To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

Renesas Electronics website: http://www.renesas.com

April 1st, 2010
Renesas Electronics Corporation

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

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

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: "Standard", "High Quality", and "Specific". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as "Specific" without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as "Specific" or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is "Standard" unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

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

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

   "Specific": Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

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

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

RENESAS

Application Note

78K0S/Kx1+

Sample Program (Initial Settings)

LED Lighting Switch Control

This document summarizes the initial settings for the sample program and describes basic initial settings for the microcontroller. In the sample program, the two switch inputs and lighting of the three LEDs are controlled, after the basic initial settings for the microcontroller, such as selecting the clock frequency or I/O port, have been performed.

Target devices

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

CONTENTS

Document No. U18752EJ2V0AN00 (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

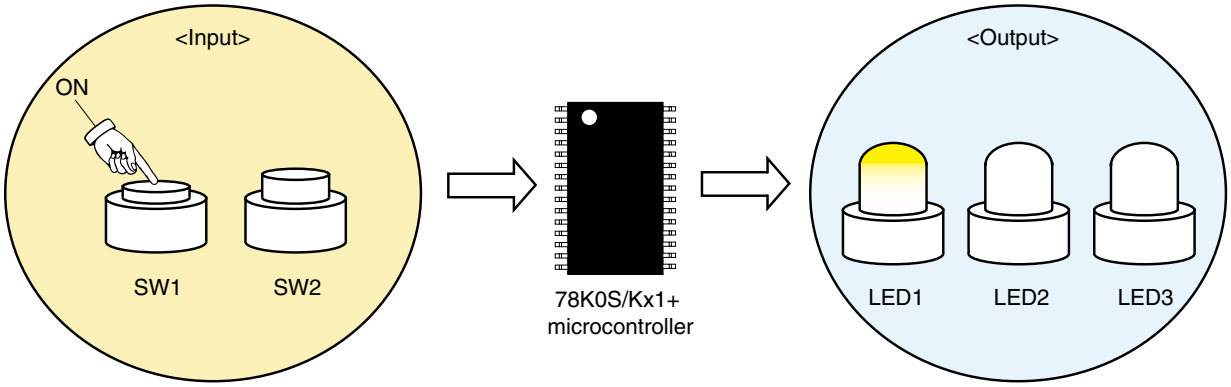
In this sample program, the basic initial settings for the 78K0S/Kx1+ microcontroller, such as setting the option byte, selecting the clock frequency, and setting I/O ports are performed. In the main processing operation after completion of the initial settings, the lighting of the three LEDs is controlled by using the two switch inputs.

(1) Main contents of initial settings

- Selecting the high-speed internal oscillator as the system clock source
- Stopping watchdog timer operation
- Setting the CPU clock frequency and peripheral hardware clock frequency to 2 MHz
- Setting I/O ports

(2) Contents of main processing operation

Lighting of the LEDs (LED1, LED2, LED3) is controlled by detecting switch inputs (SW1, SW2) with the 78K0S/Kx1+ microcontroller.



Switch Input		LED Output		
SW1	SW2	LED1	LED2	LED3
OFF	OFF	OFF	OFF	OFF
ON	OFF	ON	OFF	OFF
OFF	ON	OFF	ON	OFF
ON	ON	OFF	OFF	ON

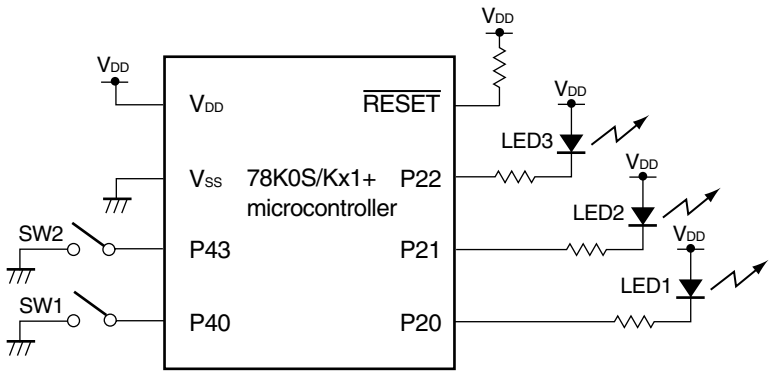
Caution For cautions when using the device, refer to the user's manual of each product (78K0S/KU1+, 78K0S/KY1+, 78K0S/KA1+, 78K0S/KB1+).

CHAPTER 2 CIRCUIT DIAGRAM

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

2.1 Circuit Diagram

A circuit diagram is shown below.



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) Switches (SW1, SW2)

These switches are used as inputs to control the lighting of the LEDs.

(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.

File Name	Description	Compressed (*.zip) File Included		
				
main.asm (Assembly language version) main.c (C language version)	Source file for hardware initialization processing and main processing of microcontroller	●Note 1	●Note 1	
op.asm	Assembler source file for setting the option byte (sets the system clock source)	●	●	
initial.prw	Work space file for integrated development environment PM+		●	
initial.prj	Project file for integrated development environment PM+		●	
initial.pri initial.prs initial.prm	Project files for system simulator SM+ for 78K0S/Kx1+		●Note 2	
initial0.pnl	I/O panel file for system simulator SM+ for 78K0S/Kx1+ (used for checking peripheral hardware operations)		●Note 2	●
initial0.wvo	Timing chart file for system simulator SM+ for 78K0S/Kx1+ (used for checking waveforms)			●

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

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

Remark  : Only the source file is included.

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

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

3.2 Internal Peripheral Functions to Be Used

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

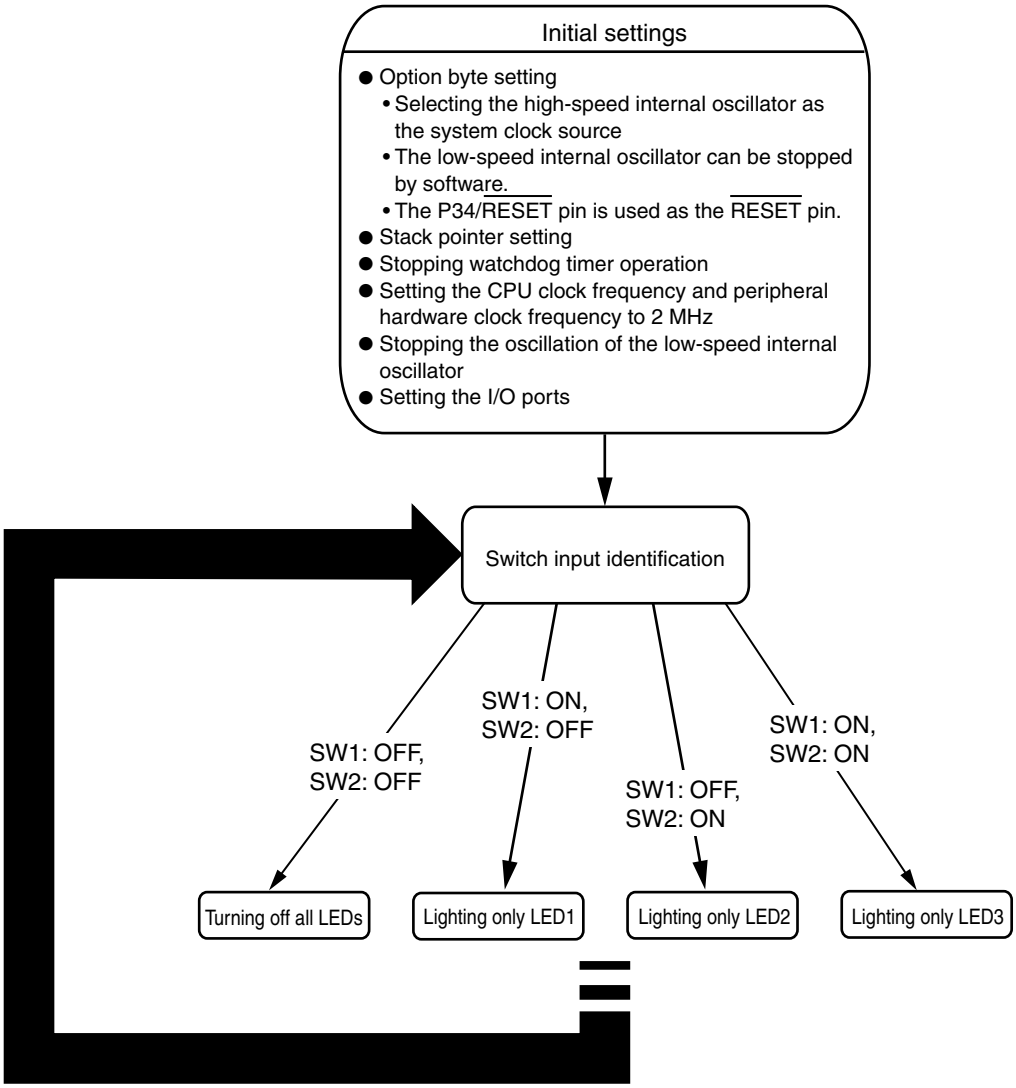
- Input ports (for switch inputs): P40, P43
- Output ports (for lighting LEDs): P20, P21, P22

3.3 Initial Settings and Operation Overview

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

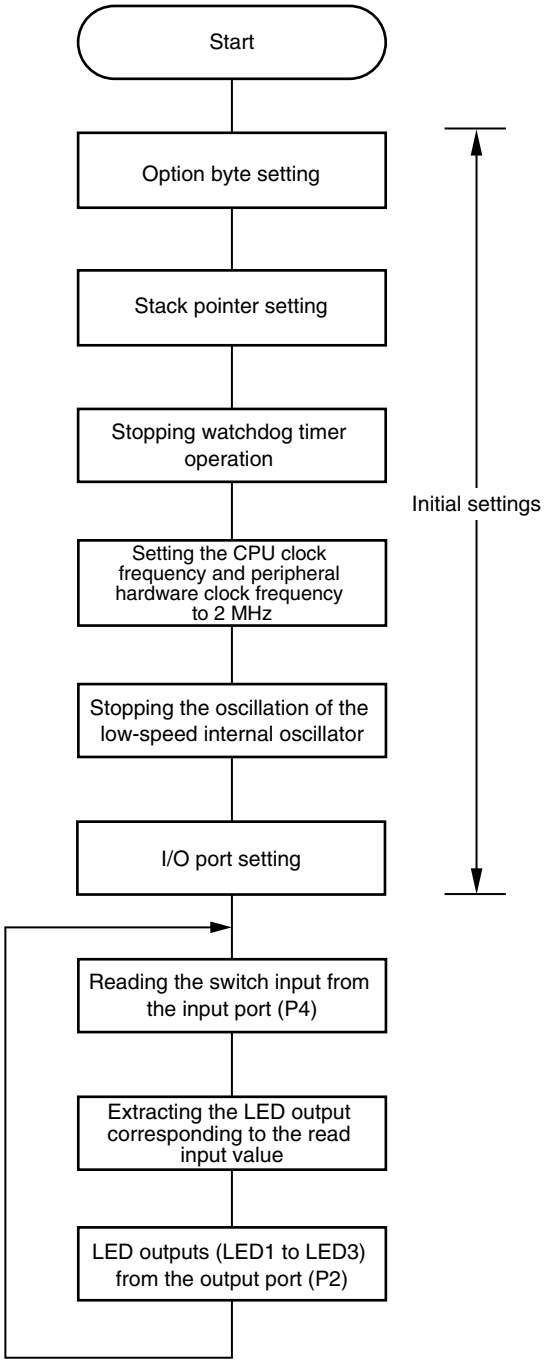
After completion of the initial settings, the lighting of the three LEDs (LED1, LED2, LED3) is controlled in accordance with the combination of the two switch inputs (SW1, SW2).

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



3.4 Flow Chart

A flow chart for the sample program is shown below.



CHAPTER 4 SETTING METHODS

This chapter describes how to set the option byte, vector table, stack pointer, watchdog timer, clock frequency, and ports, as well as the main processing.

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 Option Byte Setting

The option byte must be set. The following items are set with the option byte.

(1) Selection of system clock source
(2) Control of low-speed internal oscillator oscillation
(3) Selection of P34/$\overline{\text{RESET}}$ pin function
(4) Specification of oscillation stabilization time on power application or after reset release (only when using crystal or ceramic oscillation)

In this sample program, the option byte is set as described in **[Example 1]**, mentioned later.

Figure 4-1. Option Byte Format

Address: 0080H

1	DEFOS TS1	DEFOS TS0	1	RMCE	OSCSE L1	OSCSE L0	LIOCP

Control of low-speed internal oscillator oscillation[Note 1]

0	Oscillation is stopped by software setting (LSRSTOP bit is set to 1).
1	Cannot be stopped.

Selection of system clock source[Note 1]

0	0	Crystal or ceramic oscillation clock (10 MHz (MAX.))
0	1	External clock input (10 MHz (MAX.))
1	x	High-speed internal oscillation clock (8 MHz (TYP.))

Selection of P34/$\overline{\text{RESET}}$ pin function

0	Used as P34 (dedicated input port).
1	Used as a $\overline{\text{RESET}}$ pin.

Specification of oscillation stabilization time on power application or after reset release[Note 2]

0	0	2^{10}/fx
0	1	2^{12}/fx
1	0	2^{15}/fx
1	1	2^{17}/fx

Note 1. Stopping the low-speed internal oscillator oscillation, stopping watchdog timer operation, and setting the clock frequency are performed after reset release. For details, refer to **4.4 Watchdog Timer Setting** and **4.5 Clock Setting**.

(Note 2, Caution, and Remark are on the next page.)

Note 2. If the high-speed internal oscillation clock or an external clock input is selected as the system clock, no oscillation stabilization time is required, so the setting becomes invalid (don't care).

Caution Bits 4 and 7 must be set to 1.

Remark x: don't care

[Example 1] The high-speed internal oscillation clock is used, used as the $\overline{\text{RESET}}$ pin, and the low-speed internal oscillator can be stopped (same setting as the sample program).

Address: 0080H

1	x	x	1	1	1	x	0

Control of low-speed internal oscillator oscillation[Note]

0	Oscillation is stopped by software setting (LSRSTOP bit is set to 1).

* In this sample program, the watchdog timer is not used. Oscillation is therefore set so that it can be stopped by software.

Selection of system clock source[Note]

1	x	High-speed internal oscillation clock (8 MHz (TYP.))

Selection of P34/$\overline{\text{RESET}}$ pin function

1	Used as a $\overline{\text{RESET}}$ pin.

Specification of oscillation stabilization time on power application or after reset release

x	x	(Crystal or ceramic oscillation is not used, so setting is invalid.)

Note Stopping the low-speed internal oscillator oscillation, stopping watchdog timer operation, and setting the clock frequency are performed after reset release. For details, refer to **4.4 Watchdog Timer Setting** and **4.5 Clock Setting**.

The option byte setting value is "1xx111x0 (x: don't care, bits 4 and 7 must be set to 1)". When the software is described together with the protect byte setting (hereinafter, writing to all blocks is enabled), the following results. ("x" is set to 0 in the example below.)

```
OPBT   CSEG   AT     0080H
       DB     10011100B
       DB     11111111B
```

To use C language, prepare an assembly language source file (file name: "*.asm (*: arbitrary)") such as the one shown below, specify it as the project source file, and build it together with other source files (main.c).

```
OPBT   CSEG   AT     0080H
       DB     10011100B
       DB     11111111B
       end
```

Remark The protect byte (address: 0081H) is used to set the prohibited areas for writing and block erasure whose set contents are valid only during self programming. For details, refer to the user's manual of each product (**78K0S/KU1+**, **78K0S/KY1+**, **78K0S/KA1+**, **78K0S/KB1+**).

[Example 2] A crystal or ceramic oscillation clock is used, used as the $\overline{\text{RESET}}$ pin, the low-speed internal oscillator can be stopped, and the oscillation stabilization time is minimized (2^{10}/fx).
(This is the same as in Example 1, except for the underlined parts.)

Address: 0080H

1	0	0	1	1	0	0	0

Control of low-speed internal oscillator oscillation

0	Oscillation is stopped by software setting (LSRSTOP bit is set to 1).

Selection of system clock source

0	0	Crystal or ceramic oscillation clock (10 MHz (MAX.))

Selection of P34/$\overline{\text{RESET}}$ pin function

1	Used as a $\overline{\text{RESET}}$ pin.

Specification of oscillation stabilization time on power application or after reset release

0	0	2^{10}/fx

The option byte setting value is “10011000 (bits 4 and 7 must be set to 1)”. When the software is described together with the protect byte setting, the following results.

```
OPBT    CSEG    AT      0080H
        DB      10011000B
        DB      11111111B
```

[Column] What are CSEG (Code Segment), DSEG (Data Segment), and BSEG (Bit Segment)?
CSEG, DSEG, and BSEG are pseudo instructions which indicate where generated codes of instructions, data, or the like are to be allocated. Instructions and data which are described after such pseudo instructions have been issued are allocated in the ROM area with a CSEG pseudo instruction, in the RAM area with a DSEG pseudo instruction, and in the saddr area in RAM with a BSEG pseudo instruction.
For example, to allocate the option byte setting content to addresses starting from 0080H in the internal ROM (flash memory), first, the CSEG pseudo instruction and AT attribute are used to specify 0080H as the address. Next, the DB pseudo instruction is used to define values that are to be set to addresses following 0080H, which are then described in the program coded in assembly language.
The DB and DW pseudo instructions can be used only in a ROM area specified with the CSEG pseudo instruction. Descriptions of the DB or DW pseudo instructions in a RAM area specified with the DSEG or BSEG pseudo instruction will not cause errors, but must not be used. In this case, an object is generated and debug operation can be performed, since with MINICUBE2 (on-chip debug emulator) or SM+ (system simulator), coded instructions and data are expanded to the RAM area. With an actual device, however, operation is disabled since these cannot be expanded to the RAM area.
For details of the CSEG, DSEG, and BSEG pseudo instructions, refer to the **RA78K0S Language User’s Manual**.

[Example 3] A crystal or ceramic oscillation clock is used, used as the P34 pin, the low-speed internal oscillator can be stopped, and the oscillation stabilization time is minimized ($2^{10}/f_X$).
(This is the same as in Example 2, except for the underlined part.)

Address: 0080H

1	0	0	1	0	0	0	0

Control of low-speed internal oscillator oscillation

0	Oscillation is stopped by software setting (LSRSTOP bit is set to 1).

Selection of system clock source

0	0	Crystal or ceramic oscillation clock (10 MHz (MAX.))

Selection of P34/$\overline{\text{RESET}}$ pin function

0	Used as a P34 pin.

Specification of oscillation stabilization time on power application or after reset release

0	0	$2^{10}/fx$

The option byte setting value is “10010000 (bits 4 and 7 must be set to 1)”. When the software is described together with the protect byte setting, the following results.

```
OPBT   CSEG   AT     0080H
       DB     10010000B
       DB     11111111B
```

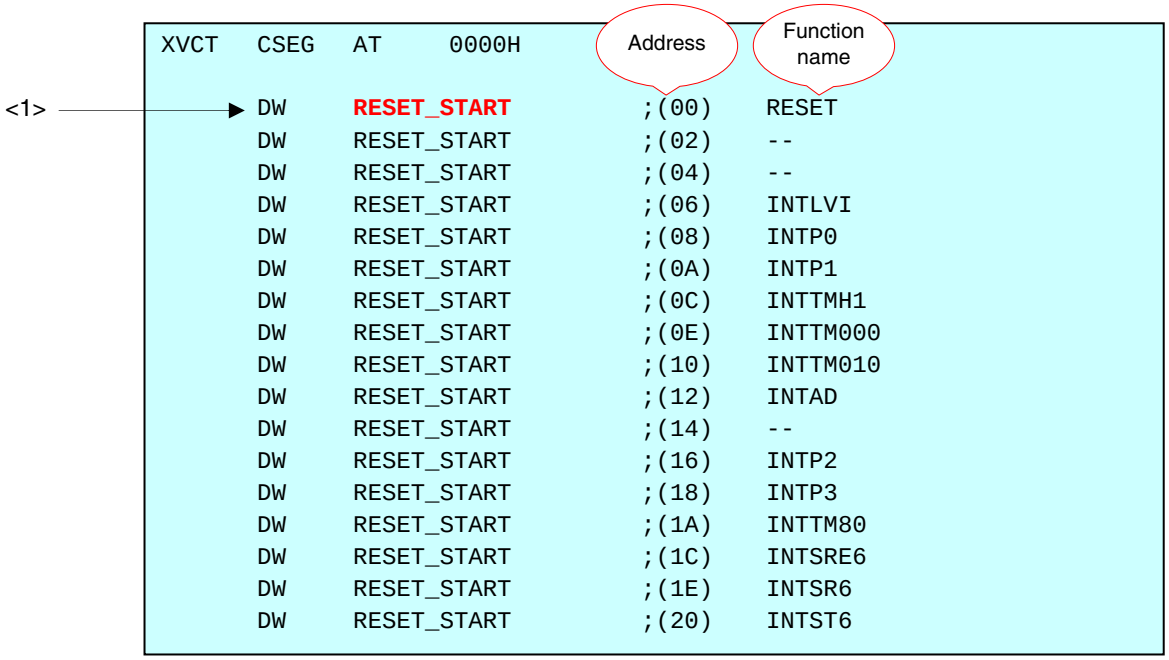
4.2 Vector Table Setting

In the vector table area, the program start address, which is used when branching occurs due to the generation of resets and various interrupt requests, is stored.

In this sample program, interrupt servicing is not performed, so only the reset vector which is used during reset start is set. This setting is required when coding in assembly language. When coding in C language, this setting is not required, since the reset vector is automatically set in the startup routine.

<R> For how to set the vector table and interrupts as well as setting examples, refer to "**Sample Program (Interrupt) External Interrupt Generated by Switch Input**".

[Example 1] Setting only the reset vector to be used during reset start (same as in the sample program setting)




```
XVCT    CSEG    AT      0000H
<1>     DW      RESET_START         ;(00)    RESET
        DW      RESET_START         ;(02)    --
        DW      RESET_START         ;(04)    --
        DW      RESET_START         ;(06)    INTLVI
        DW      RESET_START         ;(08)    INTP0
        DW      RESET_START         ;(0A)    INTP1
        DW      RESET_START         ;(0C)    INTTMH1
        DW      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
```

After reset release, the program starts from the address (RESET_START at <1>, above) specified with the reset vector.

In this sample program, vector table addresses except 0000H are not used. RESET_START is set to all remaining vector table addresses, as with 0000H. By setting in this way, branching to RESET_START occurs, even if an interrupt occurs, and the same processing as after reset release is performed.

[Column] What are #pragma directives?

#pragma directives are preprocessing instructions which are used in the C language and are coded at the beginning of source programs.

The following are major #pragma directives.

- #pragma sfr: Operations related to the SFR area can be described at the C source level.
- #pragma ei: The EI instruction can be described at the C source level.
- #pragma di: The DI instruction can be described at the C source level.
- #pragma nop: The NOP instruction can be described at the C source level. (The clock can be advanced without operating the CPU.)
- #pragma interrupt: Interrupt functions can be described at the C source level.

For details of the #pragma directives, refer to the chapter regarding expansion functions, in the **CC78K0S Language User's Manual**.

4.3 Stack Pointer Setting

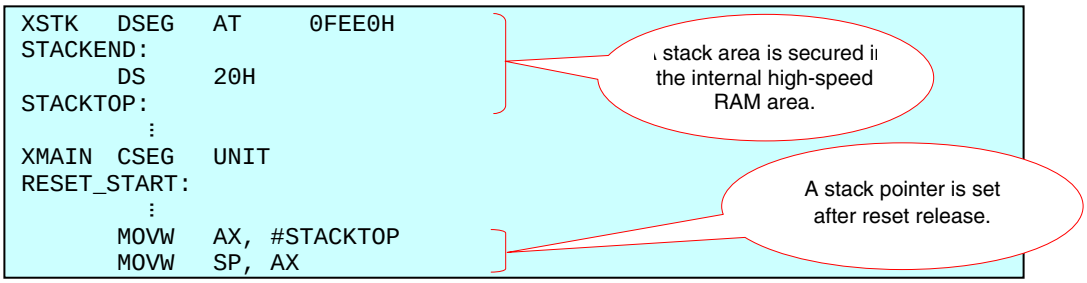
A stack area is a memory area in which data, such as of program counters, register values, and PSW (program status word) is temporarily stored. A stack area can be specified only to the internal high-speed RAM area. The start address of this stack area is set with a stack pointer to secure the stack area.

A stack area is used when the following instructions are executed or interrupts occur.

- PUSH, CALL, CALLT, interrupt: Saving data to a stack area
- POP, RET, RETI: Restoring data from a stack area

The securing of a stack area is a required setting when coding in assembly language. When coding in C language, this setting is not required, since a stack area is automatically secured in the startup routine.

[Example 1] Using FEE0H to FEFFH (32 bytes) in the internal high-speed RAM area as the stack area (Same as in the sample program setting)

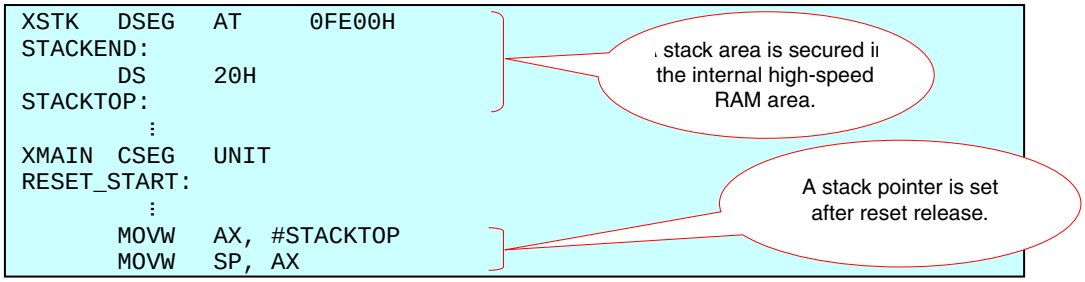


In this example, address FF00H (= FEE0H + 20H) is specified for the stack pointer. FF00H is not only a high-speed RAM area, but also an SFR area, so it is converted into a high-speed RAM area, and the stack pointer becomes FB00H.

When actually storing data to the stack, the stack pointer becomes FAFFH, as a result of decrementing (–1) FB00H, into which FF00H was converted. This, however, is not a high-speed RAM area, so it is converted into a high-speed RAM area, and becomes FEFFH. This is the same value as when address FF00H is specified to the stack pointer, and data is stored starting from address FEFFH.

Thanks to the description above, the last 32 bytes (FEE0H to FEFFH) of the internal high-speed RAM area can be secured as the stack area.

[Example 2] Using FE00H to FE1FH (32 bytes) in the internal high-speed RAM area as the stack area



In this example, address FE20H (= FE00H + 20H) is specified for the stack pointer. This setting avoids the saddr area and secures the stack area.

Caution The setting of example 2, above, can only be set to products with a 256-byte internal high-speed RAM.

4.4 Watchdog Timer Setting

The WDTM register is used to select the operation clock for the watchdog timer and the overflow time.

In this sample program, the watchdog timer is not used for program loop detection, so the WDTM register is set as described in **[Example 1]**, mentioned later.

Caution The operation clock and overflow time for the watchdog timer must be set during the initial settings.

Figure 4-2. Format of Watchdog Timer Mode Register (WDTM)

WDTM

0	1	1	WDCS4	WDCS3	WDCS2	WDCS1	WDCS0

Overflow time setting[Note 1]

			During low-speed internal oscillation clock operation	During system clock operation
0	0	0	$2^{11}/f_{RL}$	$2^{13}/f_X$
0	0	1	$2^{12}/f_{RL}$	$2^{14}/f_X$
0	1	0	$2^{13}/f_{RL}$	$2^{15}/f_X$
0	1	1	$2^{14}/f_{RL}$	$2^{16}/f_X$
1	0	0	$2^{15}/f_{RL}$	$2^{17}/f_X$
1	0	1	$2^{16}/f_{RL}$	$2^{18}/f_X$
1	1	0	$2^{17}/f_{RL}$	$2^{19}/f_X$
1	1	1	$2^{18}/f_{RL}$	$2^{20}/f_X$

Operation clock selection[Note 2]

0	0	Low-speed internal oscillation clock (f_{RL})
0	1	System clock (f_X)
1	x	Watchdog timer operation stop

Notes 1. When “Watchdog timer operation stop” is selected, the overflow time setting is invalid (don’t care).

2. To stop watchdog timer operation or to use the system clock as the watchdog timer operation clock, the option byte must be used to set to “Oscillation is stopped by software setting (LSRSTOP bit is set to 1)”. For details, refer to **4.1 Option Byte Setting**.

Caution Bits 7, 6, and 5 must be set to 0, 1, and 1, respectively.

Remark x: don’t care

[Example 1] Watchdog timer stop (same as in the sample program setting)

WDTM

0	1	1	1	x	x	x	x

Overflow time setting

x	x	x	(Setting is invalid, because watchdog timer operation is stopped.)

Operation clock selection

1	x	Watchdog timer operation stop[Note]

* In this sample program, the watchdog timer is not used for program loop detection, so it is set to operation stop.

Note To stop watchdog timer operation, the option byte must be used to set to “Oscillation is stopped by software setting (LSRSTOP bit is set to 1)”. For details, refer to **4.1 Option Byte Setting**.

The WDTM register setting value is “0111xxxx (x: don’t care, bits 7, 6, and 5 must be set to 0, 1, and 1, respectively)”. (In the example below, “x” of bit 3 is set to 0, and “x” of bits 2 to 0 is set to 1.)

• Assembly language

```
MOV     WDTM,   #01110111B
```

• C language

```
WDTM = 0b01110111;
```

[Example 2] The low-speed internal oscillation clock (f_{RL}) is used as the watchdog timer operation clock and the overflow time is set to the maximum period ($2^{18}/f_{RL}$) (same as the WDTM value after reset release).

WDTM

0	1	1	0	0	1	1	1

Overflow time setting

1	1	1	$2^{18}/f_{RL}$

Operation clock selection

0	0	Low-speed internal oscillation clock (f_{RL})

To use the WDTM register as described above, its setting value is the same as its value after reset release, so setting by a program is not required.

[Column] Binary-value description

To describe a binary value, append “B” or “Y” after the binary value in assembly language, or append “0b” or “0B” before the binary value in C language.

[Example 3] The system clock (fx) is used as the watchdog timer operation clock and the overflow time is set to the minimum period (2^{13}/fx).

WDTM

0	1	1	0	1	0	0	0

Overflow time setting

0	0	0	2^{13}/fx

Operation clock selection

0	1	System clock (fx)Note

Note To use the system clock as the watchdog timer operation clock, the option byte must be used to set to “Oscillation is stopped by software setting (LSRSTOP bit is set to 1)”. For details, refer to **4.1 Option Byte Setting**.

The WDTM register setting value is “01101000 (bits 7, 6, and 5 must be set to 0, 1, and 1, respectively)”.

• Assembly language

```
MOV     WDTM,   #01101000B
```

• C language

```
WDTM = 0b01101000;
```

[Column] hdwinit function and main function

To create a program in C language, the hdwinit function is called to initialize peripheral devices (SFR) immediately after CPU reset. Initial settings, such as the watchdog timer setting and clock frequency selection are therefore basically described in the hdwinit function.

The main function is called after calling the hdwinit function, so main processing is described in the main function.

Do not call the hdwinit function from the main function. In this case, the hdwinit function is executed twice and the watchdog timer setting, which is only allowed to be set once is executed twice. As a result, an internal reset signal is generated during the second execution disabling the program to advance from the initial setting.

For details, refer to the **CC78K0S Language User’s Manual** and **Processing to be executed first** under Programming on the NEC Electronics FAQ Web page.

4.5 Clock Setting

(1) Clock frequency setting

The CPU clock frequency (f_{CPU}) and the frequency of the clock supplied to peripheral hardware (f_{XP}) are generated by dividing the frequency of the system clock set with the option byte.

The PPCC and PCC registers are used to select the CPU clock frequency (f_{CPU}) and the frequency of the clock supplied to peripheral hardware (f_{XP}).

In this sample program, the PPCC and PCC registers are set as described in **[Example 1]**, mentioned later, so that $f_{CPU} = f_{XP} = 2$ MHz.

Figure 4-3. Formats of Processor Clock Control Register (PCC) and Preprocessor Clock Control Register (PPCC)

PPCC

0	0	0	0	0	0	PPCC1	PPCC0

PCC

0	0	0	0	0	0	PCC1	0

PPCC1	PPCC0	PCC1	CPU Clock Frequency (f_{CPU})	Peripheral Hardware Clock Frequency (f_{XP})
0	0	0	f_X	f_X
0	0	1	$f_X/4$	f_X
0	1	0	$f_X/2$	$f_X/2$
0	1	1	$f_X/8$	$f_X/2$
1	0	0	$f_X/4$	$f_X/4$
1	0	1	$f_X/16$	$f_X/4$
Other than the above			Setting prohibited	

Cautions 1. Bits 2 to 7 of PPCC and bits 0 and 2 to 7 of PCC must be set to 0.

2. The clock frequency range that can be used varies, depending on the power supply voltage.

Resonator	Conditions	CPU Clock Frequency (f_{CPU})	Peripheral Hardware Clock Frequency (f_{XP})
Ceramic resonator, crystal resonator, external clock	4.0 to 5.5 V	125 kHz ≤ f_{CPU} ≤ 10 MHz	500 kHz ≤ f_{XP} ≤ 10 MHz
	3.0 to 4.0 V	125 kHz ≤ f_{CPU} ≤ 6 MHz	
	2.7 to 3.0 V	125 kHz ≤ f_{CPU} ≤ 5 MHz	
	2.0 to 2.7 V[Note]	125 kHz ≤ f_{CPU} ≤ 2 MHz	500 kHz ≤ f_{XP} ≤ 5 MHz
High-speed internal oscillator	4.0 to 5.5 V	500 kHz (TYP.) ≤ f_{CPU} ≤ 8 MHz (TYP.)	2 MHz (TYP.) ≤ f_{XP} ≤ 8 MHz (TYP.)
	2.7 to 4.0 V	500 kHz (TYP.) ≤ f_{CPU} ≤ 4 MHz (TYP.)	
	2.0 to 2.7 V[Note]	500 kHz (TYP.) ≤ f_{CPU} ≤ 2 MHz (TYP.)	2 MHz (TYP.) ≤ f_{XP} ≤ 4 MHz (TYP.)

Note Use standard products and (A)-grade products within a voltage range of 2.2 to 5.5 V, since the detection voltage (V_{POC}) of the power-on-clear (POC) circuit is 2.1 V ±0.1 V.

Use (A2)-grade products within a voltage range of 2.26 to 5.5 V, since the detection voltage (V_{POC}) of the power-on-clear (POC) circuit is 2.26 V (MAX.).

Remark f_X: System clock frequency

The system clock to be used is set using the option byte. For details, refer to **4.1 Option Byte Setting**.

[Example 1] $f_{CPU} = f_{XP} = f_X/4$

(This is the same as in the sample program setting. In this sample program, “high-speed internal oscillation clock (8 MHz (TYP.))” is selected as the system clock (f_X) by the option byte setting, so the CPU clock frequency (f_{CPU}) and peripheral hardware clock frequency (f_{XP}) are 2 MHz (= 8 MHz/4).)

PPCC

0	0	0	0	0	0	1	0

PCC

0	0	0	0	0	0	0	0

PPCC1	PPCC0	PCC1	CPU Clock Frequency (f_{CPU})	Peripheral Hardware Clock Frequency (f_{XP})
1	0	0	$f_X/4$	$f_X/4$

Remark f_X: System clock frequency

The system clock to be used is set using the option byte. For details, refer to **4.1 Option Byte Setting**.

The PPCC register setting value is “00000010 (bits 2 to 7 must be set to 0)” and the PCC register setting value is “00000000 (bits 0 and 2 to 7 must be set to 0)”.

For standard products and (A)-grade products, the detection voltage (V_{POC}) of the power-on-clear (POC) circuit is 2.1 V ±0.1 V. To satisfy the condition of “power supply voltage ≥ 2.0 V” (setting condition of the CPU clock frequency in this sample program) at the point POC is released, the CPU operates normally when this sample program operates. This is same with (A2)-grade products.

• Assembly language

```
MOV     PPCC,   #00000010B
MOV     PCC,    #00000000B
```

• C language

```
PPCC = 0b00000010;
PCC = 0b00000000;
```

[Example 2] f_{CPU} = f_X/8, f_{XP} = f_X/2

(When “high-speed internal oscillation clock (8 MHz (TYP.))” is selected as the system clock (f_X) by the option byte setting, the CPU clock frequency (f_{CPU}) is 1 MHz (= 8 MHz/8) and the peripheral hardware clock frequency (f_{XP}) is 4 MHz (= 8 MHz/2).)

PPCC

0	0	0	0	0	0	**0**	**1**

PCC

0	0	0	0	0	0	**1**	0

PPCC1	PPCC0	PCC1	CPU Clock Frequency (f_{CPU})	Peripheral Hardware Clock Frequency (f_{XP})
0	1	1	f_X/8	f_X/2

Remark f_X: System clock frequency

The system clock to be used is set using the option byte. For details, refer to **4.1 Option Byte Setting**.

The PPCC register setting value is “0000001 (bits 2 to 7 must be set to 0)” and the PCC register setting value is “0000010 (bits 0 and 2 to 7 must be set to 0)”.

The clock frequency range that can be used varies, depending on the power supply voltage, so caution is required when setting the PPCC and PCC. (For details, refer to **Figure 4-3. Formats of Processor Clock Control Register (PCC) and Preprocessor Clock Control Register (PPCC)**.)

• Assembly language

```
MOV     PPCC,   #00000001B
MOV     PCC,    #00000010B
```

• C language

```
PPCC = 0b00000001;
PCC = 0b00000010;
```

(2) Low-speed internal oscillator setting

The low-speed internal oscillation clock is used as the clock source for the watchdog timer and 8-bit timer H1.

The LSRCM register is used to oscillate or stop the low-speed internal oscillator.

In this sample program, the low-speed internal oscillation clock is not used, so the LSRCM register is set as described in **[Example 1]**, mentioned below.

Figure 4-4. Format of Low-Speed Internal Oscillation Mode Register (LSRCM)

LSRCM

0	0	0	0	0	0	0	LSR STOP

Oscillation/stop of low-speed internal oscillator[Note]

0	Oscillation
1	Stop

Note To stop the low-speed internal oscillator, the option byte must be used to set to “Oscillation is stopped by software setting (LSRSTOP bit is set to 1)”. For details, refer to **4.1 Option Byte Setting**.

Caution Bits 1 to 7 of LSRCM must be set to 0.

[Example 1] Oscillation stop of the low-speed internal oscillator (same as in the sample program setting)

LSRCM

0	0	0	0	0	0	0	1

Oscillation/stop of low-speed internal oscillator[Note]

1	Stop

Note To stop the low-speed internal oscillator, the option byte must be used to set to “Oscillation is stopped by software setting (LSRSTOP bit is set to 1)”. For details, refer to **4.1 Option Byte Setting**.

The LSRCM register setting value is “0000000**1** (bits 7 to 1 must be set to 0)”.

• Assembly language

```
MOV     LSRCM,   #00000001B
```

• C language

```
LSRCM = 0b00000001;
```

[Example 2] Oscillation of the low-speed internal oscillator

LSRCM

0	0	0	0	0	0	0	**0**

Oscillation/stop of low-speed internal oscillator[Note]

0	Oscillation

Note When oscillation of the low-speed internal oscillator is set to “Cannot be stopped” by using the option byte, the LSRCM setting is invalid (don’t care). For details, refer to **4.1 Option Byte Setting**.

The LSRCM register setting value is “0000000**0** (bits 7 to 1 must be set to 0)”.

• Assembly language

```
MOV     LSRCM,   #00000000B
```

• C language

```
LSRCM = 0b00000000;
```

(3) Setting of the oscillation stabilization time after releasing the STOP mode (when crystal or ceramic oscillation is used for the system clock source)

The OSTS register is used to set the oscillation stabilization time after releasing the STOP mode. The OSTS setting is valid only when crystal or ceramic oscillation is used for the system clock source.

In this sample program, the high-speed internal oscillation clock is used, so the OSTS register is not set.

Figure 4-5. Format of Oscillation Stabilization Time Selection Register (OSTS)

OSTS

0	0	0	0	0	0	OSTS1	OSTS0

Specification of the oscillation stabilization time after releasing the STOP mode[Note]

0	0	2^{10}/fx
0	1	2^{12}/fx
1	0	2^{15}/fx
1	1	2^{17}/fx

Note When the high-speed internal oscillation clock or an external clock input is selected as the system clock, the OSTS setting is not required.

Caution Bits 7 to 2 must be set to 0.

[Example 1] Setting the oscillation stabilization time to 409.6 μs when a crystal or ceramic oscillation clock is used (fx = 10 MHz)

OSTS

0	0	0	0	0	0	**0**	**1**

0	1	2^{12}/fx (409.6 μs)

The OSTS register setting value is “00000001 (bits 7 to 2 must be set to 0)”.

• Assembly language

```
MOV     OSTS,   #00000001B
```

• C language

```
OSTS = 0b00000001;
```

4.6 Port Setting

Caution The on-chip ports vary, depending on each product, so the ports to be set also vary.

	78K0S/KA1+	78K0S/KB1+	78K0S/KU1+	78K0S/KY1+
Port 0	–	P00 to P03	–	–
Port 2	P20 to P23	P20 to P23	P20 to P23	P20 to P23
Port 3	P30, P31, P34Note	P30 to P33, P34Note	P32, P34Note	P32, P34Note
Port 4	P40 to P45	P40 to P47	P40, P43	P40 to P47
Port 12	P121 to P123	P120 to P123	–	–
Port 13	P130	P130	–	–

Note P34 is shared with $\overline{\text{RESET}}$. Selecting the function to be used is set using the option byte. For details, refer to **4.1 Option Byte Setting**.

(1) Port I/O setting

The PMxx registers are used to set whether ports are to be used as input ports or output ports. Ports are set as input ports after reset release.

The PMxx format is described, taking the PM2 register as an example.

In this sample program, port 2 is set as described in **[Example 1]**, mentioned later, and port 4 as in **[Example 2]**.

Figure 4-6. Format of Port Mode Register 2 (PM2)

PM2

1	1	1	1	PM23	PM22	PM21	PM20

Selection of the P2n (n = 0 to 3) pin I/O mode

0	Output mode
1	Input mode

Caution Bits 7 to 4 must be set to 1.

(2) Setting the output port output latch

The Pxx registers are used to set output port output latches to high level or low level. Output port output latches are set to low-level output after reset release.

The Pxx format is described, taking the P2 register as an example.

In this sample program, port 2 is set as described in **[Example 1]**, mentioned later.

Figure 4-7. Format of Port Register 2 (P2)

P2

0	0	0	0	P23	P22	P21	P20

Selection of the P2n (n = 0 to 3) pin output latch level

0	Low-level output
1	High-level output

Caution Bits 7 to 4 must be set to 0.

(3) Setting the connection of internal pull-up resistors to input ports

The PUxx registers are used to set whether internal pull-up resistors are to be connected to the input ports. Internal pull-up resistors are not connected after reset release.

The PUxx format is described, taking the PU2 register as an example.

In this sample program, port 4 is set as described in **[Example 2]**, mentioned later.

Figure 4-8. Format of Pull-up Resistor Option Register 2 (PU2)

PU2

0	0	0	0	PU23	PU22	PU21	PU20

Selection of the P2n (n = 0 to 3) pin internal pull-up resistor

0	Internal pull-up resistor is not connected.
1	Internal pull-up resistor is connected.

Caution Bits 7 to 4 must be set to 0.

[Example 1]
- Setting P20 to P23 as output ports
- Setting the P20 to P22 output latches to high-level output, setting the P23 output latch to low-level output (Same as in the sample program setting)

PM2

1	1	1	1	0	0	0	0

Selection of the P2n (n = 0 to 3) pin I/O mode

0	Output mode

* In this sample program, P20 to P22 are used as output ports for lighting LEDs, so port 2 is set as an output port.

P2

0	0	0	0	0	1	1	1

Selection of the P20 to P22 pin output latch levels

1	High-level output

Selection of the P23 pin output latch level

0	Low-level output

* In this sample program, P20 to P22 are used as output ports for lighting LEDs, so the P20 to P22 output latch levels are set to high-level output in the initial settings.
(LEDs are lit when the outputs from P20 to P22 are at low level (refer to **2.1 Circuit Diagram**).)

The PM2 register setting value is “11110000 (bits 7 to 4 must be set to 1)”, and the P2 register setting value is “00000111 (bits 7 to 4 must be set to 0)”.

- Assembly language

```
MOV   PM2, #11110000B
MOV   P2,  #00000111B
```

- C language

```
PM2 = 0b11110000;
P2 = 0b00000111;
```

[Example 2] • Setting P40 and P43 as input ports
• Setting internal pull-up resistors to be connected to P40 and P43
(Same as in the sample program setting)

PM4

• 78K0S/KB1+, 78K0S/KY1+

x	x	x	x	1	x	x	1

Selection of the P40 and P43 pin I/O modes

1	Input mode

• 78K0S/KA1+

0Note	0Note	x	x	1	x	x	1

Selection of the P40 and P43 pin I/O modes

1	Input mode

Note Bits 7 and 6 are set to 1 after reset release. They must be set to 0 in the initial settings.

• 78K0S/KU1+

0Note	0Note	0Note	0Note	1	0Note	0Note	1

Selection of the P40 and P43 pin I/O modes

1	Input mode

* In this sample program, P40 and P43 are used as switch input ports, so P40 and P43 are set as input ports.

Note Bits 7 to 4, 2, and 1 are set to 1 after reset release. They must be set to 0 in the initial settings.

PU4

• 78K0S/KB1+, 78K0S/KY1+

x	x	x	x	1	x	x	1

Selection of connection of internal pull-up resistors to P40 and P43 pins

1	Internal pull-up resistor is connected.

• 78K0S/KA1+

0	0	x	x	1	x	x	1

Selection of connection of internal pull-up resistors to P40 and P43 pins

1	Internal pull-up resistor is connected.

Caution Bits 7 and 6 must be set to 0.

• 78K0S/KU1+

0	0	0	0	1	0	0	1

Selection of connection of internal pull-up resistors to P40 and P43 pins

1	Internal pull-up resistor is connected.

Caution Bits 7 to 4, 2, and 1 must be set to 0.

* In this sample program, internal pull-up resistors are connected to P40 and P43 to control the P40 and P43 input levels. (High-level input is selected when the switch is off, and low-level input is selected when the switch is on (refer to **2.1 Circuit Diagram**).)

The PM4 register setting value is “xxxx1xx1 (x: don’t care)” with the 78K0S/KB1+ and 78K0S/KY1+, “00xx1xx1 (x: don’t care)” with the 78K0S/KA1+, and “00001001” with the 78K0S/KU1+. The PU4 register setting value is “xxxx1xx1 (x: don’t care)” with the 78K0S/KB1+ and 78K0S/KY1+, “00xx1xx1 (x: don’t care)” with the 78K0S/KA1+, and “00001001” with the 78K0S/KU1+. (In the example below, “x” in the PM4 and PU4 register values is set to 0.)

• Assembly language

```
MOV  PM4, #00001001B
MOV  PU4, #00001001B
```

• C language

```
PM4 = 0b00001001;
PU4 = 0b00001001;
```

4.7 Main Processing

The following operations are performed with the main processing in assembly language.

<1> The P4 register data is read.

<2> Among the read 8-bit data, bits except those of the input ports (P40, P43) are cleared to 0.

<3> Output data corresponding to the combination of the P40 and P43 input levels are extracted from addresses 0100H to 0109H ("LEDDATA" table).

<4> The extracted data is output to the P2 register.

Thanks to operations <1> and <2>, only the combination of the input levels of P40 and P43 connected to the switches (SW1 and SW2) can be identified. In operation <3>, only data of addresses 0100H, 0101H, 0108H, and 0109H, among addresses 0100H to 0109H are used (because there are only four patterns of input level combinations).

ROM area setting

```
XTBL1  CSEG   AT     0100H
LEDDATA:
       DB     00000011B          ;00: SW1=ON, SW2=ON   -> LED3=ON
       DB     00000101B          ;01: SW1=OFF,SW2=ON   -> LED2=ON
       DB     00000111B          ;02: Dummy
       DB     00000111B          ;03: Dummy
       DB     00000111B          ;04: Dummy
       DB     00000111B          ;05: Dummy
       DB     00000111B          ;06: Dummy
       DB     00000111B          ;07: Dummy
       DB     00000110B          ;08: SW1=ON, SW2=OFF  -> LED1=ON
       DB     00000111B          ;09: SW1=OFF,SW2=OFF  -> LED1,2,3=OFF
```

Address 0100H

Main processing

```
       MOVW   HL,    #LEDDATA     ; Set the table address to the
                                   ; HL register
;*************************************************************************
;
;      Main loop
;
;*************************************************************************
MAIN_LOOP:

<1>    MOV    A,     P4           ; Read switch-input status
<2>    AND    A,     #00001001B   ; Mask the data except the switch-input status
       MOV    L,     A            ; Set switch-input status to the lower
<3>                                ; 8 bits of the table address
       MOV    A,     [HL]         ; Read LED output data from the table
<4>    MOV    P2,    A            ; Output the LED light
       BR     $MAIN_LOOP          ; Go to the MAIN_LOOP
```

Correspondences of bits 0 and 3 between SW1 and SW2, and P4
(a) SW1 = SW2 = ON: P40 = P43 = 0
(b) SW1 = OFF, SW2 = ON: P40 = 1, P43 = 0
(c) SW1 = ON, SW2 = OFF: P40 = 0, P43 = 1
(d) SW1 = SW2 = OFF: P40 = P43 = 1

The data of address 0100H, 0101H, 0108H, or 0109H (data of the underlined part) is read.

The address of the A register after operation is 00H ((a) above), 01H ((b) above), 08H ((c) above), or 09H ((d) above).

Correspondences between the P2 output value and LEDs 1 to 3
(a) P2 = 00000011: Lights only LED3.
(b) P2 = 00000101: Lights only LED2.
(c) P2 = 00000110: Lights only LED1.
(d) P2 = 00000111: Lights off all LEDs.

The main processing in C language operates similarly to that in assembly language.
In C language, the correspondence between the input data and output data is set as an array.

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

      Main loop

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

      const unsigned char OUTDATA[10] =
       {0x03,0x05,0x07,0x07,0x07,0x07,0x07,0x07,0x06,0x07};
                                    /* Array of LED output pattern */
      unsigned char INDATA;         /* Switch-input variable */

      while(1){
            INDATA = P4 & 0b00001001;  /* Valid switch-input status */
            P2 = OUTDATA[INDATA];      /* Read LED output data from the table */
      }
```

Ten units of data are defined within the brackets wherein output data is set. In this main processing, however, only the underlined parts are used as output data. (Parts except the underlined parts are dummy data.)

The correspondences between the input data and output data are as follows.

Switch Input	P40, P43	INDATA	OUTDATA	LED Lighting
SW1 = ON, SW2 = ON	P40 = 0, P43 = 0	0b00000000	0x03	Lights only LED3.
SW1 = OFF, SW2 = ON	P40 = 1, P43 = 0	0b00000001	0x05	Lights only LED2.
SW1 = ON, SW2 = OFF	P40 = 0, P43 = 1	0b00001000	0x06	Lights only LED1.
SW1 = OFF, SW2 = OFF	P40 = 1, P43 = 1	0b00001001	0x07	Lights off all LEDs.

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 that has been downloaded by selecting the  icon.

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

<R> 5.1 Building the Sample Program

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

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

[Column] Build errors

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

<1> Select [Compiler Options] from the [Tool] menu.
<2> The [Compiler Options] dialog box will be displayed. Select the [Startup Routine] tab.
<3> Uncheck the [Using Fixed Area of Standard Library] check box. (Leave the other check boxes as they are.)

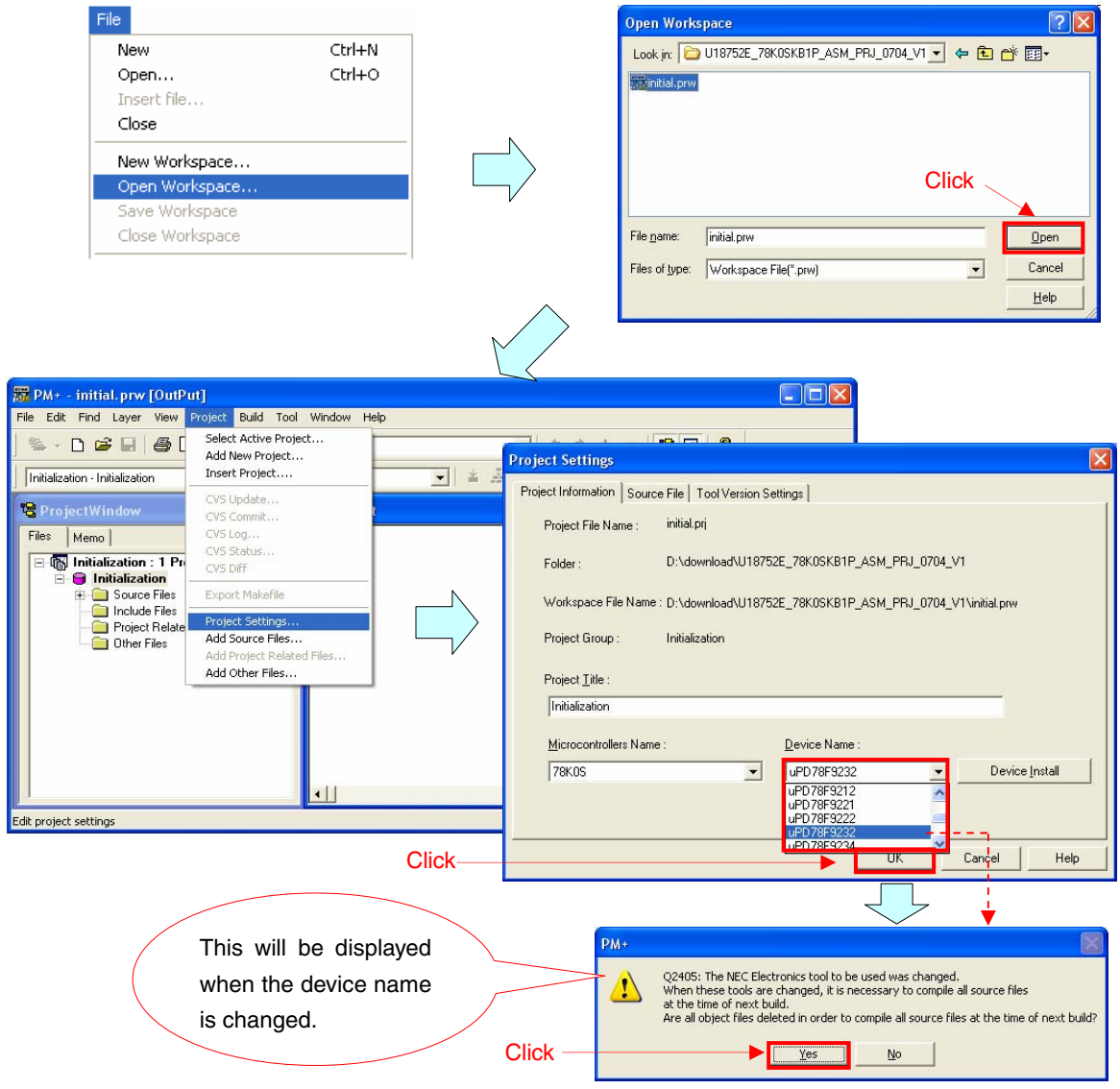
A RAM area of 118 bytes that has been secured as a fixed standard library area will be enabled for use when the [Using Fixed Area of Standard Library] check box is unchecked; however, the standard libraries (such as the getchar function and malloc function) will be disabled for use.

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

(1) Start PM+.

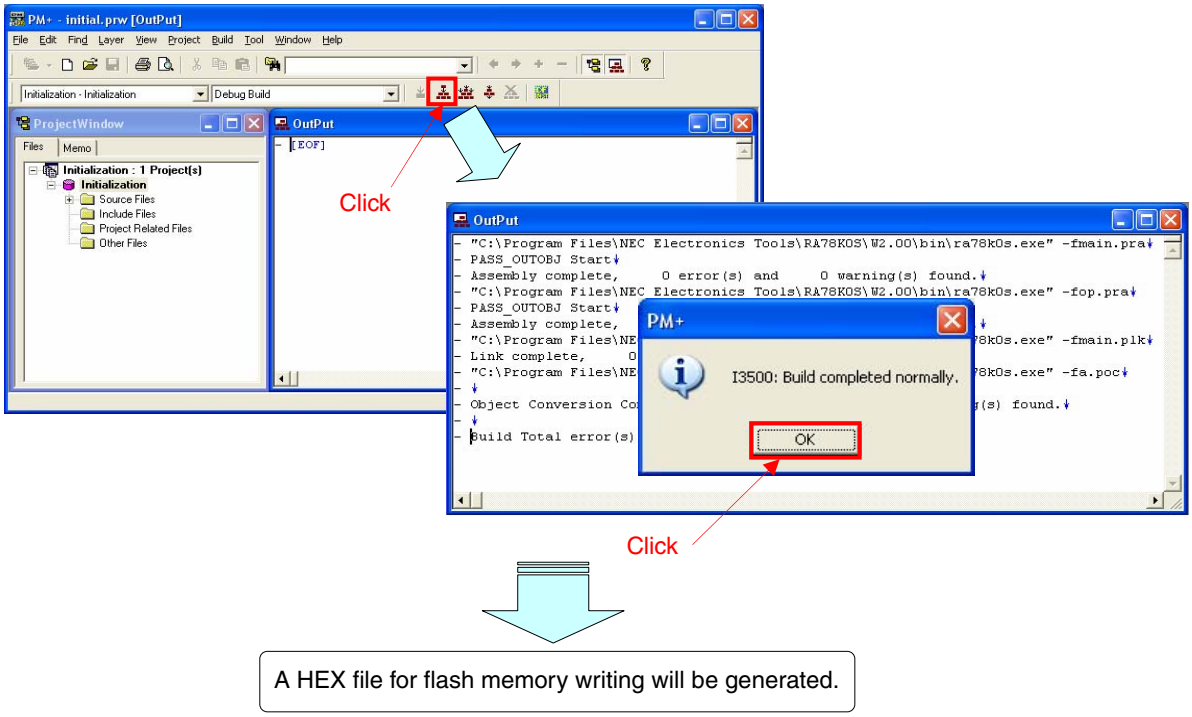
(2) Select “initial.prw” by clicking [Open Workspace] from the [File] menu and click [Open]. A workspace into which the source file will be automatically read will be created.

(3) Select [Project Settings] from the [Project] menu. When the [Project Settings] window opens, select the name of the device to be used (the device with the largest ROM or RAM size will be selected by default), and click [OK].



(4) Click  ([Build] 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.



5.2 Operation with SM+

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

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

(a) When starting SM+ in PM+

<1> Select [Register Ex-tool] from the [Tool] menu and register “SM+ for 78K0S/Kx1+”.

<2> Select [Ex-tool Bar] from the [View] menu and add the SM+ icon to the PM+ toolbar.

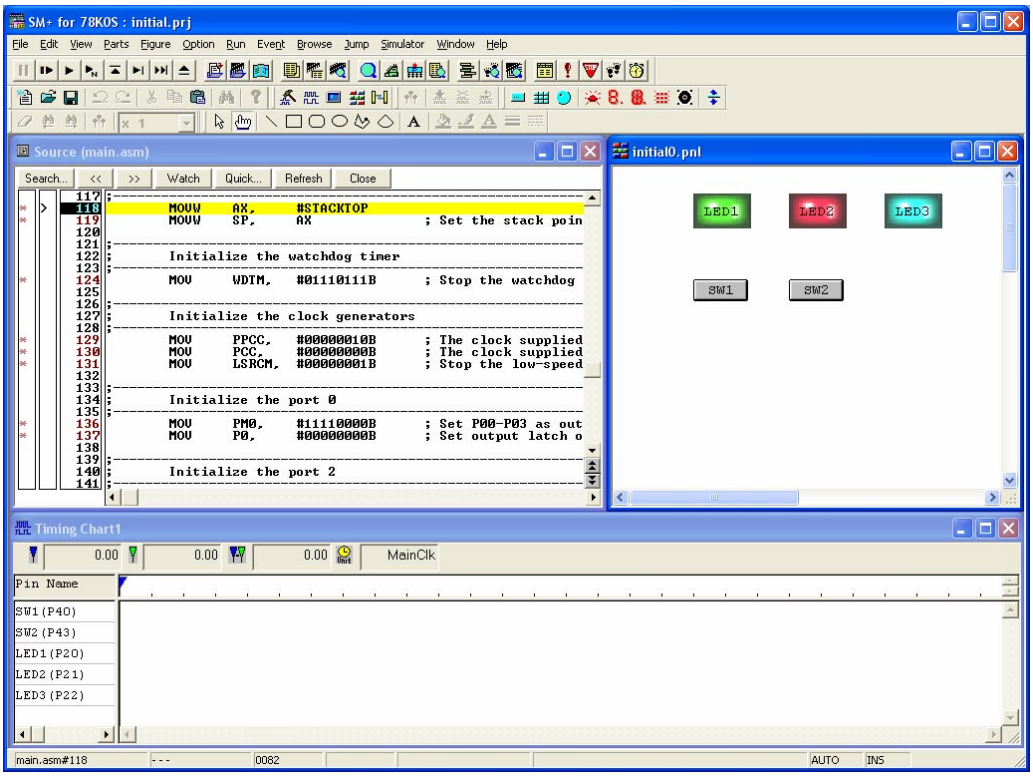
<3> Click the SM+ icon and start SM+.

(See the PM+ help for details on how to register external tools.)

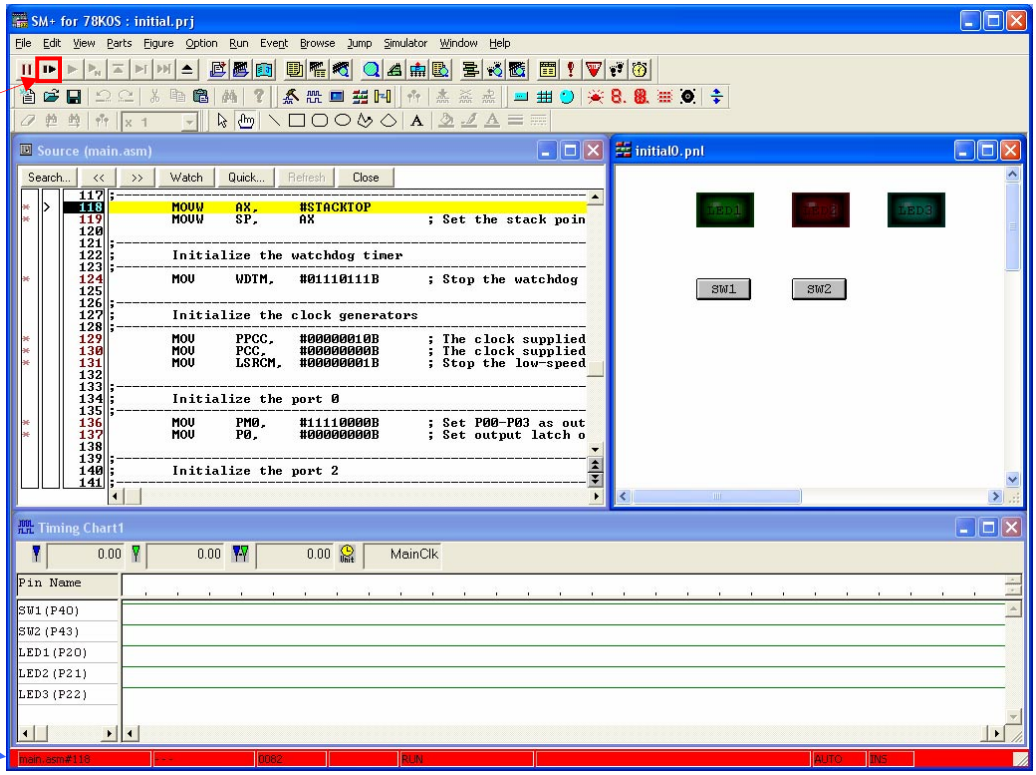
(b) When not starting SM+ in PM+

•Start SM+ from the Windows start menu.

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

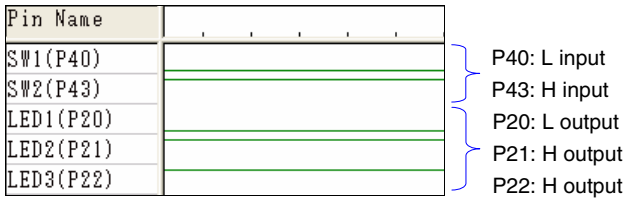


(3) Click  ([Restart] button). The program will be executed after the CPU is reset and the following screen will be displayed.

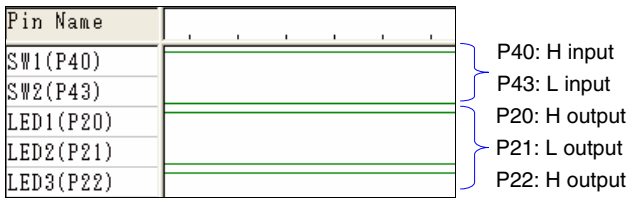


(4) Click the [SW1] and [SW2] buttons in the I/O panel window, during program execution. Check the lighting of [LED1] to [LED3] in the I/O panel window, as well as the waveforms in the timing chart window change, depending on the combination of the [SW1] and [SW2] buttons.

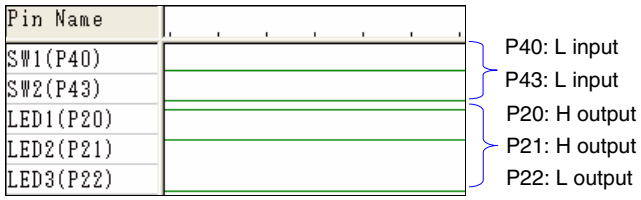
Example 1. SW1: ON, SW2: OFF

I/O panel window	Timing chart window
	

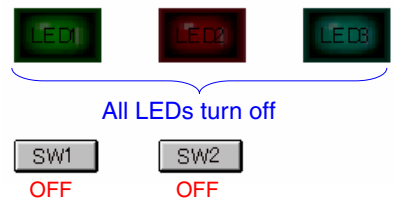
Example 2. SW1: OFF, SW2: ON

I/O panel window	Timing chart window
	

Example 3. SW1: ON, SW2: ON

I/O panel window	Timing chart window
	

Example 4. SW1: OFF, SW2: OFF

I/O panel window	Timing chart window
	

Remark H: High level, L: Low level

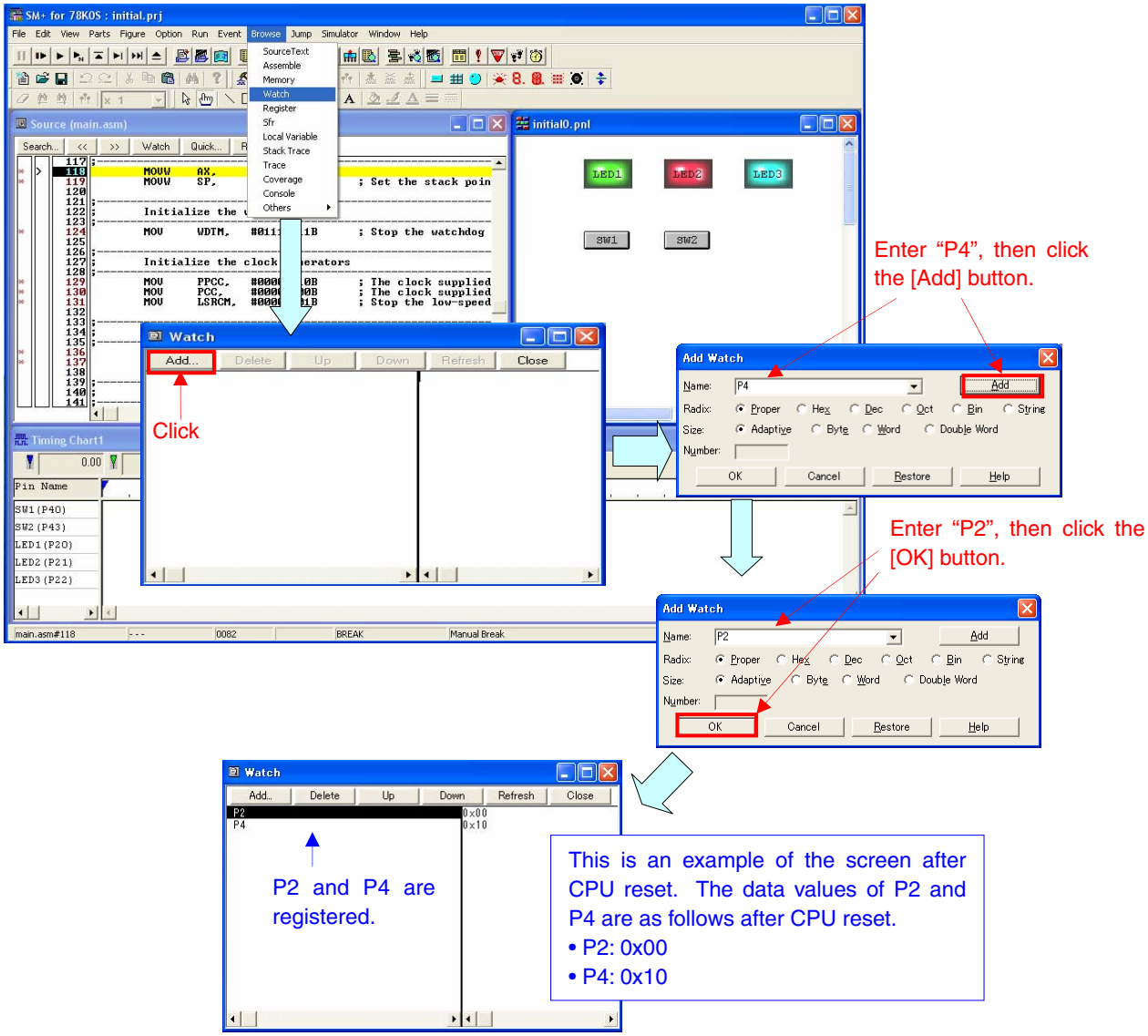
[Supplement] The changes in the data values of ports 2 and 4 can be checked by using the SM+ watch function.

<1> Select [Watch] from the [Browse] menu to open the [Watch] window.

<2> Click [Add] to open the [Add Watch] window. (At this time, the [Watch] window is kept opened.)

<3> Enter “P4” in the [Name] field and click the [Add] button to register “P4” in the [Watch] window. (At this time, the [Add Watch] window is kept opened.)

<4> Next, enter “P2” in the [Name] field and click the [OK] button to register “P2” in the [Watch] window and close the [Add Watch] window.



<5> Execute the program and click the [SW1] and [SW2] buttons in the I/O panel window. Check that the data values of P2 and P4 in the [Watch] window change, depending on the combination of the [SW1] and [SW2] buttons.

Combination of SW1 and SW2	Data Value in [Watch] Window
SW1: ON, SW2: OFF	P2: 0x06, P4: 0x08
SW1: OFF, SW2: ON	P2: 0x05, P4: 0x01
SW1: ON, SW2: ON	P2: 0x03, P4: 0x00
SW1: OFF, SW2: OFF	P2: 0x07, P4: 0x09

CHAPTER 6 RELATED DOCUMENTS

Document Name			Japanese/English
78K0S/KU1+ User's Manual			PDF
78K0S/KY1+ User's Manual			PDF
78K0S/KA1+ User's Manual			PDF
78K0S/KB1+ User's Manual			PDF
78K/0S Series Instructions User's Manual			PDF
RA78K0S Assembler Package User's Manual		Language	PDF
		Operation	PDF
CC78K0S C Compiler User's Manual		Language	PDF
		Operation	PDF
PM+ Project Manager User's Manual			PDF
SM+ System Simulator Operation User's Manual			PDF
<R> Flash Programming Manual (Basic) MINICUBE2 version		78K0S/KU1+	PDF
		78K0S/KY1+	PDF
		78K0S/KA1+	PDF
		78K0S/KB1+	PDF
<R> 78K0S/Kx1+ Application Note	Sample Program Startup Guide		PDF
	Sample Program (Interrupt) External Interrupt Generated by Switch Input		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
;*******************************************************************************
;     Initialization
;*******************************************************************************
;<<History>>
;     2007.4.--   Release
;*******************************************************************************
;
;<<Overview>>
;
; This sample program initializes the microcontroller by setting functions
; such as clock frequency and the port to the input or output.
; After the initialization, three LED lights are controlled by two switches.
;
; <Principal setting contents in initialization>
;
; - Set the vector table
; - Set the stack pointer
; - Stop the watchdog timer operation
; - Set the CPU clock frequency at 2 MHz
; - Stop the low-speed internal oscillator
; - Set the ports to the input or output mode
; - Set the connection of on-chip pull-up resistors (input port only)
; - Set the output latches of the output ports
;
;
; <Switch input and LED output>
;
;  +---------------------------------------+
;  |  SW1  |  SW2  |  LED1 |  LED2 |  LED3 |
;  | (P40) | (P43) | (P20) | (P21) | (P22) |
;  |-------|-------|-------|-------|-------|
;  |  OFF  |  OFF  |  OFF  |  OFF  |  OFF  |
;  |  ON   |  OFF  |  ON   |  OFF  |  OFF  |
;  |  OFF  |  ON   |  OFF  |  ON   |  OFF  |
;  |  ON   |  ON   |  OFF  |  OFF  |  ON   |
;  +---------------------------------------+
;
;
;<<I/O port settings>>
;
;  Input:  P40, P43
;  Output: P00-P03, P20-P23, P30-P33, P41, 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    RESET_START       ;(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 ROM data table
;
;=============================================================================
XTBL1 CSEG  AT    0100H
LEDDATA:
      DB    00000011B         ;00: SW1=ON, SW2=ON  -> LED3=ON
      DB    00000101B         ;01: SW1=OFF,SW2=ON  -> LED2=ON
      DB    00000111B         ;02: Dummy
      DB    00000111B         ;03: Dummy
      DB    00000111B         ;04: Dummy
      DB    00000111B         ;05: Dummy
      DB    00000111B         ;06: Dummy
      DB    00000111B         ;07: Dummy
      DB    00000110B         ;08: SW1=ON, SW2=OFF -> LED1=ON
      DB    00000111B         ;09: SW1=OFF,SW2=OFF -> LED1,2,3=OFF

;=============================================================================
;
;     Define the memory stack area
;
;=============================================================================
XSTK  DSEG  AT    0FEE0H
STACKEND:
      DS    20H               ; Memory stack area = 32byte
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 at FF00H

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

;-------------------------------------------------------------------------------
;     Initialize the clock generators
;-------------------------------------------------------------------------------
      MOV   PPCC, #00000010B  ; The clock supplied to the peripheral hardware
(fxp) = fx/4 (= 2MHz)
      MOV   PCC,  #00000000B  ; The clock supplied to the CPU (fcpu) = fxp (=
2MHz)
      MOV   LSRCM,      #00000001B  ; Stop the low-speed internal oscillator

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

;-------------------------------------------------------------------------------
;     Initialize the port 2
;-------------------------------------------------------------------------------
      MOV   PM2,  #11110000B  ; Set P20-P23 as output mode
      MOV   P2,   #00000111B  ; Set output latches of P20-P22 as high, P23 as
low

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

;-------------------------------------------------------------------------------
;     Initialize the port 4
;-------------------------------------------------------------------------------
      MOV   PM4,  #00001001B  ; Set P40 and P43 as input mode, P41, P42 and P44-
P47 as output mode
      MOV   PU4,  #00001001B  ; Connect on-chip pull-up resistors to P40 and P43
      MOV   P4,   #00000000B  ; Set output latches of P41, P42 and P44-P47 as
low

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

;-------------------------------------------------------------------------------
```

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

;-------------------------------------------------------------------------------
;     Initialize the general-purpose register
;-------------------------------------------------------------------------------
      MOVW  HL,   #LEDDATA    ; Set the table address to the HL register

;*******************************************************************************
;
;     Main loop
;
;*******************************************************************************
MAIN_LOOP:
      MOV   A, P4             ; Read switch-input status
      AND   A, #00001001B           ; Mask the data except the switch-input
status
      MOV   L, A              ; Set switch-input status to the lower 8 bits of
the table address
      MOV   A, [HL]                 ; Read LED output data from the table
      MOV   P2, A             ; Output the LED light
      BR    $MAIN_LOOP        ; Go to the MAIN_LOOP


end
```

● main.c (C language version)

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

      NEC Electronics     78K0S/KB1+

************************************************************************
      78K0S/KB1+  Sample program
************************************************************************
      Initialization
************************************************************************
<<History>>
      2007.4.--   Release
************************************************************************

<<Overview>>

 This sample program initializes the microcontroller by setting the functions
 such as clock frequency and the port to the input or output.
 After the initialization, three LED lights are controlled by two switches.

 <Principal setting contents in initialization>

 - Stop the watchdog timer operation
 - Set the CPU clock frequency at 2 MHz
 - Stop the low-speed internal oscillator
 - Set the ports to the input or output mode
 - Set the connection of on-chip pull-up resistors (input port only)
 - Set the output latches of the output ports


 <Switch input and LED output>
  +--------------------------------------+
  |  SW1  |  SW2  |  LED1 |  LED2 |  LED3 |
  | (P40) | (P43) | (P20) | (P21) | (P22) |
  |-------|-------|-------|-------|-------|
  |  OFF  |  OFF  |  OFF  |  OFF  |  OFF  |
  |  ON   |  OFF  |  ON   |  OFF  |  OFF  |
  |  OFF  |  ON   |  OFF  |  ON   |  OFF  |
  |  ON   |  ON   |  OFF  |  OFF  |  ON   |
  +--------------------------------------+


<<I/O port settings>>

  Input:  P40, P43
  Output: P00-P03, P20-P23, P30-P33, P41, 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 */
```

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

      Initialization after RESET

************************************************************************/
void hdwinit(void){
/*----------------------------------------------------------------------
      Initialize the watchdog timer
----------------------------------------------------------------------*/
      WDTM  = 0b01110111;        /* Stop the watchdog timer operation */

/*----------------------------------------------------------------------
      Initialize the clock generators
----------------------------------------------------------------------*/
      PPCC  = 0b00000010;        /* The clock supplied to the peripheral */
                                 /* hardware (fxp) = fx/4 (= 2MHz) */
      PCC   = 0b00000000;        /* The clock supplied to the CPU (fcpu) = */
                                 /* fxp (= 2MHz) */
      LSRCM = 0b00000001;        /* Stop the low-speed internal oscillator */

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

/*----------------------------------------------------------------------
      Initialize the port 2
----------------------------------------------------------------------*/
      PM2   = 0b11110000;        /* Set P20-P23 as output mode */
      P2    = 0b00000111;        /* Set output latches of P20-P22 as high, */
                                 /* P23 as low */

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

/*----------------------------------------------------------------------
      Initialize the port 4
----------------------------------------------------------------------*/
      PM4   = 0b00001001;        /* Set P40 and P43 as input mode, P41, */
                                 /* P42 and P44-P47 as output mode */
      PU4   = 0b00001001;        /* Connect on-chip pull-up resistors to */
                                 /* P40 and P43 */
      P4    = 0b00000000;        /* Set output latches of P41, P42 and */
                                 /* P44-P47 as low */

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

/*----------------------------------------------------------------------
      Initialize the port 13
```

```
------------------------------------------------------------------------------*/
      P13   = 0b00000001;           /* Set output latches of P120-P123 as */
                                    /* high */
}

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

      Main loop

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

      const unsigned char OUTDATA[10] = {0x03,0x05,0x07,0x07,0x07,0x07,0x07,
                                         0x07,0x06,0x07};
      /* Array of LED output pattern */
      unsigned char INDATA;         /* Switch-input variable */

      while(1){
      INDATA = P4 & 0b00001001;     /* Valid switch-input status */
      P2 = OUTDATA[INDATA];         /* Read LED output data from the table */
      }
}
```

● 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
                                    (8MHz) 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	August 2007	–	–
2nd edition	September 2008	p.12	4.2 Vector Table Setting • Modification of the title of the manual (“Interrupt (preliminary name, under preparation)” → “Sample Program (Interrupt) External Interrupt Generated by Switch Input”)
		p.29	CHAPTER 5 OPERATION CHECK USING SYSTEM SIMULATOR SM+ • Modification of description in Caution ((as of April 2007) → (as of July 2008))
		pp.29 to 31	Modification of 5.1 Building the Sample Program
		p.31	5.2 Operation with SM+ • Addition of (1)
		p.35	CHAPTER 6 RELATED DOCUMENTS • Addition of Flash Programming Manual (Basic) MINICUBE2 version • Addition of the link to 78K0S/Kx1+ Sample Program Startup Guide

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