To our customers,

Old Company Name in Catalogs and Other Documents

On April 1\textsuperscript{st}, 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 1\textsuperscript{st}, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (http://www.renesas.com)
Send any inquiries to http://www.renesas.com/inquiry.
Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.

2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.

3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.

4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.

5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.

6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.

7. Renesas Electronics products are classified according to the following three quality grades: “Standard”, “High Quality”, and “Specific”. The recommended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as “Specific” without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as “Specific” or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is “Standard” unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

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

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

   “Specific”: Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.

8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.

9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.

10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.

11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.

12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

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

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

78K0R/Lx3

Sample Program (Initial Settings)

LED Lighting Switch Control

This application note describes a sample program that executes the basic initial settings of the 78K0R/Lx3 microcontroller, such as setting up the option byte, specifying the clock frequency, and setting up the I/O ports.

Target devices
78K0R/LF3 microcontroller
78K0R/LG3 microcontroller
78K0R/LH3 microcontroller

CONTENTS
CHAPTER 1 OVERVIEW ................................................................................... 3
CHAPTER 2 CIRCUIT DIAGRAM ................................................................... 4
  2.1 Circuit Diagram .................................................................................... 4
  2.2 Used Devices Other than Microcontroller ............................................ 4
CHAPTER 3 SOFTWARE ................................................................................. 5
  3.1 Included Files ........................................................................................ 5
  3.2 Internal Peripheral Functions to Be Used .......................................... 6
  3.3 Initial Settings and Operation Overview ............................................. 6
  3.4 Flow Chart .......................................................................................... 7
CHAPTER 4 SETTING METHODS .................................................................. 8
  4.1 Setting Up Option Byte ...................................................................... 9
  4.2 Setting Up Vector Table ................................................................. 14
  4.3 Setting Up Stack Pointer ................................................................. 15
  4.4 Setting Up and Controlling Watchdog Timer .................................... 16
  4.5 Setting Up Clock ............................................................................. 17
  4.6 Setting Up Ports ............................................................................. 23
  4.7 Main Processing ............................................................................... 28
CHAPTER 5 RELATED DOCUMENTS ................................................................. 30
APPENDIX A PROGRAM LIST ..................................................................... 31
APPENDIX B REVISION HISTORY ................................................................. 56
• The information in this document is current as of July, 2009. 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.
(Note 2) "NEC Electronics products" means any product developed or manufactured by or for NEC Electronics (as defined
above).
CHAPTER 1 OVERVIEW

This sample program executes the basic initial settings of the 78K0R/Lx3 microcontroller, such as setting up the option byte, specifying the clock frequency, and setting up the I/O ports. This sample program also turns on and off three LEDs using two switch inputs in the main processing that follows the completion of the initial settings.

(1) Primary initial settings

<Option byte settings>
- Disabling the watchdog timer
- Setting the internal high-speed oscillator frequency to 8 MHz
- Disabling LVI from being started by default
- Enabling on-chip debug to operate

<Settings during initialization immediately after a reset ends>
- Setting up I/O ports
- Securing a supply voltage of 2.7 V or more by using the function of low-voltage detector\textsuperscript{Note}
- Specifying that the CPU/peripheral hardware clock run on the internal high-speed oscillation clock (8 MHz)
- Stopping the X1/XT1 oscillator

\textbf{Note}  For details of the low-voltage detector, refer to the User’s Manual.

(2) Main processing operation

Lighting of the LEDs (LED1, LED2, LED3) is controlled by detecting switch inputs (SW1, SW2) with the 78K0R/Lx3 microcontroller.

<table>
<thead>
<tr>
<th>Switch Input</th>
<th>LED Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>SW1</td>
<td>SW2</td>
</tr>
<tr>
<td>OFF</td>
<td>OFF</td>
</tr>
<tr>
<td>ON</td>
<td>OFF</td>
</tr>
<tr>
<td>OFF</td>
<td>ON</td>
</tr>
<tr>
<td>ON</td>
<td>ON</td>
</tr>
</tbody>
</table>

\textbf{Caution}  For cautions when using the device, refer to the User’s Manual.
CHAPTER 2 CIRCUIT DIAGRAM

This chapter provides a circuit diagram and describes the devices used in this sample program other than the microcontroller.

2.1 Circuit Diagram

A circuit diagram is shown below.

Cautions
1. Use the microcontroller at a voltage in the range of $2.94 \text{ V} \leq V_{DD} \leq 5.5 \text{ V}$ (because a low voltage is detected within a range of $2.48 \pm 0.1 \text{ V} < V_{DD}$).
2. Make $EV_{DD}$, $AV_{DD0}$, $AV_{DD1}$, and $V_{REFOUT}/AV_{REFP}$ the same potential as $V_{DD}$.
3. Make $AV_{SS}$ the same potential as $EV_{SS}$ or $V_{SS}$ and connect it directly to GND.
4. Connect REGC to $V_{SS}$ via a capacitor (0.47 to 1 $\mu$F).
5. Handle unused pins that are not shown in the circuit diagram as follows:
   - I/O ports: Set them to output mode and leave them open (unconnected).
   - Input ports: Connect them independently to $V_{DD}$ or $V_{SS}$ via a resistor.
6. In this sample program, the P40/TOOL0 and P41/TOOL1 pins are used for on-chip debugging.

2.2 Used Devices Other than Microcontroller

The following devices are used in addition to the microcontroller:

(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 files included in the compressed file to be downloaded, internal peripheral functions of the microcontroller to be used, and initial settings and provides an operation overview of the sample program and a flow chart.

3.1 Included Files

The following table shows the files included in the compressed file to be downloaded.

<table>
<thead>
<tr>
<th>File Name</th>
<th>Description</th>
<th>Compressed (*.zip) File Included</th>
</tr>
</thead>
<tbody>
<tr>
<td>main.asm</td>
<td>Source file for hardware initialization processing and main processing of microcontroller</td>
<td><img src="image" alt="Note" /> <img src="image" alt="Note" /></td>
</tr>
<tr>
<td>main.c</td>
<td>Source file for hardware initialization processing and main processing of microcontroller</td>
<td><img src="image" alt="Note" /> <img src="image" alt="Note" /></td>
</tr>
<tr>
<td>op.asm</td>
<td>Assembler source file for setting the option byte (This file is used for setting up the watchdog timer, selecting the internal high-speed oscillator frequency, and setting up the LVI default start function.)</td>
<td><img src="image" alt="Note" /> <img src="image" alt="Note" /></td>
</tr>
<tr>
<td>78K0RLx3_sample_program.prw</td>
<td>Work space file for integrated development environment PM+</td>
<td><img src="image" alt="Note" /></td>
</tr>
<tr>
<td>78K0RLx3_sample_program.prj</td>
<td>Project file for integrated development environment PM+</td>
<td><img src="image" alt="Note" /></td>
</tr>
</tbody>
</table>

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

**Remark** ![ZIP](image) : Only the source file is included.

![PM+](image) : 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.

- P00, P01: Used for switch input.
- P00, P31, P32: Used to light LEDs.
- Low-voltage detector: Used to check that VDD is 2.7 V or more.

3.3 Initial Settings and Operation Overview

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

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.

Notes 1. The option byte is automatically referenced by the microcontroller immediately after a reset ends. In this sample program, the following settings are specified using the option byte:
- Disabling the watchdog timer
- Setting the internal high-speed oscillator frequency to 8 MHz
- Disabling LVI from being started by default
- Enabling on-chip debug to operate

2. The general-purpose registers of the 78K0R/Lx3 Series microcontrollers are configured in four register banks so that the registers used for normal processing and those used when an interrupt occurs can be changed on a bank basis in order to create an efficient program. In this sample program, only register bank 0 is used.

3. The low-voltage detector is enabled, and then the system is made to wait at least 10 μs until the low-voltage detector stabilizes.

Caution With the sample program of the C language version, the settings of register banks and stack pointer are not described in the source program (main.c) because they are made by the start-up routine. For details of the start-up routine, refer to the CC78K0R Operation User’s Manual.
CHAPTER 4 SETTING METHODS

This chapter describes how to set up the option byte, vector table, stack pointer, watchdog timer, clock frequency, and I/O ports, and provides details about the main processing.

To execute a program written in C, another program that performs ROMization to integrate the former program into the system and starts a user-created program (main function) is required. The latter program is called a startup routine. In general, a startup routine is the first program that runs after the microcontroller is reset (initialized). It initially sets up the hardware such as the CPU, memory, and I/O ports and specifies the initial settings for running the main function. In general, the startup routine, the main routine, and then subroutines are executed, and interrupts are serviced.

In the C version of this sample program, clock settings and initial settings for peripheral hardware are specified using the hdwinit function, after which the main function is executed. Therefore, the main processing is included in the main function. In the assembly language version, the microcontroller is reset (initialized), a program is executed from the RESET_START address written at address 0000H in the vector table, clock settings and initial settings for peripheral hardware are specified as by the hdwinit function in the C version, and then the main processing begins.

For details about the startup routine, refer to the chapter about the startup routine in the CC78K0R Operation User's Manual.

For how to set register, refer to the User's Manual.

For assembler instructions, refer to the 78K0R Microcontroller Instructions User's Manual.

[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 the CPU is reset. Initial settings, such as setting up the I/O ports and selecting the clock frequency are therefore basically included in the hdwinit function.

The main function is called after calling the hdwinit function, so main processing is included 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 specified 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 CC78K0R Language User's Manual and Processing to be executed first under Programming on the NEC Electronics FAQ Web page.
4.1 Setting Up Option Byte

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

1. Watchdog timer counter operation setting
2. Watchdog timer interval time setting
3. Watchdog timer window open period setting
4. LVI default start operation control
5. Internal high-speed oscillator frequency selection
6. On-chip debug operation control
Notes 1. Set the same value as 000C0H to 010C0H when the boot swap operation is used because 000C0H is replaced by 010C0H.

2. The window open period is 100% when WDSTBYON = 0, regardless the value of WINDOW1 and WINDOW0.

3. ( ): fIL = 33 kHz (MAX.)

Caution The watchdog timer continues its operation during self programming and EEPROM emulation of the flash memory. During processing, the interrupt acknowledge time is delayed. Set the overflow time and window size taking this delay into consideration.

Remarks 1. fIL: Internal low-speed oscillation clock frequency

2. The values written in red in the above figure are specified in this sample program.
### Figure 4-1-2. Format of Option Byte (2/4)

#### Address: 000C1H/010C1H

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th>FRQSEL2</th>
<th>FRQSEL1</th>
<th>LVIOFF</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### LVIOFF  Setting of LVI on power application
- **0**: LVI is ON by default (LVI default start function enabled) upon reset release (upon power application)
- **1**: LVI is OFF by default (LVI default start function stopped) upon reset release (upon power application)

#### FRQSEL2 FRQSEL1  Internal high-speed oscillator frequency
- **1 0**: 1 MHz
- **1 1**: 8 MHz
- **Other than above**: Setting prohibit

**Note**  Set the same value as 000C0H to 010C0H when the boot swap operation is used because 000C0H is replaced by 010C0H.

**Cautions**

1. Be sure to set bits 7 to 3 to “1”.
2. Even when the LVI default start function is used, if it is set to LVI operation prohibition (bit 7 (LVION) of the LVIM register = 0) by the software, it operates as follows:
   - Does not perform low-voltage detection during LVION = 0.
   - If a reset is generated while LVION = 0, LVION will be re-set to 1 when the CPU starts after reset release. There is a period when low-voltage detection cannot be performed normally, however, when a reset occurs due to WDT and illegal instruction execution. This is due to the fact that while the pulse width detected by LVI must be 200 \( \mu s \) max., LVION = 1 is set upon reset occurrence, and the CPU starts operating without waiting for the LVI stabilization time.

**Remark**  The values written in red in the above figure are specified in this sample program.

### Figure 4-1-3. Format of Option Byte (3/4)

#### Address: 000C2H/010C2H

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Note**  Be sure to set FFH to 000C2H, as these addresses are reserved areas. Also set FFH to 010C2H when the boot swap operation is used because 000C2H is replaced by 010C2H.
Figure 4-1-4. Format of Option Byte (4/4)

Address: 000C3H/010C3H

Note Set the same value as 000C3H to 010C3H when the boot swap operation is used because 000C3H is replaced by 010C3H.

Cautions 1. Bits 7 and 0 (OCDENSET and OCDERSD) can only be specified a value. Be sure to set 000010B to bits 6 to 1.

2. The value on bits 3 to 1 will be written over when the on-chip debug function is in use and thus it will become unstable after the setting. However, be sure to set the default values (0, 1, and 0) to bits 3 to 1 at setting.

Remarks 1. The values written in red in the above figure are specified in this sample program.

2. In this sample program, the option byte is set by the source file (file name: "op.asm"). Therefore, the option byte does not have to be set by the linker option of the RA78K0R.

The values specified for the option byte, above, are as follows in the program.

<table>
<thead>
<tr>
<th>OCDENSET</th>
<th>OCDERSD</th>
<th>On-chip debug operation control</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0</td>
<td>0</td>
<td>Operation disabled</td>
</tr>
<tr>
<td>0 1</td>
<td></td>
<td>Setting prohibited</td>
</tr>
<tr>
<td>1 0</td>
<td></td>
<td>Operation enabled. Erase data of the flash memory in case authentication of the on-chip debug security ID fails.</td>
</tr>
<tr>
<td>1 1</td>
<td></td>
<td>Operation enabled. Does not erase data of the flash memory in case authentication of the on-chip debug security ID fails.</td>
</tr>
</tbody>
</table>

Note Set the same value as 000C3H to 010C3H when the boot swap operation is used because 000C3H is replaced by 010C3H.

Cautions 1. Bits 7 and 0 (OCDENSET and OCDERSD) can only be specified a value. Be sure to set 000010B to bits 6 to 1.

2. The value on bits 3 to 1 will be written over when the on-chip debug function is in use and thus it will become unstable after the setting. However, be sure to set the default values (0, 1, and 0) to bits 3 to 1 at setting.

Remarks 1. The values written in red in the above figure are specified in this sample program.

2. In this sample program, the option byte is set by the source file (file name: "op.asm"). Therefore, the option byte does not have to be set by the linker option of the RA78K0R.

The values specified for the option byte, above, are as follows in the program.

XOPTB CSEG OPT_BYTE
DB 01101110B
DB 11111111B
DB 11111111B
DB 10000100B

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 with other source files (main.c).

XOPTB CSEG OPT_BYTE
DB 01101110B
DB 11111111B
DB 11111111B
DB 10000100B
END
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 000C0H in the internal ROM (flash memory), first, the CSEG pseudo instruction and OPT_BYTE attribute are used. Next, the DB pseudo instruction is used to define values that are to be set to addresses following 000C0H, 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 RA78K0R Language User's Manual.
4.2 Setting Up Vector Table

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, interrupts are not serviced, 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.

[Setting example] Setting up only the reset vector to be used when starting a reset (same as in the sample program settings)

<table>
<thead>
<tr>
<th>Address</th>
<th>Function name</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000H</td>
<td>RESET_START</td>
</tr>
<tr>
<td>00004H</td>
<td>IINIT</td>
</tr>
<tr>
<td>00006H</td>
<td>IINIT</td>
</tr>
<tr>
<td>00008H</td>
<td>IINIT</td>
</tr>
<tr>
<td>0000AH</td>
<td>IINIT</td>
</tr>
<tr>
<td>0005AH</td>
<td>IINIT</td>
</tr>
<tr>
<td>0005CH</td>
<td>IINIT</td>
</tr>
<tr>
<td>0005EH</td>
<td>IINIT</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Immediately after the reset ends, the program starts from the address (RESET_START at <1>, above) specified using the reset vector.

In this sample program, vector table addresses except 00000H are not used. IINIT is specified for all remaining vector table addresses (<2> above). If these settings are specified, even if an interrupt occurs, the processing branches to IINIT (<3> above), and then returns from the interrupt without performing processing, assuming the interrupt to be unnecessary.

[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 specified at the C source level.
- #pragma ei: The EI instruction can be specified at the C source level.
- #pragma di: The DI instruction can be specified at the C source level.
- #pragma nop: The NOP instruction can be specified at the C source level. (The clock can be advanced without operating the CPU.)
- #pragma interrupt: Interrupt functions can be specified at the C source level.

For details about the #pragma directives, refer to the chapter regarding expansion functions, in the CC78K0R Language User’s Manual.
4.3 Setting Up Stack Pointer

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 in the internal RAM. The start address of this stack area is specified using a stack pointer to allocate the stack area.

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

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

A stack area must be allocated when coding in assembly language. When coding in C language, this setting is not required, because a stack area is automatically allocated in the startup routine.

[Example] Using the first 32 bytes in the internal RAM as the stack area (same as in the sample program settings)

```
DSTK DSEG BASEP
STACKEND: ; Memory stack area = 32 bytes
    DS 20H ; Start address of the memory stack area
STACKTOP:
    ; (Omitted)
RESET_START:
    ; (Omitted)
    MOVW SP, #LOWW STACKTOP ; Initialize the stack pointer
```

By writing the above code, the first 32 bytes in the internal RAM can be allocated as the stack area.

The start address in the internal RAM varies depending on the device. The stack is allocated to the area of the following addresses:

- \( \mu \text{PD78F1500}, \mu \text{PD78F1503}, \mu \text{PD78F1506} \): FEF00H to FEF1FH
- \( \mu \text{PD78F1501}, \mu \text{PD78F1504}, \mu \text{PD78F1507} \): FE700H to FE71FH
- \( \mu \text{PD78F1502}, \mu \text{PD78F1505}, \mu \text{PD78F1508} \): FE300H to FE31FH

In this sample program, the start of the internal RAM is specified without writing an absolute address by using the DSEG pseudo instruction BASEP\textsuperscript{Note}.

\textbf{Note} For details, refer to the RA78K0R Language User's Manual.
4.4 Setting Up and Controlling Watchdog Timer

The watchdog timer is set up using the option byte. For details, see 4.1 Setting Up Option Byte.
When using the watchdog timer (when WDTON is 1), the watchdog timer is controlled using the watchdog timer enable register (WDTE). The watchdog timer counter is cleared and then starts counting again when ACH is written to WDTE. WDTE is set to 9AH by generating a reset signal.

**Note** The WDTE reset value varies depending on the value specified for WDTON of the option byte (000C0H).

<table>
<thead>
<tr>
<th>WDTON Setting</th>
<th>WDTE Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (Watchdog timer count operation disabled)</td>
<td>1AH</td>
</tr>
<tr>
<td>1 (Watchdog timer count operation enabled)</td>
<td>9AH</td>
</tr>
</tbody>
</table>

**Cautions**

1. If a value other than ACH is written to WDTE, an internal reset signal is generated.
2. If a 1-bit memory manipulation instruction is executed for WDTE, an internal reset signal is generated.
3. The value read from WDTE is 9AH/1AH (this differs from the written value (ACH)).

---

**[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.
4.5 Setting Up Clock

The CPU and the clock signal supplied to the peripheral hardware (fCLK) are generated by dividing the frequency of the main system clock (fMAIN).

(1) Selecting the clock operation mode
Select the clock operation mode by using the clock operation mode control register (CMC).

Figure 4-2-1. Format of Clock Operation Mode Control Register (CMC)

<table>
<thead>
<tr>
<th>EXCLK</th>
<th>OSCSEL</th>
<th>OSCSELS</th>
<th>AMPHS1</th>
<th>AMPHS0</th>
<th>AMPH</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
<td>AMPHS1</td>
<td>AMPHS0 AmpH</td>
</tr>
<tr>
<td>AMPH</td>
<td>Control of high-speed system clock oscillation frequency</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>2 MHz ≤ fMX ≤ 10 MHz</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>10 MHz &lt; fMX ≤ 20 MHz</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AMPHS1</td>
<td>AMPHS0 XT1 oscillator oscillation mode selection</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0 Low power consumption oscillation (default)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1 Normal oscillation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0 Ultra-low power consumption oscillation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OSCSELS</td>
<td>Subsystem clock pin operation mode XT1/P123 pin XT2/P124 pin</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>Input port mode Input port</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>XT1 oscillation mode Crystal resonator connection</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXCLK</td>
<td>OSCSEL</td>
<td>High-speed system clock pin operation mode X1/P121 pin X2/EXCLK/P122 pin</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0 Input port mode Input port</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1 X1 oscillation mode Crystal/ceramic resonator connection</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0 Input port mode Input port</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1 External clock input mode Input port External clock input</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Cautions 1. CMC can be written only once after reset release, by an 8-bit memory manipulation instruction.

2. After reset release, set CMC before X1 or XT1 oscillation is started as set by the clock operation status control register (CSC).

3. Be sure to set AMPH to 1 if the X1 clock oscillation frequency exceeds 10 MHz.

4. To use CMC with its initial value (00H), be sure to set it to 00H after releasing reset in order to prevent malfunction when a program loop occurs.

5. The XT1 oscillator is designed as a low-gain circuit for achieving low power consumption. Note the following points when designing the XT1 oscillator.
   - The pins and circuit board include parasitic capacitance. Therefore, confirm that there are no problems by performing oscillation evaluation on the circuit board to be actually used.
   - When low power consumption oscillation or ultra-low power consumption oscillation is selected, lower power consumption than when selecting normal oscillation can be achieved. However, in this case, the XT1 oscillation margin is reduced, so perform sufficient oscillation evaluation of the resonator to be used for XT1 oscillation before using the resonator.
   - Keep the wiring length between the XT1 and XT2 pins and resonator as short as possible and parasitic capacitance and wire resistance as small as possible. This is particularly important when ultra-low power consumption oscillation (AMPHS1 = 1) is selected.
   - Configure the circuit board by using material with little parasitic capacitance and wire resistance.
   - Place a ground pattern that has the same potential as VSS (if possible) around the XT1 oscillator.
   - Do not cross the signal lines between the XT1 and XT2 pins and the resonator with other signal lines. Do not route the signal lines near a signal line through which a high fluctuating current flows.
   - Moisture absorption by the circuit board and condensation on the board in a highly humid environment may cause the impedance between the XT1 and XT2 pins to drop and disable oscillation. When using the circuit board in such an environment, prevent the circuit board from absorbing moisture by taking measures such as coating the circuit board.
   - Coat the surface of the circuit board by using material that does not generate capacitance or leakage between the XT1 and XT2 pins.

6. Be sure to clear bits 5 and 3 to “0”.

Remarks 1. The values written in red in the above figure are specified in this sample program.

2. $f_{MX}$: High-speed system clock frequency
(2) Controlling the operations of the high-speed system clock, internal high-speed oscillation clock, and subsystem clock

Control the high-speed system clock, internal high-speed oscillation clock, and subsystem clock by using the clock operation status control register (CSC).

Figure 4-2-2. Format of Clock Operation Status Control Register (CSC)

Cautions
1. After reset release, set the clock operation mode control register (CMC) before starting X1 oscillation as set by MSTOP or XT1 oscillation as set by XTSTOP.
2. To start X1 oscillation as set by MSTOP, check the oscillation stabilization time of the X1 clock by using the oscillation stabilization time counter status register (OSTC).
3. Do not stop the clock selected for the CPU/peripheral hardware clock \( f_{CLK} \) with the OSC register.
4. The setting of the flags of the register to stop clock oscillation (invalidate the external clock input) and the condition before clock oscillation is to be stopped are as follows.

<table>
<thead>
<tr>
<th>Clock</th>
<th>Condition Before Stopping Clock (Invalidating External Clock Input)</th>
<th>Setting of CSC Register Flags</th>
</tr>
</thead>
<tbody>
<tr>
<td>X1 clock</td>
<td>CPU and peripheral hardware clocks operate with a clock other than the high-speed system clock.  ( \text{CLS} = 0 \text{ and } MCS = 0 )  ( \text{CLS} = 1 )</td>
<td>MSTOP = 1</td>
</tr>
<tr>
<td>External main system clock</td>
<td>CPU and peripheral hardware clocks operate with a clock other than the subsystem clock. ( \text{CLS} = 0 )</td>
<td>XTSTOP = 1</td>
</tr>
<tr>
<td>Subsystem clock</td>
<td>CPU and peripheral hardware clocks operate with a clock other than the subsystem clock. ( \text{CLS} = 0 )</td>
<td>XTSTOP = 1</td>
</tr>
<tr>
<td>Internal high-speed oscillation clock</td>
<td>CPU and peripheral hardware clocks operate with a clock other than the internal high-speed oscillator clock. ( \text{CLS} = 0 \text{ and } MCS = 1 ) ( \text{CLS} = 1 )</td>
<td>HIOSTOP = 1</td>
</tr>
</tbody>
</table>

5. Be sure to clear bits 5 to 1 to “0”.

Application Note U2002SEJ1V0AN 19
(3) Controlling the booster of the flash memory for high-speed operation
Control the booster of the flash memory for high-speed operation by using the operation speed mode control register (OSMC).

**Figure 4-2-3. Format of Operation Speed Mode Control Register (OSMC)**

<table>
<thead>
<tr>
<th>RTCLPC</th>
<th>FLPC</th>
<th>FSEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0001</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0100</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0101</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1000</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1001</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1100</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1101</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1110</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1111</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

**Note** Refer to the chapter of standby function of the User’s Manual for the peripheral functions whose operations are enabled.

**Cautions**

1. Write “1” to FSEL before the following two operations.
   - Changing the clock prior to dividing fCLK to a clock other than fIH.
   - Operating the DMA controller.

2. The CPU waits when “1” is written to the FSEL flag. The wait time is 15 μs to 20 μs (target) when fCLK = fIH, and 30 μs to 40 μs (target) when fCLK = fIH/2. However, counting the oscillation stabilization time of fx can continue even while the CPU is waiting.

3. To increase fCLK to 10 MHz or higher, set FSEL to “1”, then change fCLK after two or more clocks have elapsed.

4. Confirm that the clock is operating at 10 MHz or less before setting FSEL = 0.

5. To shift to STOP mode while VDD ≤ 2.7 V, set FSEL = 0 after setting fCLK to 10 MHz or less.

6. The HALT mode current when operating on the subsystem clock can be reduced by setting RTCLPC to 1. However, the clock cannot be supplied to peripheral functions except the real-time counter in the subsystem clock HALT mode. Set bit 7 (RTCEN) of PER0 to 1 and bits 0 to 6 of PER0 to 0 before setting the subsystem clock HALT mode.

7. Once FLPC has been set from 0 to 1, setting it back to 0 from 1 other than by reset is prohibited.

8. When setting FSEL to “1”, do so while RMC = 00H. When setting FLPC to “1”, do so while RMC = 5AH.

9. Be sure to clear bits 6 to 2 to “0”.
## (4) Setting the CPU/peripheral hardware clock

Select the CPU/peripheral hardware clock and a division ratio by using the system clock control register (CKC).

**Figure 4-2-4. Format of System Clock Control Register (CKC)**

<table>
<thead>
<tr>
<th>CKC</th>
<th>CLS</th>
<th>CSS</th>
<th>MCS</th>
<th>MCM0</th>
<th>SDIV</th>
<th>MDIV2</th>
<th>MDIV1</th>
<th>MDIV0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSS</td>
<td>MCM0</td>
<td>SDIV</td>
<td>MDIV2</td>
<td>MDIV1</td>
<td>MDIV0</td>
<td>Selection of CPU/peripheral hardware clock (fCLK)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>×</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>fIH</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>×</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>fIH/2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>×</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>fIH/2^1 Note 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>×</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>fIH/2^1 Note 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>×</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>fIH/2^1 Note 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>×</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>fIH/2^1 Note 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>×</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>fMX</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>×</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>fMX/2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>×</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>fMX/2^1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>×</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>fMX/2^1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>×</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>fMX/2^1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>fSUB/2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Other than above Setting prohibited

<table>
<thead>
<tr>
<th>MCS</th>
<th>Status of Main system clock (fMAIN)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Internal high-speed oscillation clock (fIH)</td>
</tr>
<tr>
<td>1</td>
<td>High-speed system clock (fMX)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CLS</th>
<th>Status of CPU/peripheral hardware clock (fCLK)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Main system clock (fMAIN)</td>
</tr>
<tr>
<td>1</td>
<td>Subsystem clock (fSUB)</td>
</tr>
</tbody>
</table>

**Notes**
1. Setting is prohibited when $fIH = 1$ MHz.
2. Setting is prohibited when $fMX < 4$ MHz.
3. Changing the value of the MCM0 bit is prohibited while CSS is set to 1.
4. Bits 7 and 5 are read-only.

**Remarks**
1. $fIH$: Internal high-speed oscillation clock frequency
   $fMX$: High-speed system clock frequency
   $fSUB$: Subsystem clock frequency
2. ×: don't care
3. The values written in red in the above figure are specified in this sample program.
An example of writing the values specified in (1) to (4) in the program is shown below.

- Assembly language

```assembly
MOV CMC, #00000000B
MOV CSC, #11000000B
MOV OSMC, #10000000B
MOV CKC, #00001000B
```

- C language

```c
CMC = 0b00000000;
CSC = 0b11000000;
OSMC = 0b10000000;
CKC = 0b00001000;
```
4.6 Setting Up Ports

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

<table>
<thead>
<tr>
<th>78K0R/LG3</th>
<th>78K0R/LF3</th>
<th>78K0R/LH3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port 0</td>
<td>P00 to P02</td>
<td>P00 to P02</td>
</tr>
<tr>
<td>Port 1</td>
<td>P10 to P15</td>
<td>P10 to P16</td>
</tr>
<tr>
<td>Port 2</td>
<td>P20 to P26</td>
<td>P20 to P27</td>
</tr>
<tr>
<td>Port 3</td>
<td>P30 to P33</td>
<td>P30 to P34</td>
</tr>
<tr>
<td>Port 4</td>
<td>P40, P41</td>
<td>P40, P41</td>
</tr>
<tr>
<td>Port 5</td>
<td>P50 to P57</td>
<td>P50 to P57</td>
</tr>
<tr>
<td>Port 6</td>
<td>–</td>
<td>P60, P61</td>
</tr>
<tr>
<td>Port 7</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Port 8</td>
<td>–</td>
<td>P80 to P82</td>
</tr>
<tr>
<td>Port 9</td>
<td>P90 to P92</td>
<td>P90 to P97</td>
</tr>
<tr>
<td>Port 10</td>
<td>P100</td>
<td>P100</td>
</tr>
<tr>
<td>Port 11</td>
<td>P110, P111</td>
<td>P110, P111</td>
</tr>
<tr>
<td>Port 12</td>
<td>P120 to P124</td>
<td>P120 to P124</td>
</tr>
<tr>
<td>Port 13</td>
<td>P130</td>
<td>P130</td>
</tr>
<tr>
<td>Port 14</td>
<td>P140 to P147</td>
<td>P140 to P147</td>
</tr>
<tr>
<td>Port 15</td>
<td>P157</td>
<td>P150 to P152, P157</td>
</tr>
</tbody>
</table>

(1) Specifying ports as input or output ports

The PMxx registers are used to specify whether ports are used as input ports or output ports. Ports are specified as input ports immediately after a reset ends.

The PMxx format is described, taking the PM0 register as an example.

Figure 4-3-1. Format of Port Mode Register 0 (PM0)

<table>
<thead>
<tr>
<th></th>
<th>PM02</th>
<th>PM01</th>
<th>PM00</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

PM0n (n = 0 to 2)  Selection of P0n (n = 0 to 2) pin I/O mode

| 0   | Output mode (output buffer on) |
| 1   | Input mode (output buffer off)  |

Caution  Be sure to set bits 7 to 3 to “1”.
(2) Setting up the output latches of output ports
The Pxx registers are used to set up the output latches of output ports to high level or low level. The output latches of output ports are set to low-level output immediately after a reset ends.
The Pxx format is described, taking the P0 register as an example.

Figure 4-3-2. Format of Port Register 0 (P0)

<table>
<thead>
<tr>
<th>P0n (n = 0 to 2)</th>
<th>Output data control (in output mode)</th>
<th>Input data read (in input mode)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Output 0.</td>
<td>Input low level.</td>
</tr>
<tr>
<td>1</td>
<td>Output 1.</td>
<td>Input high level.</td>
</tr>
</tbody>
</table>

Caution  Be sure to clear bits 7 to 3 to “0”.

(3) Specifying the connections of on-chip pull-up resistors to input ports
The PUxx registers are used to specify whether on-chip pull-up resistors are connected to input ports. On-chip pull-up resistors are not connected immediately after a reset ends.
The PUxx format is described, taking the PU0 register as an example.

Figure 4-3-3. Format of Pull-up Resistor Option Register 0 (PU0)

<table>
<thead>
<tr>
<th>PU0n (n = 0 to 2)</th>
<th>Connection of P0n (n = 0 to 2) pin on-chip pull-up resistor</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>On-chip pull-up resistor is not connected.</td>
</tr>
<tr>
<td>1</td>
<td>On-chip pull-up resistor is connected.</td>
</tr>
</tbody>
</table>

Caution  Be sure to clear bits 7 to 3 to “0”.
[Example 1] P0 is set up as follows to use it for switch input:
- P00 and P01 are specified as input ports.
- An on-chip pull-up resistor is connected to P00 and P01.
  (Same as in the sample program settings)

PM0

<table>
<thead>
<tr>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
</table>

Selection of P00 and P01 pin I/O modes

1 Input mode

Note Unused pins are assumed to be output port pins.

PU0

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
</table>

Connection of on-chip pull-up resistors to P00 and P01 pins

1 On-chip pull-up resistor is connected.

In this sample program, switch input signals (SW1 and SW2) are used as low-active signals. Therefore, a low-level signal (0) is input to the switch input ports (P00 and P01) when the switches are turned on and a high-level signal (1) is input when the switches are turned off.

The relationship between the switch input signals (SW1 and SW2) and switch input ports (P00 and P01) is as follows:

<table>
<thead>
<tr>
<th>Switch Input</th>
<th>Switch Input Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>SW1</td>
<td>SW2</td>
</tr>
<tr>
<td>On</td>
<td>On</td>
</tr>
<tr>
<td>Off</td>
<td>On</td>
</tr>
<tr>
<td>On</td>
<td>Off</td>
</tr>
<tr>
<td>Off</td>
<td>Off</td>
</tr>
<tr>
<td></td>
<td>P00</td>
</tr>
<tr>
<td></td>
<td>P01</td>
</tr>
<tr>
<td>On</td>
<td>0</td>
</tr>
<tr>
<td>Off</td>
<td>1</td>
</tr>
<tr>
<td>On</td>
<td>0</td>
</tr>
<tr>
<td>Off</td>
<td>1</td>
</tr>
</tbody>
</table>

These settings are specified in the program as follows:

[Assembly language]

```
MOV P0, #00000000B
MOV PM0, #11111111B
MOV PU0, #00000011B
```

[C language]

```
PO = 0b00000000;
PM0 = 0b11111111;
PU0 = 0b00000011;
```
[Example 2] P3 is set up as follows to turn on the LEDs:

- P30, P31, and P32 are specified as output ports.
- The P30, P31, and P32 output latches are set to high-level output.

(Same as in the sample program settings)

**PM3**

- 78K0R/LF3

<table>
<thead>
<tr>
<th>1</th>
<th>1</th>
<th>1</th>
<th>0&lt;sup&gt;Note&lt;/sup&gt;</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>P30, P31, and P32 pin I/O modes</td>
<td>Output mode</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- 78K0R/LG3 and 78K0R/LH3

<table>
<thead>
<tr>
<th>1</th>
<th>1</th>
<th>1</th>
<th>0&lt;sup&gt;Note&lt;/sup&gt;</th>
<th>0&lt;sup&gt;Note&lt;/sup&gt;</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>P30, P31, and P32 pin I/O modes</td>
<td>Output mode</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note** Unused pins are assumed to be output port pins.

**P3**

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selection of P30, P31, and P32 pin output latch level</td>
<td>High-level output</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In this sample program, the signals used to turn on the LEDs (LED1, LED2, and LED3) are used as active-low signals. Therefore, the LEDs are turned on if 0 is output from the LED output ports (P30, P31, and P32) or turned off if 1 is output from the ports.

The relationship between the values of the LED output ports (P30, P31, and P32) and whether the LEDs are turned on (LED1, LED2, and LED3) is as follows:

<table>
<thead>
<tr>
<th>LED Output Port</th>
<th>LED Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>P30 P31 P32</td>
<td>LED1</td>
</tr>
<tr>
<td>0 1 1</td>
<td>On</td>
</tr>
<tr>
<td>1 0 1</td>
<td>Off</td>
</tr>
<tr>
<td>1 1 0</td>
<td>Off</td>
</tr>
<tr>
<td>1 1 1</td>
<td>Off</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LED Output Port</th>
<th>LED Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>P30 P31 P32</td>
<td>LED2</td>
</tr>
<tr>
<td>0 1 1</td>
<td>Off</td>
</tr>
<tr>
<td>1 0 1</td>
<td>On</td>
</tr>
<tr>
<td>1 1 0</td>
<td>Off</td>
</tr>
<tr>
<td>1 1 1</td>
<td>Off</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LED Output Port</th>
<th>LED Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>P30 P31 P32</td>
<td>LED3</td>
</tr>
<tr>
<td>0 1 1</td>
<td>Off</td>
</tr>
<tr>
<td>1 0 1</td>
<td>Off</td>
</tr>
<tr>
<td>1 1 0</td>
<td>On</td>
</tr>
<tr>
<td>1 1 1</td>
<td>Off</td>
</tr>
</tbody>
</table>
These settings are specified in the program as follows:

- **78K0R/LF3**
  [Assembly language]
  
  ```
  MOV P3, #00000111B
  MOV PM3, #11110000B
  ```

  [C language]
  
  ```
  P3 = 0b00000111;
  PM3 = 0b11110000;
  ```

- **78K0R/LG3 and 78K0R/LH3**
  [Assembly language]
  
  ```
  MOV P3, #00000111B
  MOV PM3, #11100000B
  ```

  [C language]
  
  ```
  P3 = 0b00000111;
  PM3 = 0b11100000;
  ```
4.7 Main Processing

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

<1> Read data from P0.
<2> Among the read eight bits, clear the bits other than those for the switch input ports (P00 and P01) to 0.
<3> Read the data to output in accordance with the combination of the P00 and P01 input levels from addresses 01000H to 01003H (in the LEDDATA table).
<4> Output the read output data to P3.

By performing operations <1> and <2>, only the combination of the inputs of the switches (SW1 and SW2) connected to P00 and P01 can be determined. In this sample program, the signals from the switches are used as active-low signals. Therefore, a low-level signal (0) is input to P00 and P01 if the switches are turned on and a high-level signal (1) is input to P00 and P01 if the switches are turned off.

The output data (bits 2 to 0) is read from address 01000H (a) above, 01001H (b) above, 01002H (c) above, or 01003H (d) above.

The address of the A register after the operation is 00H (a) above, 01H (b) above, 02H (c) above, or 03H (d) above.

The relationship between the values of the LED output ports (P30, P31, and P32) and whether the LEDs are turned on (LED1, LED2, and LED3) is as follows:

<table>
<thead>
<tr>
<th>LED Output Port</th>
<th>LED Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>P30</td>
<td>P31</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
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 specified as an array.

```c
void main(void)
{
    const unsigned char aLedOut[4] = {0b00000011, 0b00000101, 0b00000110, 0b00000111}; /* Table for turning on the LEDs */
    unsigned char ucSwitchBuffer; /* Switch input data storage area */

    while(1){
        /* Acquire valid switch information */
        ucSwitchBuffer = ( P0 & 0b00000011 );

        /* Read the data to display from the table and display */
        P3 = ( aLedOut[ucSwitchBuffer] & 0b00000111 );
    }
}
```

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

<table>
<thead>
<tr>
<th>Switch Input</th>
<th>P00, P01</th>
<th>ucSwitchBuffer</th>
<th>aLedOut</th>
<th>LED Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>SW1 = ON, SW2 = ON</td>
<td>P00 = 0, P01 = 0</td>
<td>0b00000000</td>
<td>0b00000011</td>
<td>Turn on only LED3.</td>
</tr>
<tr>
<td>SW1 = OFF, SW2 = ON</td>
<td>P00 = 1, P01 = 0</td>
<td>0b00000001</td>
<td>0b00000101</td>
<td>Turn on only LED2.</td>
</tr>
<tr>
<td>SW1 = ON, SW2 = OFF</td>
<td>P00 = 0, P01 = 1</td>
<td>0b00000010</td>
<td>0b00000110</td>
<td>Turn on only LED1.</td>
</tr>
<tr>
<td>SW1 = OFF, SW2 = OFF</td>
<td>P00 = 1, P01 = 1</td>
<td>0b00000011</td>
<td>0b00000111</td>
<td>Turn off all LEDs.</td>
</tr>
</tbody>
</table>
### CHAPTER 5 RELATED DOCUMENTS

<table>
<thead>
<tr>
<th>Document Name</th>
<th>English</th>
</tr>
</thead>
<tbody>
<tr>
<td>78K0R/Lx3 User's Manual</td>
<td>PDF</td>
</tr>
<tr>
<td>78K0R Microcontroller Instructions User's Manual</td>
<td>PDF</td>
</tr>
<tr>
<td>RA78K0R Assembler Package User's Manual</td>
<td>Language PDF</td>
</tr>
<tr>
<td></td>
<td>Operation PDF</td>
</tr>
<tr>
<td>CC78K0R C Compiler User's Manual</td>
<td>Language PDF</td>
</tr>
<tr>
<td></td>
<td>Operation PDF</td>
</tr>
<tr>
<td>PM+ Project Manager User's Manual</td>
<td>PDF</td>
</tr>
</tbody>
</table>
As a program list example, the 78K0R/LH3 microcontroller source program is shown below.

- main.asm (assembly language version)
  
  ;*******************************************************************************
  ;
  ; NEC Electronics     78K0R/LH3 Series
  ;
  ;*******************************************************************************
  ; 78K0R/LH3 Series     Sample Program (Initial Settings)
  ;*******************************************************************************
  ; LED Lighting Switch Control
  ;*******************************************************************************
  ;<<History>>
  ; 2009.3.-- Release
  ;*******************************************************************************
  ;*******************************************************************************

  ;<<Overview>>
  ; This sample program initializes the microcontroller by specifying settings such as
  ; selecting the clock frequency and setting up I/O ports. After the initialization,
  ; three LED lights are controlled by two switches in the main loop.

  ; <Primary initial settings>
  ; (Option byte settings)
  ; - Disabling the watchdog timer
  ; - Setting the internal high-speed oscillator frequency to 8 MHz
  ; - Disabling LVI from being started by default
  ; - Enabling on-chip debug to operate
  ; (Settings during initialization immediately after a reset ends)
  ; - Setting up I/O ports
  ; - Securing a supply voltage of 2.7 V or more by using the function of low-voltage detector
  ; - Specifying that the CPU/peripheral hardware clock run on the internal high-speed oscillation clock (8 MHz)
  ; - Stopping the X1/XT1 oscillator

  ; <Switch input and LED status>
<table>
<thead>
<tr>
<th>Switch 1</th>
<th>Switch 2</th>
<th>LED1</th>
<th>LED2</th>
<th>LED3</th>
</tr>
</thead>
<tbody>
<tr>
<td>(P00)</td>
<td>(P01)</td>
<td>(P30)</td>
<td>(P31)</td>
<td>(P32)</td>
</tr>
</tbody>
</table>

* 0 is input to the ports if the switches are turned on and 1 is input to the ports if the switches are turned off.

* The LEDs are turned off if 1 is output from the ports or turned on if 0 is output from the ports.

```
; I/O port settings
Input: P00, P01
Output: P30 to P32
* Set all unused ports that can be specified as output ports as output ports.

;*******************************************************************************

; Vector table

;===============================================================================

TVECT1 CSEG AT 00000H
DW  RESET_START       ; 00000H  RESET input, POC, LVI, WDT, TRAP
TVECT2 CSEG AT 00004H
DW  IINIT             ; 00004H  INTWDTI
DW  IINIT             ; 00006H  INTLVI
DW  IINIT             ; 00008H  INTP0
DW  IINIT             ; 0000AH  INTP1
DW  IINIT             ; 0000CH  INTP2
DW  IINIT             ; 0000EH  INTP3
DW  IINIT             ; 00010H  INTP4
DW  IINIT             ; 00012H  INTP5
```
APPENDIX A PROGRAM LIST

DW IINIT ; 00014H INTST3
DW IINIT ; 00016H INTSR3
DW IINIT ; 00018H INTSRE3
DW IINIT ; 0001AH INTDMA0
DW IINIT ; 0001CH INTDMA1
DW IINIT ; 0001EH INTST0/INTCSI00
DW IINIT ; 00020H INTSR0/INTCSI01
DW IINIT ; 00022H INTSRE0
DW IINIT ; 00024H INTST1/INTCSI10/INTIIC10
DW IINIT ; 00026H INTSR1
DW IINIT ; 00028H INTSRE1
DW IINIT ; 0002AH INTIICA
DW IINIT ; 0002CH INTTM00
DW IINIT ; 0002EH INTTM01
DW IINIT ; 00030H INTTM02
DW IINIT ; 00032H INTTM03
DW IINIT ; 00034H INTAD
DW IINIT ; 00036H INTRTC
DW IINIT ; 00038H INTRTC1
DW IINIT ; 0003AH INTKR
DW IINIT ; 0003CH INTST2/INTCSI20/INTIIC20
DW IINIT ; 0003EH INTSR2
DW IINIT ; 00040H INTSRE2
DW IINIT ; 00042H INTTM04
DW IINIT ; 00044H INTTM05
DW IINIT ; 00046H INTTM06
DW IINIT ; 00048H INTTM07
DW IINIT ; 0004AH INTP6
DW IINIT ; 0004CH INTP7
DW IINIT ; 0004EH INTP8
DW IINIT ; 00050H INTP9
DW IINIT ; 00052H INTP10
DW IINIT ; 00054H INTP11
DW IINIT ; 00056H INTTM10
DW IINIT ; 00058H INTTM11
DW IINIT ; 0005AH INTTM12
DW IINIT ; 0005CH INTTM13
DW IINIT ; 0005EH INTMD
APPENDIX A  PROGRAM LIST

;===============================================================================
; Define the ROM data table
;===============================================================================
XTBL1CSEG  AT  01000H
LEDDATA:
  DB  00000011B  ; [01000H] Switch 1 turned on, switch 2 turned on: Turn on LED3
  DB  00000101B  ; [01001H] Switch 1 turned off, switch 2 turned on: Turn on LED2
  DB  00000110B  ; [01002H] Switch 1 turned on, switch 2 turned off: Turn on LED1
  DB  00000111B  ; [01003H] Switch 1 turned off, switch 2 turned off: Turn off all LEDs

;===============================================================================
; Define the memory stack area
;===============================================================================
DSTK DSEG  BASEP
STACKEND:
  DS  20H  ; Memory stack area = 32 bytes
STACKTOP:   ; Start address of the memory stack area

;*******************************************************************************
; Servicing interrupts by using unnecessary interrupt sources
;*******************************************************************************
XMAINCSEG  UNIT
IINIT:
  ; If an unnecessary interrupt occurred, the processing branches to this line.
  ; The processing then returns to the initial original processing because no processing is performed here.
  RETI
RESET_START:

;-----------------------------------------------------------------------------
;  Disable interrupts
;-----------------------------------------------------------------------------
DI   ; Disable interrupts

;-----------------------------------------------------------------------------
;  Set up the register bank
;-----------------------------------------------------------------------------
SEL   RB0 ; Set up the register bank

;-----------------------------------------------------------------------------
;  Initialize the stack pointer
;-----------------------------------------------------------------------------
MOVW   SP, #LOWW STACKTOP ; Initialize the stack pointer

;-----------------------------------------------------------------------------
;  Specify the I/O port
;-----------------------------------------------------------------------------
CALL   !!SINIPORT ; Set all ports that can be specified as output ports as output ports.

;-----------------------------------------------------------------------------
;  Low-voltage detection
;-----------------------------------------------------------------------------
CALL   !!SINILVI ; Securing a supply voltage of 2.7 V or more

;-----------------------------------------------------------------------------
;  Specify the clock frequency
;-----------------------------------------------------------------------------
CALL   !!SINICLK ; Operating internal high-speed oscillation clock at 8 MHz
; ; Initialize the general-purpose register
; ;--------------------------------------------------------
; MOVW HL, #LEDATA ; Specify the table address for turning on the LEDs

; ; Enable interrupts
; ; (To use interrupts, enable interrupts here.)
; ;--------------------------------------------------------
; EI ; To enable interrupts,
; ; Delete ";" before EI.

BR MAIN_LOOP ; Go to the main loop

; ; I/O port setting
;
; ;*************************************************************
SINIPORT:

; Specify the digital I/O

; ;*************************************************************
MOV ADPC, #00010000B ; A/D port configuration register
; ;|+++++++-- ADPC4 to ADPC0
; ;|| [Analog input (A)/digital I/O (D) switching]
; ;||| +------------- ANI15/P157
; ;||| |++++++++-- ANI10 to ANI8/P152 to P150
; ;||| ||||+++++++-- ANI7 to ANI0/P27 to P20
; ;||| 00000: AAAAAAAAAA
; ;||| 00001: AAAAAAAAAAADD
; ;||| 00010: AAAAAAAAAADD
; ;||| 00011: AAAAAAAADD
; ;||| 00100: AAAAAAADDD
; ;||| 00101: AAAAAAADDDD
; ;||| 00110: AAAAAAADDDDD
; ;||| 00111: AAAAAAADDDDDD
; ;||| 01000: AAAADDDDDDDD
; Initialize port 0

MOV P0, #00000000B ; Set the P00 to P02 output latches to low level
MOV PM0, #11111011B ; Specify P00 and P01 as input ports
                   ; Specify P02 as an output port
MOV PU0, #00000011B ; Connect on-chip pull-up resistors to P00 and P01
                   ; Connect an on-chip pull-up resistor to P02
                   ; P00: Use for switch 1 input
                   ; P01: Use for switch 2 input
                   ; P02: Unused

; Initialize port 1

MOV P1, #00000000B ; Set the P10 to P17 output latches to low level
MOV PM1, #00000000B ; Specify P10 to P17 as output ports
                   ; P10 to P17: Unused

; Initialize port 2

MOV P2, #00000000B ; Set the P20 to P27 output latches to low level
MOV PM2, #00000000B ; Specify P20 to P27 as output ports
                   ; P20 to P27: Unused

; Initialize port 3

MOV P3, #00000111B ; Set the P30 to P32 output latches to high level
                   ; Set the P33 and P34 output latches to low level
MOV PM3, #11100000B ; Specify P30 to P34 as output ports
                   ; P30: Use for turning on LED1
; P31: Use for turning on LED2
; P32: Use for turning on LED3
; P33 and P34: Unused

;-----------------------------------------------------------------------------
; Initialize port 4
;-----------------------------------------------------------------------------
MOV P4, #00000000B ; Set the P40 and P41 output latches to low level
MOV PM4, #11111100B ; Specify P40 and P41 as output ports
; P40 and P41: Unused

;-----------------------------------------------------------------------------
; Initialize port 5
;-----------------------------------------------------------------------------
MOV P5, #00000000B ; Set the P50 to P57 output latches to low level
MOV PM5, #00000000B ; Specify P50 to P57 as output ports
; P50 to P57: Unused

;-----------------------------------------------------------------------------
; Initialize port 6
;-----------------------------------------------------------------------------
MOV P6, #00000000B ; Set the P60 and P61 output latches to low level
MOV PM6, #11111100B ; Specify P60 and P61 as output ports
; P60 and P61: Unused

;-----------------------------------------------------------------------------
; Initialize port 7
;-----------------------------------------------------------------------------
MOV P7, #00000000B ; Set the P70 to P77 output latches to low level
MOV PM7, #00000000B ; Specify P70 to P77 as output ports
; P70 to P77: Unused

;-----------------------------------------------------------------------------
; Initialize port 8
;-----------------------------------------------------------------------------
MOV P8, #00000000B ; Set the P80 to P87 output latches to low level
MOV PM8, #00000000B ; Specify P80 to P87 as output ports
; P80 to P87: Unused
;Initialize port 9
MOV P9, #00000000B ; Set the P90 to P97 output latches to low level
MOV PM9, #00000000B ; Specify P90 to P97 as output ports
; P90 to P97: Unused

;Initialize port 10
MOV P10, #00000000B ; Set the P100 to P102 output latches to low level
MOV PM10, #11111000B ; Specify P100 to P102 as output ports
; P100 to P102: Unused

;Initialize port 11
MOV P11, #00000000B ; Set the P110 and P111 output latches to low level
MOV PM11, #11111100B ; Specify P110 and P111 as output ports
; P110 and P111: Unused

;Initialize port 12
MOV P12, #00000000B ; Set the P120 output latch to low level
MOV PM12, #11111110B ; Specify P120 as an output port
; P120 to P124: Unused
; * P121 to P124 are input-only ports.

;Initialize port 14
MOV P14, #00000000B ; Set the P140 to P147 output latches to low level
MOV PM14, #00000000B ; Specify P140 to P147 as output ports
; P140 to P147: Unused

;Initialize port 15
MOV P15, #00000000B ; Set the P150 to P152, and P157 output latches to low level
MOV PM15, #01111000B ; Specify P150 to P152, P157 as output ports
; P150 to P152, and P157: Unused
RET

;*******************************************************************************
;*******************************************************************************
; Low-voltage detection
;*******************************************************************************
; Secure a supply voltage of 2.7 V or more by using the function of low-voltage detector.
;*******************************************************************************

SINILVI:
; Set up the low-voltage detector
SET1 LVIMK ; Disable the INTLVI interrupt
CLR1 LVISEL ; Specify VDD as the detection voltage
MOV LVIS, #00001001B ; Low-voltage detection level select register
; ||||++++------ LVIS3 to LVIS0
; |||| [Detection level]
; |||| 0000: VLVI0 (4.22 ± 0.1 V)
; |||| 0001: VLVI1 (4.07 ± 0.1 V)
; |||| 0010: VLVI2 (3.92 ± 0.1 V)
; |||| 0011: VLVI3 (3.76 ± 0.1 V)
; |||| 0100: VLVI4 (3.61 ± 0.1 V)
; |||| 0101: VLVI5 (3.45 ± 0.1 V)
; |||| 0110: VLVI6 (3.30 ± 0.1 V)
; |||| 0111: VLVI7 (3.15 ± 0.1 V)
; |||| 1000: VLVI8 (2.99 ± 0.1 V)
; |||| 1001: VLVI9 (2.84 ± 0.1 V)
; |||| 1010: VLVI10 (2.68 ± 0.1 V)
; |||| 1011: VLVI11 (2.53 ± 0.1 V)
; |||| 1100: VLVI12 (2.38 ± 0.1 V)
; |||| 1101: VLVI13 (2.22 ± 0.1 V)
; |||| 1110: VLVI14 (2.07 ± 0.1 V)
; |||| 1111: VLVI15 (1.91 ± 0.1 V)
;++++----------- Be sure to set 0
CLR1 LVIMD ; Specify that an interrupt signal is generated when a low voltage is detected
SET1 LVION ; Enable low-voltage detection

; Make the system wait until the low-voltage detector stabilizes (10 us or more)
MOV B, #10 ; Specify the number of counts
HRES100:
  NOP ; (1 clk)
  DEC B ; (1 clk)
  BNZ $HRES100 ; Has the wait period ended? No, (2 clk/4 clk)

; Make the system wait until VLVI is less than or equal to VDD
HRES300:
  NOP
  BT LVIF, $HRES300 ; VDD < VLVI? Yes,
  CLR1 LVION ; Stop the low-voltage detector
  RET

;******************************************************************************
; Specify the clock frequency
;******************************************************************************
SINICLK:
  MOV CMC, #00000000B ; Clock operation mode
  ;|||+------ AMPH
  ;||| [Control of high-speed system clock oscillation frequency]
  ;|| 0: 2 MHz ≤ fMX < 10 MHz
  ;|| 1: 10 MHz < fMX ≤ 20 MHz
  ;||+------- AMPHS1, AMPHS0
  ;|| [XT1 oscillator oscillation mode selection]
  ;|| 00: Low power consumption oscillation (default)
  ;|| 01: Normal oscillation
  ;|| 10: Ultra-low power consumption oscillation
  ;|| 11: Ultra-low power consumption oscillation
  ;||+------- Be sure to set 0.
APPENDIX A   PROGRAM LIST

;+------------ OSCSELS  
;[Subsystem clock pin operation mode]
;  0: Input port mode
;  1: XT1 oscillation mode
;  Be sure to set 0
;++------------ EXCLK/OSCSEL  
;[High-speed system clock pin operation mode]
;  00: Input port mode
;  01: X1 oscillation mode
;  10: Input port mode
;  11: External clock input mode

MOV CSC, #11000000B ; Clock operation status control
;++-------- HIOSTOP  
;[Internal high-speed oscillation clock operation control]
;  0: Internal high-speed oscillator operating
;  1: Internal high-speed oscillator stopped
;  Be sure to set 0
;++-------- XTSTOP  
;[Subsystem clock operation control]
;  0: XT1 oscillator operating
;  1: XT1 oscillator stopped
;++-------- MSTOP  
;[High-speed system clock operation control]
;  0: X1 oscillator operating
;  1: X1 oscillator stopped

MOV OSMC, #10000000B ; Operation speed mode
;++-------- FSEL/FLPC  
;[fCLK frequency selection]
;  00: Operates at a frequency of 10 MHz or less (default)
;  01: Operates at a frequency higher than 10 MHz
;  10: Operates at a frequency of 1 MHz
;  11: Setting prohibited
;  Be sure to set 0
;++-------- RTCLPC  
;[Setting in subsystem clock HALT mode]
; 0: Enables subsystem clock supply to peripheral functions
; 1: Stops subsystem clock supply to peripheral functions except real-time counter

MOV CKC, #00001000B ; Clock selection
; | +----------------------------- CSS/MCM0/MDIV2 to MDIV0
; | | [Selection of CPU/peripheral hardware clock (fCLK)]
; | | 00x000: fIH
; | | 00x001: fIH/2 (default)
; | | 00x010: fIH/2^2
; | | 00x011: fIH/2^3
; | | 00x100: fIH/2^4
; | | 00x101: fIH/2^5
; | | 01x000: fMX
; | | 01x001: fMX/2
; | | 01x010: fMX/2^2
; | | 01x011: fMX/2^3
; | | 01x100: fMX/2^4
; | | 01x101: fMX/2^5
; | | 1x0xxx: fSUB
; | | 1xlxxx: fSUB/2
; | | ( x : don't care )
; | +---------------- MCS <Read Only>
; | [Status of Main system clock (fMAIN)]
; | 0: Internal high-speed oscillation clock (fIH)
; | 1: High-speed system clock (fMX)
; +---------------- CLS <Read Only>
; [Status of CPU/peripheral hardware clock (fCLK)]
; 0: Main system clock (fMAIN)
; 1: Subsystem clock (fSUB)

RET

;******************************************************************************
; Main loop
;******************************************************************************
MAIN_LOOP:
    MOV    A,   P0     ; Read the switch input status
    AND    A,   #00000011B  ; Mask bits other than those for the switches
    MOV    L,   A     ; Set the switch input status to the lower 8 bits of the
                      ; table address
    MOV    A,   [HL]  ; Read the display data for LED
    MOV    P3,   A     ; Control LED
    BR     $MAIN_LOOP ; Go to the start of the main loop
end
main.c (C language version)

NEC Electronics  78K0R/LH3 Series

78KOR/LH3 Series Sample Program (Initial Settings)

LED Lighting Switch Control

<<History>>
2009.1.-- Release

<<Overview>>
This sample program initializes the microcontroller by specifying settings such as selecting the clock frequency and setting up I/O ports. After the initialization, three LED lights are controlled by two switches in the main loop.

<Primary initial settings>
(Option byte settings)
; - Disabling the watchdog timer
; - Setting the internal high-speed oscillator frequency to 8 MHz
; - Disabling LVI from being started by default
; (Settings during initialization immediately after a reset ends)
; - Setting up I/O ports
; - Securing a supply voltage of 2.7 V or more by using the function of low-voltage detector
; - Specifying that the CPU/peripheral hardware clock run on the internal high-speed oscillation clock (8 MHz)
; - Stopping the X1/XT1 oscillator

<Switch input and LED status>

<table>
<thead>
<tr>
<th>Switch 1</th>
<th>Switch 2</th>
<th>LED1</th>
<th>LED2</th>
<th>LED3</th>
</tr>
</thead>
<tbody>
<tr>
<td>(P00)</td>
<td>(P01)</td>
<td>(P30)</td>
<td>(P31)</td>
<td>(P32)</td>
</tr>
<tr>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
</tr>
<tr>
<td>ON</td>
<td>OFF</td>
<td>ON</td>
<td>OFF</td>
<td>OFF</td>
</tr>
</tbody>
</table>
* 0 is input to the ports if the switches are turned on and 1 is input to the ports if
the switches are turned off.
* The LEDs are turned off if 1 is output from the ports or turned on if 0 is output from
the ports.

<I/O port settings>
Input: P00, P01
Output: P30 to P32
* Set all unused ports that can be specified as output ports as output ports.

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

/*==============================================================================
Preprocessing directive (#pragma)
==============================================================================*/
#pragma SFR     /* SFR names can be described at the C source level */
#pragma DI     /* DI instructions 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 */

/*==============================================================================
Function prototype declaration
==============================================================================*/

void fn_InitPort( void ); /* I/O port setting */
void fn_InitLvi( void );  /* Low voltage detection */
void fn_InitClock( void ); /* Clock frequency setting */

/******************************************************************************/
Initialization after RESET

*******************************************************************************

void hdwinit( void )
{
    /*---------------------------------------------*
    Disable interrupts
    ---------------------------------------------*/
    DI();  /* Disable interrupts */

    /*---------------------------------------------*
    Specify the I/O port
    ---------------------------------------------*/
    fn_InitPort();  /* Set all ports that can be specified as output ports as output ports */

    /*---------------------------------------------*
    Low-voltage detection
    ---------------------------------------------*/
    fn_InitLvi();  /* Securing a supply voltage of 2.7 V or more */

    /*---------------------------------------------*
    Specify the clock frequency
    ---------------------------------------------*/
    fn_InitClock();  /* Operating internal high-speed oscillation clock at 8 MHz */

    /*---------------------------------------------*
    Enable interrupts
    (To use interrupts, enable interrupts here.)
    ---------------------------------------------*/
    /* EI(); */  /* To enable interrupts, */
    /* uncomment this line. */
}

*******************************************************************************

I/O port setting
void fn_InitPort( void )
{
    /**************************************************************************/
    /* Specify the digital I/O */
    /**************************************************************************/
    ADPC = 0b00010000; /* A/D port configuration register */
    /* ||||+++++++ ADPC4 to ADPC0 */
    /* |||| [Analog input (A)/digital I/O (D) switching] */
    /* |||| ++++++++ ANI15/P157 */
    /* |||| |+++++++ ANI10 to ANI8/P152 to P150 */
    /* |||| ||||+++++++ ANI7 to ANI0/P27 to P20 */
    /* |||| 00000: AAAAAAAAAAA */
    /* |||| 00001: AAAAAAAAAAD */
    /* |||| 00010: AAAAAAAAAADD */
    /* |||| 00011: AAAAAAAAAADD */
    /* |||| 00100: AAAAAAADDAD */
    /* |||| 00101: AAAAAAADDD */
    /* |||| 00110: AAAAAADDDDD */
    /* |||| 00111: AAAAADDDDDDD */
    /* |||| 01000: AAAADDDDDDD */
    /* |||| 01001: AAADDDDDDDDD */
    /* |||| 01010: ADDDDDDDDDDD */
    /* |||| 01011: ADDDDDDDDDDD */
    /* |||| 10000: DDDDDDDDDDDD */
    /* ++++++++ Be sure to set 0 */
    /**************************************************************************/
    /* Initialize port 0 */
    /**************************************************************************/
    P0 = 0b00000000; /* Set the P00 to P02 output latches to low level */
    PM0 = 0b1111011; /* Specify P00 and P01 as input ports */
    /* Specify P02 as an output port */
    PU0 = 0b00000011; /* Connect on-chip pull-up resistors to P00 and P01 */
    /* Connect on-chip pull-up resistor to P02 */
    /* P00: Use for switch 1 input */
    /* P01: Use for switch 2 input */
    /* P02: Unused */
/*------------------------------------------*/
Initialize port 1

P1 = 0b00000000; /* Set the P10 to P17 output latches to low level */
PM1 = 0b00000000; /* Specify P10 to P17 as output ports */
    /* P10 to P17: Unused */

/*------------------------------------------*/
Initialize port 2

P2 = 0b00000000; /* Set the P20 to P27 output latches to low level */
PM2 = 0b00000000; /* Specify P20 to P27 as output ports */
    /* P20 to P27: Unused */

/*------------------------------------------*/
Initialize port 3

P3 = 0b00000111; /* Set the P30 to P32 output latches to high level */
    /* Set the P33 and P34 output latches to low level */
PM3 = 0b11100000; /* Specify P30 to P34 as output ports */
    /* P30: Use for turning on LED1 */
    /* P31: Use for turning on LED2 */
    /* P32: Use for turning on LED3 */
    /* P33 and P34: Unused */

/*------------------------------------------*/
Initialize port 4

P4 = 0b00000000; /* Set the P40 and P41 output latches to low level */
PM4 = 0b11111100; /* Specify P40 and P41 as output ports */
    /* P40 and P41: Unused */

/*------------------------------------------*/
Initialize port 5

P5 = 0b00000000; /* Set the P50 to P57 output latches to low level */
PM5 = 0b00000000; /* Specify P50 to P57 as output ports */
    /* P50 to P57: Unused */
APPENDIX A PROGRAM LIST

Initializing ports:

- **Port 6**
  
  ```
  P6 = 0b00000000; /* Set the P60 and P61 output latches to low level */
  PM6 = 0b11111100; /* Specify P60 and P61 as output ports */
  /* P60 and P61: Unused */
  ```

- **Port 7**
  
  ```
  P7 = 0b00000000; /* Set the P70 to P77 output latches to low level */
  PM7 = 0b00000000; /* Specify P70 to P77 as output ports */
  /* P70 to P77: Unused */
  ```

- **Port 8**
  
  ```
  P8 = 0b00000000; /* Set the P80 to P87 output latches to low level */
  PM8 = 0b00000000; /* Specify P80 to P87 as output ports */
  /* P80 to P87: Unused */
  ```

- **Port 9**
  
  ```
  P9 = 0b00000000; /* Set the P90 to P97 output latches to low level */
  PM9 = 0b00000000; /* Specify P90 to P97 as output ports */
  /* P90 to P97: Unused */
  ```

- **Port 10**
  
  ```
  P10 = 0b00000000; /* Set the P100 to P102 output latches to low level */
  PM10 = 0b11111000; /* Specify P100 to P102 as output ports */
  /* P100 to P102: Unused */
  ```

- **Port 11**
  
  ```
  P11 = 0b00000000; /* Set the P110 and P111 output latches to low level */
  ```
APPENDIX A PROGRAM LIST

PM11 = 0b11111100; /* Specify P110 and P111 as output ports */
    /* P110 and P111: Unused */

/*-----------------------------------------------*/
Initialize port 12

-----------------------------------------------*/

P12  = 0b00000000; /* Set the P120 output latch to low level */
PM12 = 0b11111110; /* Specify P120 as output port */
    /* P120 to P124: Unused */
    /* * P121 to P124 are input-only ports */

/*-----------------------------------------------*/
Initialize port 14

-----------------------------------------------*/

P14  = 0b00000000; /* Set the P140 to P147 output latches to low level */
PM14 = 0b00000000; /* Specify P140 to P147 as output ports */
    /* P140 to P147: Unused */

/*-----------------------------------------------*/
Initialize port 15

-----------------------------------------------*/

P15  = 0b00000000; /* Set the P150 to P152, and P157 output latches to low level */
PM15 = 0b01111100; /* Specify P150 to P152, and P157 as output ports */
    /* P150 to P152, and P157: Unused */

}

*******************************************************************************
Low-voltage detection

*******************************************************************************

void fn_InitLvi( void )
{
    unsigned char ucCounter; /* Count variable */

    /* Set up the low-voltage detector */
    LVIMK = 1; /* Disable the INTLVI interrupt */
LVISEL = 0; /* Specify VDD as the detection voltage */
LVIS = 0b00001001; /* Low-voltage detection level select register */
/* |||||++++--- LVIS3 to LVIS0 */
/* ||||| [Detection level] */
/* ||||| 0000: VLVI0 (4.22 ±0.1 V) */
/* ||||| 0001: VLVI1 (4.07 ±0.1 V) */
/* ||||| 0010: VLVI2 (3.92 ±0.1 V) */
/* ||||| 0011: VLVI3 (3.76 ±0.1 V) */
/* ||||| 0100: VLVI4 (3.61 ±0.1 V) */
/* ||||| 0101: VLVI5 (3.45 ±0.1 V) */
/* ||||| 0110: VLVI6 (3.30 ±0.1 V) */
/* ||||| 0111: VLVI7 (3.15 ±0.1 V) */
/* ||||| 1000: VLVI8 (2.99 ±0.1 V) */
/* ||||| 1001: VLVI9 (2.84 ±0.1 V) */
/* ||||| 1010: VLVI10 (2.68 ±0.1 V) */
/* ||||| 1011: VLVI11 (2.53 ±0.1 V) */
/* ||||| 1100: VLVI12 (2.38 ±0.1 V) */
/* ||||| 1101: VLVI13 (2.22 ±0.1 V) */
/* ||||| 1110: VLVI14 (2.07 ±0.1 V) */
/* ||||| 1111: VLVI15 (1.91 ±0.1 V) */
/* ++++------- Be sure to set 0 */
LVIMD = 0; /* Specify that an interrupt signal is generated when a low voltage is detected */
LVION = 1; /* Enable low-voltage detection */

/* Make the system wait until the low-voltage detector stabilizes (10 us or more)*/
for( ucCounter = 0; ucCounter < 4; ucCounter++ ){
    NOP();
}

/* Make the system wait until VLVI is less than or equal to VDD */
while( LVIF ){,
    NOP();
}
LVION = 0; /* Stop the low-voltage detector */

/*******************************************************************************/
Specify the clock frequency

-------------------------------------------------------------
Specify the clock frequency so that the device can run on the internal high-speed oscillation clock.
**********************************************************************************************/

void fn_InitClock( void )
{

CMC  = 0b00000000; /* Clock operation mode */
/* |||||||+--- AMPH */
/* |||||||| [Control of high-speed system clock oscillation frequency] */
/* |||||||| 0: 2 MHz ( fMX < 10 MHz */
/* ||||11|| 1: 10 MHz < fMX ( 20 MHz */
/* ||||++--- AMPHS1, AMPHS0 */
/* ||| [XT1 oscillator oscillation mode selection] */
/* || 00: Low power consumption oscillation (default) */
/* || 01: Normal oscillation */
/* || 10: Ultra-low power consumption oscillation */
/* || 11: Ultra-low power consumption oscillation */
/* |||+---- Be sure to set 0 */
/* ||+------ OSCSELS */
/* || [Subsystem clock pin operation mode] */
/* || 0: Input port mode */
/* || 1: XT1 oscillation mode */
/* |||+----- Be sure to set 0 */
/* ||+------- EXCLK/OSCSEL */
/* *[High-speed system clock pin operation mode] */
/* 00: Input port mode */
/* 01: X1 oscillation mode */
/* 10: Input port mode */
/* 11: External clock input mode */

CSC  = 0b10000000; /* Clock operation status control */
/* |||||+--- HIOSTOP */
/* |||| [Internal high-speed oscillation clock operation control] */
/* ||||| 0: Internal high-speed oscillator operating */
/* ||||| 1: Internal high-speed oscillator stopped */
/* ||+++++ Be sure to set 0 */
/* |+------- XTSTOP */
APPENDIX A PROGRAM LIST

/* | Subsystem clock operation control */
/* | 0: XT1 oscillator operating */
/* | 1: XT1 oscillator stopped */
/* +---------- MSTOP */
/* [High-speed system clock operation control] */
/* 0: X1 oscillator operating */
/* 1: X1 oscillator stopped */

OSMC = 0b10000000; /* Operation speed mode */
/* |+++++++ FSEL/FLPC */
/* | | [fCLK frequency selection] */
/* | | 00: Operates at a frequency of 10 MHz or less (default) */
/* | | 01: Operates at a frequency higher than 10 MHz */
/* | | 10: Operates at a frequency of 1 MHz */
/* | | 11: Setting prohibited */
/* |+++++++ Be sure to set 0 */
/* +---------- RTCLPC */
/* [Setting in subsystem clock HALT mode] */
/* 0: Enables subsystem clock supply to peripheral functions */
/* 1: Stops subsystem clock supply to peripheral functions except */
/* real-time counter */

CKC = 0b00001000; /* Clock selection */
/* |+|+++++++ CSS/MCM0/MDIV2 to MDIV0 */
/* | | [Selection of CPU/peripheral hardware clock (fCLK)] */
/* | | 00x000: fIH */
/* | | 00x001: fIH/2 (default) */
/* | | 00x010: fIH/2^2 */
/* | | 00x011: fIH/2^3 */
/* | | 00x100: fIH/2^4 */
/* | | 00x101: fIH/2^5 */
/* | | 01x000: fMX */
/* | | 01x001: fMX/2 */
/* | | 01x010: fMX/2^2 */
/* | | 01x011: fMX/2^3 */
/* | | 01x100: fMX/2^4 */
/* | | 01x101: fMX/2^5 */
/* | | 1x0xxx: fSUB */
/* | | 1x1xxx: fSUB/2 */
APPENDIX A PROGRAM LIST

/ * | | ( x : don’t care ) */
/ * | +------- MCS <Read Only> */
/ * | [Status of Main system clock (fMAIN)] */
/ * | 0: Internal high-speed oscillation clock (fIH) */
/ * | 1: High-speed system clock (fMX) */
/ * +------- CLS <Read Only> */
/ * [Status of CPU/peripheral hardware clock (fCLK)] */
/ * 0: Main system clock (fMAIN) */
/ * 1: Subsystem clock (fSUB) */

*******************************************************************************

Main loop
*******************************************************************************

void main(void)
{
    const unsigned char aLedOut[4] = {0b00000011,0b00000101,0b00000110,0b00000111}; /* Table for turning on the LEDs */
    unsigned char ucSwitchBuffer; /* Switch input data storage area */

    while(1){
        /* Acquire valid switch information */
        ucSwitchBuffer = ( P0 & 0b00000011 );

        /* Read the data to display from the table and display */
        P3 = ( aLedOut[ucSwitchBuffer] & 0b00000111 );
    }
}
APPENDIX B  REVISION HISTORY

<table>
<thead>
<tr>
<th>Edition</th>
<th>Date Published</th>
<th>Page</th>
<th>Revision</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st edition</td>
<td>September 2009</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>