other than the low-voltage detector ... Turn off all
LEDs
;  - Reset generated by the low-voltage detector ... Retain the LED lighting
pattern before the reset
;
;
;  <Number of switch inputs and LED lighting patterns>
;
;    +-------------------------------------+
;    |  SW Inputs |  LED3 |  LED2 |  LED1 |
```

```
;    |   (P43)     | (P22) | (P21) | (P20) |
;    |------------|-----------------------|
;    |   0 times  |  OFF  |  OFF  |  OFF  |
;    |   1 time   |  OFF  |  OFF  |  ON   |
;    |   2 times  |  OFF  |  ON   |  OFF  |
;    |   3 times  |  OFF  |  ON   |  ON   |
;    |   4 times  |  ON   |  OFF  |  OFF  |
;    |   5 times  |  ON   |  OFF  |  ON   |
;    |   6 times  |  ON   |  ON   |  OFF  |
;    |   7 times  |  ON   |  ON   |  ON   |
;    +----------------------------------+
;    # The lighting patterns from the zeroth switch input are repeated after
the eighth switch input.
;
;
;<<I/O port settings>>
;
;  Input: P43
;  Output: P00-P03, P20-P23, P30-P33, P40-P42, P44-P47, P120-P123, P130
;  # All unused ports are set as the output mode.
;
;*****************************************************************************


;=============================================================================
;
;     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    RESET_START       ;(1A) INTTM80
      DW    RESET_START       ;(1C) INTSRE6
      DW    RESET_START       ;(1E) INTSR6
      DW    RESET_START       ;(20) INTST6

;=============================================================================
;
;     Define the RAM
;
;=============================================================================
XRAM  DSEG  SADDR
CNT_1:      DS    1           ; For major loop
CNT_2:      DS    1           ; For minor loop
LEDDATA:    DS    1           ; LED display pattern variable

;=============================================================================
```

```
;
;     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

;-----------------------------------------------------------------------------
;     Detect low-voltage + initialize the watchdog timer + set the clock
;-----------------------------------------------------------------------------
;-----  Initialize the watchdog timer  -----
      MOV   WDTM, #01110111B  ; Stop the watchdog timer operation

;-----  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, #00000111B  ; Set the low-voltage detection level (VLVI) to
2.85 V +-0.15 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

      MOV   LEDDATA, #00000111B     ; Initialize the LED display data
```

```
;-----  Set the clock <2>  -----
SET_CLOCK:
      MOV   PPCC, #00000001B  ; The clock supplied to the peripheral hardware
(fxp) = fx/2 (= 4 MHz)
                              ; -> The clock supplied to the CPU (fcpu) = fxp
= 4 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   A,    LEDDATA     ; Read the LED display data
      MOV   P2,   A           ; Set LED output (P20-P22) and output latch of
P23 as low
      MOV   PM2,  #11110000B  ; Set P20-P23 as output mode

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

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

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

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

;-------------------------------------------------------------------------------
;     Set the interrupt
;-------------------------------------------------------------------------------
      MOV   INTM0,      #00000000B  ; Set the valid edge of INTP1 to falling
edge
      CLR1  PIF1              ; Clear invalid interrupt requests in advance
      CLR1  PMK1              ; Release the INTP1 interrupt mask

      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   CNT_1,      #215  ; Assign the count value for the major loop
      NOP
      NOP
LOOP_1:
      MOV   CNT_2,      #17   ; Assign the count value for the minor loop
      NOP
LOOP_2:
      NOP
      DBNZ  CNT_2,      $LOOP_2           ; Minor loop
      DBNZ  CNT_1,      $LOOP_1           ; Major loop

      CLR1  PIF1              ; Clear the INTP1 interrupt request

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

;-----  LED lighting processing  -----
      DEC   LEDDATA           ; Decrement the current LED display data by 1
      AND   LEDDATA, #00000111B     ; Mask bits other than bits 0 to 2
      MOV   A,    LEDDATA     ; Read the LED display data
      MOV   P2,   A           ; Output the LED light

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

end
```

- main.c (C language version)

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

      NEC Electronics     78K0S/KB1+

******************************************************************************
      78K0S/KB1+  Sample program
******************************************************************************
      Low-voltage detection
******************************************************************************
<<History>>
      2007.6.--   Release
******************************************************************************

<<Overview>>

This sample program presents an example of using the low-voltage detection
(LVI) function.
A low-voltage detector (LVI) is used to set so that an internal reset signal
is generated when VDD is less than VLVI (2.85 V +-0.15 V).
After completion of the initial settings, the LED lighting pattern changes,
depending on the number of switch inputs.
(This is the same processing described in Sample Program Interrupt.)
Here, the number of switch inputs is initialized when a reset is generated
by other than LVI, but when a reset is generated by LVI, the number of switch
inputs before the reset is restored and an LED lighting pattern is displayed
accordingly, because RAM data is retained.


  <Principal setting contents>

  - Declare a function run by an interrupt: INTP1 -> fn_intp1()
  - Stop the watchdog timer operation
  - Set the low-voltage detection voltage (VLVI) to 2.85 V +-0.15 V
  - Generate an internal reset signal (low-voltage detector) when VDD < VLVI
after VDD >= VLVI
  - Set the CPU clock frequency to 4 MHz
  - Set the valid edge of external interrupt INTP1 to falling edge
  - Set the chattering detection time during switch input to 10 ms


  <LED lighting pattern after low-voltage detection and reset release>

  - Reset generated by other than the low-voltage detector ... Turn off all
LEDs
  - Reset generated by the low-voltage detector ... Retain the LED lighting
pattern before the reset


  <Number of switch inputs and LED lighting patterns>

    +------------------------------------+
    |  SW Inputs |  LED3 |  LED2 |  LED1 |
    |   (P43)    | (P22) | (P21) | (P20) |
    |------------|-----------------------|
    |   0 times  |  OFF  |  OFF  |  OFF  |
    |   1 time   |  OFF  |  OFF  |  ON   |
    |   2 times  |  OFF  |  ON   |  OFF  |
    |   3 times  |  OFF  |  ON   |  ON   |
```

```
    |   4 times  |  ON    |  OFF  |  OFF  |
    |   5 times  |  ON    |  OFF  |  ON   |
    |   6 times  |  ON    |  ON   |  OFF  |
    |   7 times  |  ON    |  ON   |  ON   |
    +-------------------------------------+
    # The lighting patterns from the zeroth switch input are repeated after
the eighth switch input.


<<I/O port settings>>

  Input: P43
  Output: P00-P03, P20-P23, P30-P33, P40-P42, P44-P47, P120-P123, P130
  # All unused ports are set as the output mode.

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


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

      Preprocessing directive (#pragma)

==========================================================================*/
#pragma     SFR                     /* SFR names can be described at the C
source level */
#pragma     EI                      /* EI instructions can be described at the
C source level */
#pragma     NOP                     /* NOP instructions can be described at
the C source level */
#pragma interrupt INTP1 fn_intp1    /* Interrupt function declaration:INTP1 */

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

      Initialization after RESET

****************************************************************************/
      sreg unsigned char g_ucLED;   /* 8-bit variable for LED display data
(high-speed internal RAM area) */

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  = 0b00000111;      /* Set the low-voltage detection level
(VLVI) to 2.85 V +-0.15 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 */

            g_ucLED = 0b00000111;    /* Initialize the LED display data */
      }

      /* Set the clock <2> */
      PPCC  = 0b00000001;            /* The clock supplied to the peripheral
hardware (fxp) = fx/2 (= 4 MHz)
                                     -> The clock supplied to the CPU (fcpu) =
fxp = 4 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    = g_ucLED;               /* Output the LED display data */
      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 P43 as input mode, P40-P42 and P44-
P47 as output mode */

/*------------------------------------------------------------------------------
      Initialize the port 12
------------------------------------------------------------------------------*/
      P12   = 0b00000000;            /* Set output latches of P120-P123 as low
*/
      PM12  = 0b11110000;            /* Set P120-P123 as output mode */
```

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

/*------------------------------------------------------------------------------
      Set the interrupt
------------------------------------------------------------------------------*/
      INTM0 = 0b00000000;           /* Set the valid edge of INTP1 to falling
edge */
      PIF1  = 0;                    /* Clear invalid interrupt requests in
advance */
      PMK1  = 0;                    /* Release the INTP1 interrupt mask */

      return;
}

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

      Main loop

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

      EI();                         /* Enable vector interrupt */

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

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

      External interrupt INTP1

*******************************************************************************/
__interrupt void fn_intp1(){
      unsigned int unChat;    /* 16-bit variable for the chattering removal
timer */

      for (unChat = 0; unChat < 555; unChat++){ /* Wait of about 10 ms (for
chattering removal) */
            NOP();
      }

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

      if (!P4.3){       /* Processing performed if SW is on for 10 ms or more
*/
            g_ucLED -= 1;           /* Decrement the current LED display data
by 1 */
            g_ucLED &= 0b00000111;  /* Mask bits other than bits 0 to 2 */
            P2 = g_ucLED;           /* Output the LED light */
      }

      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	October 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 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/