To our customers,

Old Company Name in Catalogs and Other Documents

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

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

April 1st, 2010
Renesas Electronics Corporation

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

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

Notice

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

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

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

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

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

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

RENESAS

Application Note

78K0S/Kx1+

Sample Program (Serial Interface UART6)

Full-Duplex Communication Using Receive Ring Buffer

This document describes an operation overview of the sample program and how to use it, as well as how to set and use serial interface UART6. In the sample program, the baud rate is set to 9,600 bps, serial communication is performed, and 4-character data is transmitted in accordance with the reception of 1-character data. Similarly, in the case of a reception error, 4-character data is transmitted in accordance with the error.

Target devices
78K0S/KA1+ microcontroller
78K0S/KB1+ microcontroller

CONTENTS

Document No. U18914EJ2V0AN00 (2nd edition)
Date Published September 2008 NS

Windows and Windows XP are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

- **The information in this document is current as of July, 2008. The information is subject to change without notice. For actual design-in, refer to the latest publications of NEC Electronics data sheets or data books, etc., for the most up-to-date specifications of NEC Electronics products. Not all products and/or types are available in every country. Please check with an NEC Electronics sales representative for availability and additional information.**
- No part of this document may be copied or reproduced in any form or by any means without the prior written consent of NEC Electronics. NEC Electronics assumes no responsibility for any errors that may appear in this document.
- NEC Electronics does not assume any liability for infringement of patents, copyrights or other intellectual property rights of third parties by or arising from the use of NEC Electronics products listed in this document or any other liability arising from the use of such products. No license, express, implied or otherwise, is granted under any patents, copyrights or other intellectual property rights of NEC Electronics or others.
- Descriptions of circuits, software and other related information in this document are provided for illustrative purposes in semiconductor product operation and application examples. The incorporation of these circuits, software and information in the design of a customer's equipment shall be done under the full responsibility of the customer. NEC Electronics assumes no responsibility for any losses incurred by customers or third parties arising from the use of these circuits, software and information.
- While NEC Electronics endeavors to enhance the quality, reliability and safety of NEC Electronics products, customers agree and acknowledge that the possibility of defects thereof cannot be eliminated entirely. To minimize risks of damage to property or injury (including death) to persons arising from defects in NEC Electronics products, customers must incorporate sufficient safety measures in their design, such as redundancy, fire-containment and anti-failure features.
- NEC Electronics products are classified into the following three quality grades: "Standard", "Special" and "Specific".
  The "Specific" quality grade applies only to NEC Electronics products developed based on a customer-designated "quality assurance program" for a specific application. The recommended applications of an NEC Electronics product depend on its quality grade, as indicated below. Customers must check the quality grade of each NEC Electronics product before using it in a particular application.
  "Standard": Computers, office equipment, communications equipment, test and measurement equipment, audio and visual equipment, home electronic appliances, machine tools, personal electronic equipment and industrial robots.
  "Special": Transportation equipment (automobiles, trains, ships, etc.), traffic control systems, anti-disaster systems, anti-crime systems, safety equipment and medical equipment (not specifically designed for life support).
  "Specific": Aircraft, aerospace equipment, submersible repeaters, nuclear reactor control systems, life support systems and medical equipment for life support, etc.

The quality grade of NEC Electronics products is "Standard" unless otherwise expressly specified in NEC Electronics data sheets or data books, etc. If customers wish to use NEC Electronics products in applications not intended by NEC Electronics, they must contact an NEC Electronics sales representative in advance to determine NEC Electronics' willingness to support a given application.

(Note)

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

(2) "NEC Electronics products" means any product developed or manufactured by or for NEC Electronics (as defined above).

M8E 02. 11-1

CHAPTER 1 OVERVIEW

This sample program presents an example of using serial interface UART6 that can perform full-duplex communication. The baud rate is set to 9,600 bps, serial communication is performed, and 4-character data is transmitted in accordance with the reception of 1-character data. Similarly, in the case of a reception error, 4-character data is transmitted in accordance with the error.

1.1 Main Contents of the Initial Settings

The main contents of the initial settings are as follows.

- Selecting the crystal or ceramic oscillation clock as the system clock source[Note]
- Stopping watchdog timer operation
- Setting V_{LVI} (low-voltage detection voltage) to 4.3 V ±0.2 V
- Generating an internal reset (LVI reset) signal when it is detected that V_{DD} is less than V_{LVI}, after V_{DD} (power supply voltage) becomes greater than or equal to V_{LVI}
- Setting the CPU clock frequency to 8 MHz
- Setting the I/O ports
- Setting serial interface UART6
  - Baud rate: 9,600 bps
  - Data character length: 7 bits
  - Parity specification: Even parity
  - Number of stop bits: 1 bit
  - Start bit specification: LSB first
  - TxD6 output: Normal output
  - Generating INTSRE6 as the interrupt upon error occurrence
  - Enabling internal operation clock operation
  - Enabling transmit operation
  - Enabling receive operation

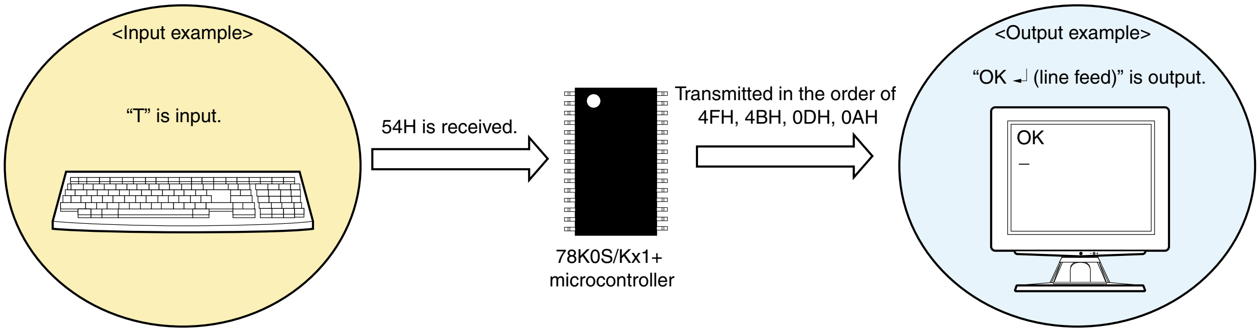
Note This is set by using the option byte.

[Column] What is full-duplex communication?

Full-duplex communication is a type of communication in which a transmit operation and a receive operation can be individually performed at the same time. Serial interface UART6 supports full-duplex communication, enabling transmission and reception to be used at the same time.

1.2 Contents Following the Main Loop

After completion of the initial settings, receive operation of serial communication is started by data input from the RxD6 pin. In this sample program, transmission and reception of ASCII codes are assumed, and 4-character data is transmitted in accordance with the reception of 1-character data. Similarly, in the case of a reception error, 4-character data is transmitted in accordance with the error.



An area of 50 bytes (= 1 byte (1 data) × 50) is secured in the RAM area as a buffer for storing receive data.

Receive data can be received successively, because it is stored into the buffer when an interrupt is serviced, and is sequentially stored from the start of the buffer area. Furthermore, to configure the buffer as a ring buffer, after the receive data has reached the end of the buffer area, the receive data is stored from the start of the buffer area again. The receive data is stored into the buffer when free space is available in the buffer, but it is discarded instead of being stored when no free space is available.

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

[Column] What is an ASCII code?
An ASCII code is a character code that represents a 1-character (alphabetic character, number, symbol, control character) by using seven bits.

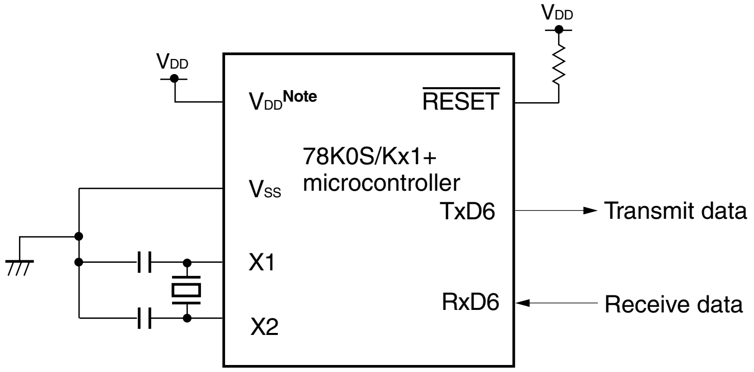
[Column] What is a ring buffer?
A ring buffer is a method to control a transmit or receive buffer. It prepares a fixed buffer area, processes the data in the order it is transferred to the buffer, and controls the buffer so that the address following the end of the buffer area becomes the start address. It is called a ring buffer because the buffer is used as a ring.

CHAPTER 2 CIRCUIT DIAGRAM

This chapter describes a circuit diagram to be used in this sample program.

2.1 Circuit Diagram

A circuit diagram is shown below.



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

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

2. Connect the AV_{SS} pin directly to GND (only for the 78K0S/KB1+ microcontroller).

3. Leave all unused pins open (unconnected), except for the pins shown in the circuit diagram and the AV_{REF} and AV_{SS} pins.

CHAPTER 3 SOFTWARE

This chapter describes the file configuration of the compressed file to be downloaded, internal peripheral functions of the microcontroller to be used, and initial settings and operation overview of the sample program, and shows a flow chart.

3.1 File Configuration

The following table shows the file configuration of the compressed file to be downloaded.

File Name	Description	Compressed (*.zip) File Included	
		ZIP	PM 32
main.asm (Assembly language version) main.c (C language version)	Source file for hardware initialization processing and main processing of microcontroller	●[Note]	●[Note]
op.asm	Assembler source file for setting the option byte (sets the system clock source)	●	●
uart6.prw	Work space file for integrated development environment PM+		●
uart6.prj	Project file for integrated development environment PM+		●

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

Remark ZIP : Only the source file is included.

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

3.2 Internal Peripheral Functions to Be Used

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

- Serial communication: Serial interface UART6
- $V_{DD} < V_{LVI}$ detection: Low-voltage detector (LVI)
- Data input and output: RxD6 and TxD6

3.3 Initial Settings and Operation Overview

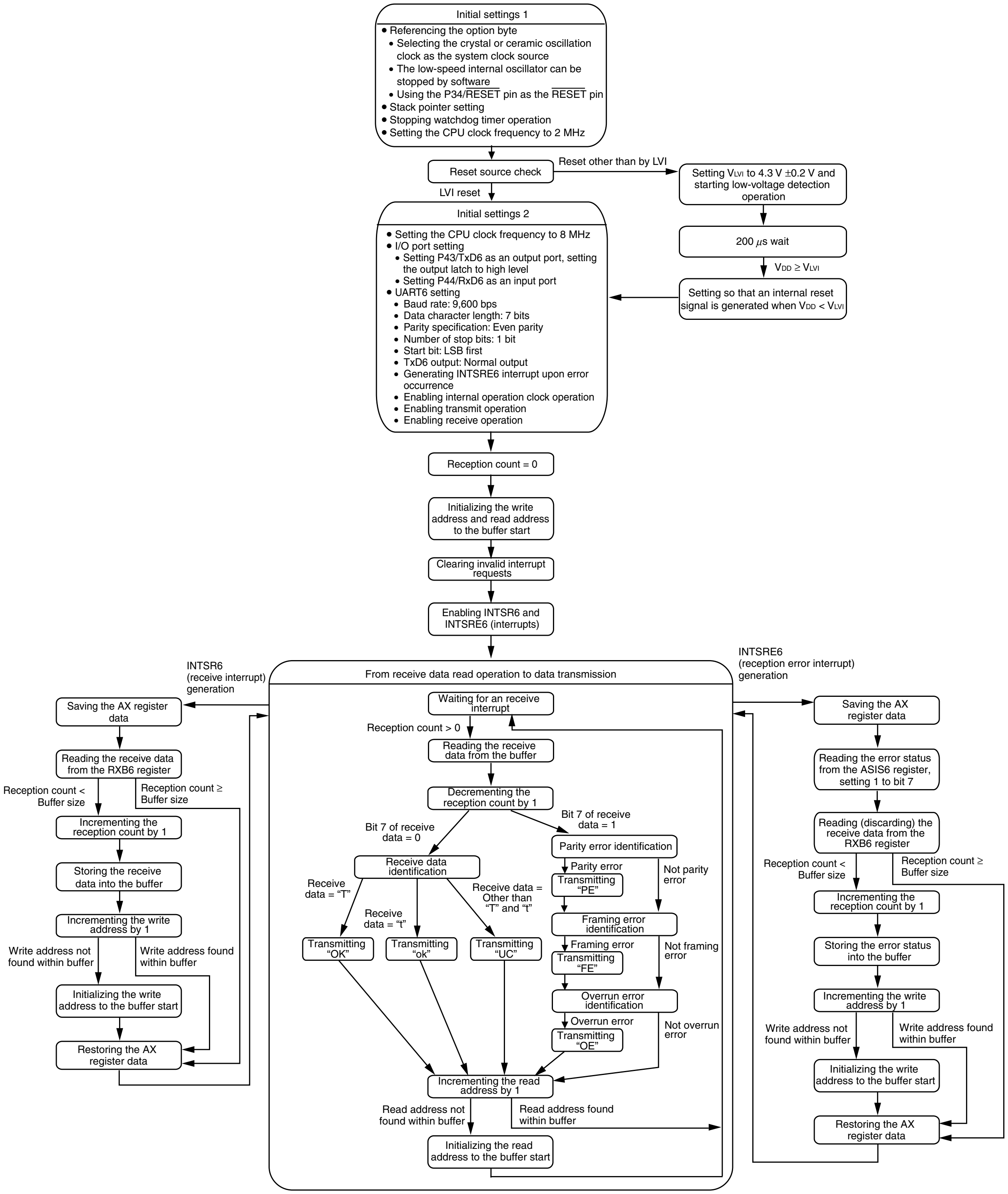
In this sample program, initial settings including the setting of the low-voltage detection function, selection of the clock frequency, setting of the I/O ports, and setting of serial interface UART6 are performed.

After completion of the initial settings, receive operation of serial communication is started by data input from the RxD6 pin. In this sample program, transmission and reception of ASCII codes are assumed, and 4-character data is transmitted in accordance with the reception of 1-character data. Similarly, in the case of a reception error, 4-character data is transmitted in accordance with the error.

An area of 50 bytes (= 1 byte (1 data) × 50) is secured in the RAM area as a buffer for storing receive data.

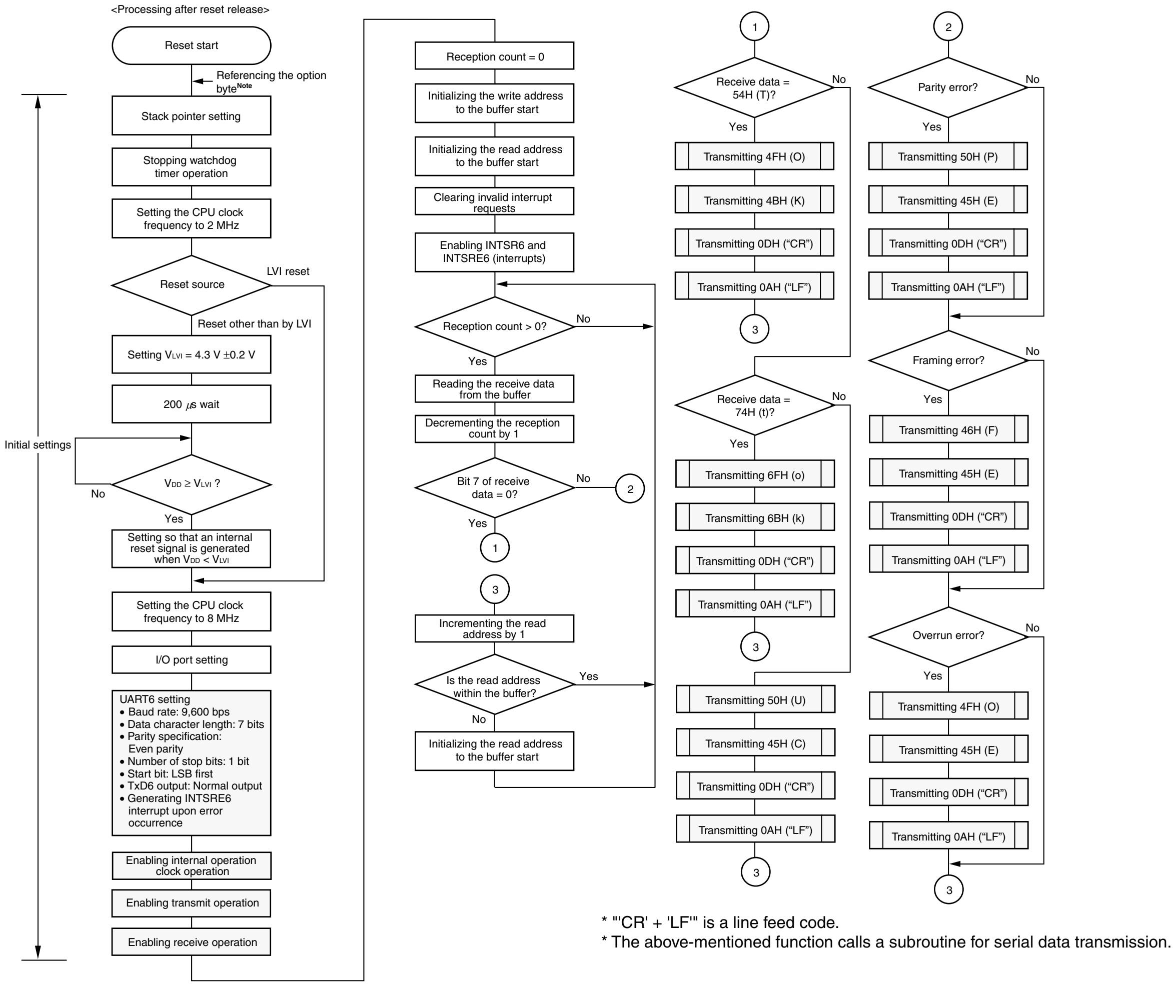
Receive data can be received successively, because it is stored into the buffer when an interrupt is serviced, and is sequentially stored from the start of the buffer area. Furthermore, to configure the buffer as a ring buffer, after the receive data has reached the end of the buffer area, the receive data is stored from the start of the buffer area again. The receive data is stored into the buffer when free space is available in the buffer, but it is discarded instead of being stored when no free space is available.

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


Initial settings 1
• Referencing the option byte
• Selecting the crystal or ceramic oscillation clock as the system clock source
• The low-speed internal oscillator can be stopped by software
• Using the P34/RESET pin as the RESET pin
• Stack pointer setting
• Stopping watchdog timer operation
• Setting the CPU clock frequency to 2 MHz
Reset source check
Reset other than by LVI
Setting VLVI to 4.3 V ±0.2 V and starting low-voltage detection operation
200 μs wait
VDD ≥ VLVI
Setting so that an internal reset signal is generated when VDD < VLVI
LVI reset
Initial settings 2
• Setting the CPU clock frequency to 8 MHz
• I/O port setting
• Setting P43/TxD6 as an output port, setting the output latch to high level
• Setting P44/RxD6 as an input port
• UART6 setting
• Baud rate: 9,600 bps
• Data character length: 7 bits
• Parity specification: Even parity
• Number of stop bits: 1 bit
• Start bit: LSB first
• TxD6 output: Normal output
• Generating INTSRE6 interrupt upon error occurrence
• Enabling internal operation clock operation
• Enabling transmit operation
• Enabling receive operation
Reception count = 0
Initializing the write address and read address to the buffer start
Clearing invalid interrupt requests
Enabling INTSR6 and INTSRE6 (interrupts)
INTSR6 (receive interrupt) generation
From receive data read operation to data transmission
INTSRE6 (reception error interrupt) generation
Saving the AX register data
Reading the receive data from the RXB6 register
Reception count < Buffer size
Reception count ≥ Buffer size
Incrementing the reception count by 1
Storing the receive data into the buffer
Incrementing the write address by 1
Write address not found within buffer
Write address found within buffer
Initializing the write address to the buffer start
Restoring the AX register data
Waiting for an receive interrupt
Reception count > 0
Reading the receive data from the buffer
Decrementing the reception count by 1
Bit 7 of receive data = 0
Bit 7 of receive data = 1
Receive data identification
Receive data = "T"
Receive data = "t"
Receive data = Other than "T" and "t"
Transmitting "OK"
Transmitting "ok"
Transmitting "UC"
Parity error identification
Parity error
Not parity error
Transmitting "PE"
Framing error identification
Framing error
Not framing error
Transmitting "FE"
Overrun error identification
Overrun error
Not overrun error
Transmitting "OE"
Incrementing the read address by 1
Read address not found within buffer
Read address found within buffer
Initializing the read address to the buffer start
Saving the AX register data
Reading the error status from the ASIS6 register, setting 1 to bit 7
Reading (discarding) the receive data from the RXB6 register
Reception count < Buffer size
Reception count ≥ Buffer size
Incrementing the reception count by 1
Storing the error status into the buffer
Incrementing the write address by 1
Write address not found within buffer
Write address found within buffer
Initializing the write address to the buffer start
Restoring the AX register data

3.4 Flow Charts

The flow charts for the sample program are shown below.


<Processing after reset release>
Reset start
Referencing the option byteNote
Stack pointer setting
Stopping watchdog timer operation
Setting the CPU clock frequency to 2 MHz
Reset source
LVI reset
Reset other than by LVI
Setting VLVI = 4.3 V ±0.2 V
200 μs wait
Initial settings
VDD ≥ VLVI ?
No
Yes
Setting so that an internal reset signal is generated when VDD < VLVI
Setting the CPU clock frequency to 8 MHz
I/O port setting
UART6 setting
• Baud rate: 9,600 bps
• Data character length: 7 bits
• Parity specification: Even parity
• Number of stop bits: 1 bit
• Start bit: LSB first
• TxD6 output: Normal output
• Generating INTSRE6 interrupt upon error occurrence
Enabling internal operation clock operation
Enabling transmit operation
Enabling receive operation
Reception count = 0
Initializing the write address to the buffer start
Initializing the read address to the buffer start
Clearing invalid interrupt requests
Enabling INTSR6 and INTSRE6 (interrupts)
Reception count > 0?
No
Yes
Reading the receive data from the buffer
Decrementing the reception count by 1
Bit 7 of receive data = 0?
No
2
Yes
1
3
Incrementing the read address by 1
Is the read address within the buffer?
Yes
No
Initializing the read address to the buffer start
1
Receive data = 54H (T)?
No
Yes
Transmitting 4FH (O)
Transmitting 4BH (K)
Transmitting 0DH (“CR”)
Transmitting 0AH (“LF”)
3
Receive data = 74H (t)?
No
Yes
Transmitting 6FH (o)
Transmitting 6BH (k)
Transmitting 0DH (“CR”)
Transmitting 0AH (“LF”)
3
Transmitting 50H (U)
Transmitting 45H (C)
Transmitting 0DH (“CR”)
Transmitting 0AH (“LF”)
3
2
Parity error?
No
Yes
Transmitting 50H (P)
Transmitting 45H (E)
Transmitting 0DH (“CR”)
Transmitting 0AH (“LF”)
Framing error?
No
Yes
Transmitting 46H (F)
Transmitting 45H (E)
Transmitting 0DH (“CR”)
Transmitting 0AH (“LF”)
Overrun error?
No
Yes
Transmitting 4FH (O)
Transmitting 45H (E)
Transmitting 0DH (“CR”)
Transmitting 0AH (“LF”)
3

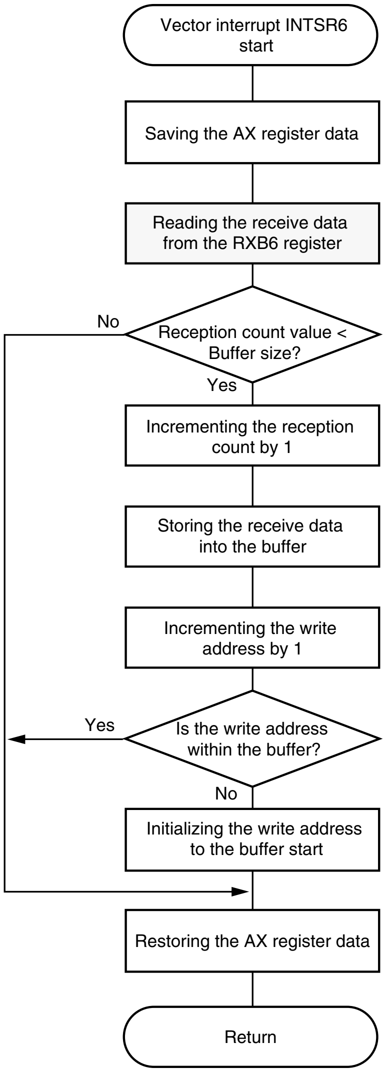
* "'CR' + 'LF'" is a line feed code.
* The above-mentioned function calls a subroutine for serial data transmission.

Note Referencing the option byte is automatically performed by the microcontroller after reset release. In this sample program, the following contents are set by referencing the option byte.

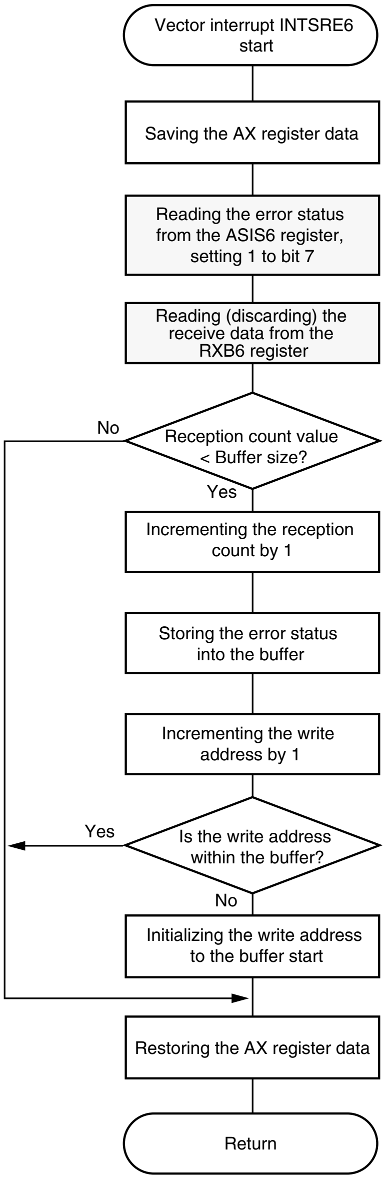
- Using the crystal or ceramic oscillation clock as the system clock source
- The low-speed internal oscillator can be stopped by using software
- Using the P34/$\overline{\text{RESET}}$ pin as the $\overline{\text{RESET}}$ pin

Remark The flow charts of <Vector interrupt INTSR6>, <Vector interrupt INTSRE6>, and <Serial data transmission subroutine> are shown on the next page.

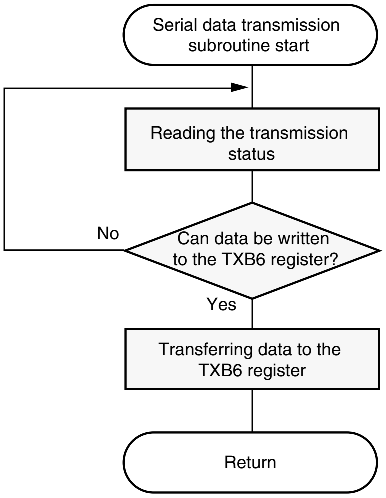
<Vector interrupt INTSR6>


Vector interrupt INTSR6 start
Saving the AX register data
Reading the receive data from the RXB6 register
Reception count value < Buffer size?
No
Yes
Incrementing the reception count by 1
Storing the receive data into the buffer
Incrementing the write address by 1
Is the write address within the buffer?
Yes
No
Initializing the write address to the buffer start
Restoring the AX register data
Return

<Vector interrupt INTSRE6>


Vector interrupt INTSRE6 start
Saving the AX register data
Reading the error status from the ASIS6 register, setting 1 to bit 7
Reading (discarding) the receive data from the RXB6 register
Reception count value < Buffer size?
No
Yes
Incrementing the reception count by 1
Storing the error status into the buffer
Incrementing the write address by 1
Is the write address within the buffer?
Yes
No
Initializing the write address to the buffer start
Restoring the AX register data
Return

<Serial data transmission subroutine>


Serial data transmission subroutine start
Reading the transmission status
No
Can data be written to the TXB6 register?
Yes
Transferring data to the TXB6 register
Return

* Assembly language: Subroutine
C language: Function

CHAPTER 4 SETTING METHODS

This chapter describes the setting of serial interface UART6.

For other initial settings, refer to the **78K0S/Kx1+ Sample Program (Initial Settings) LED Lighting Switch Control Application Note**. For interrupt, refer to the **78K0S/Kx1+ Sample Program (Interrupt) External Interrupt Generated by Switch Input Application Note**. For low-voltage detection (LVI), refer to the **78K0S/Kx1+ Sample Program (Low-Voltage Detection) Reset Generation During Detection at Less than 2.7 V Application Note**.

For how to set registers, refer to the user's manual of each product (**78K0S/KA1+**, **78K0S/KB1+**).

For assembler instructions, refer to the **78K/0S Series Instructions User's Manual**.

4.1 Setting Serial Interface UART6

Serial interface UART6 uses the following eleven types of registers.

- Clock selection register 6 (CKSR6)
- Baud rate generator control register 6 (BRGC6)
- Asynchronous serial interface operation mode register 6 (ASIM6)
- Asynchronous serial interface control register 6 (ASICL6)
- Transmit buffer register 6 (TXB6)
- Receive buffer register 6 (RXB6)
- Asynchronous serial interface transmission status register 6 (ASIF6)
- Asynchronous serial interface reception error status register 6 (ASIS6)
- Input switch control register (ISC)
- Port mode register x (PMx)[Note 1]
- Port register x (Px)[Note 1]

Notes 1. Set the pins to be used with serial interface UART6 as follows.

POWER6	TXE6	RXE6	PM43	P43	PM44	P44	UART6 Operation	Pin Function	
								TxD6/INTP1/P43	RxD6/P44
0	0	0	×[Note 2]	×[Note 2]	×[Note 2]	×[Note 2]	Stop	P43	P44
1	0	1	×[Note 2]	×[Note 2]	1	×	Reception	P43	RxD6
	1	0	0	1	×[Note 2]	×[Note 2]	Transmission	TxD6	P44
	1	1	0	1	1	×	Transmission and reception	TxD6	RxD6

2. This can be used set as a port function.

Remark ×: don't care

POWER6: Bit 7 of the ASIM6 register

TXE6: Bit 6 of the ASIM6 register

RXE6: Bit 5 of the ASIM6 register

<Example of the procedure for setting the basic operation of serial interface UART6>

<1> Using the CKSR6 register to set the base clock (f_{XCLK6}) of UART6
<2> Using the BRGC6 register to set the division value of the base clock (f_{XCLK6}) of UART6
} Baud rate setting
<3> Using the ASIM6 register to set the parity, character length, stop bit, and error interrupt
<4> Using the ASICL6 register to set the start bit and enable or disable TxD6 output reversal
<5> Setting (1) POWER6: Enabling internal operation clock operation
<6> Setting (1) TXE6: Enabling transmit operation
<7> Setting (1) RXE6: Enabling receive operation

Cautions 1. To start transmission, wait for at least one clock of the base clock (f_{XCLK6}) of UART6 to elapse and write the transmit data to the TXB6 register, after step <6>.

2. A receive enable status is entered after one clock of the base clock (f_{XCLK6}) of UART6 has elapsed, after step <7>.

3. Set the PMx register and Px register by taking the relation with the other party of communication into consideration. To use the TxD6 pin, set PM43 to 0 (output) after having set P43 to 1, in order to avoid the generation of unintended start bits (falling signal).

(1) CKSR6 register setting

This register selects the base clock (f_{XCLK6}) of serial interface UART6.

Figure 4-1. Format of Clock Selection Register 6 (CKSR6)

CKSR6

0	0	0	0	TPS63	TPS62	TPS61	TPS60

Base clock (f_{XCLK6}) selection

0	0	0	0	f_{XP}
0	0	0	1	$f_{XP}/2$
0	0	1	0	$f_{XP}/2^2$
0	0	1	1	$f_{XP}/2^3$
0	1	0	0	$f_{XP}/2^4$
0	1	0	1	$f_{XP}/2^5$
0	1	1	0	$f_{XP}/2^6$
0	1	1	1	$f_{XP}/2^7$
1	0	0	0	$f_{XP}/2^8$
1	0	0	1	$f_{XP}/2^9$
1	0	1	0	$f_{XP}/2^{10}$
1	0	1	1	$f_{XP}/2^{11}$
Other than the above				Setting prohibited

Caution Rewrite TPS63 to TPS60 after having set POWER6 to 0.

Remarks 1. The CKSR6 register can be refreshed (writing the same value) by using software during a communication operation (POWER6 = 1 and TXE6 = 1, or POWER6 = 1 and RXE6 = 1).

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

(2) BRGC6 register setting

This register selects the division value of the base clock (f_{XCLK6}) of serial interface UART6.

Figure 4-2. Format of Baud Rate Generator Control Register 6 (BRGC6)

BRGC6

MLD67	MLD66	MLD65	MLD64	MLD63	MLD62	MLD61	MLD60	k	8-bit counter output clock selection
0	0	0	0	0	x	x	x	x	Setting prohibited
0	0	0	0	1	0	0	0	8	$f_{XCLK6}/8$
0	0	0	0	1	0	0	1	9	$f_{XCLK6}/9$
0	0	0	0	1	0	1	0	10	$f_{XCLK6}/10$
• • •	• • •	• • •	• • •	• • •	• • •	• • •	• • •	• • •	• • •
1	1	1	1	1	1	0	0	252	$f_{XCLK6}/252$
1	1	1	1	1	1	0	1	253	$f_{XCLK6}/253$
1	1	1	1	1	1	1	0	254	$f_{XCLK6}/254$
1	1	1	1	1	1	1	1	255	$f_{XCLK6}/255$

Cautions 1. Rewrite MLD67 to MLD60 after having set both TXE6 and RXE6 to 0.

2. The baud rate value is the 8-bit counter output clock divided by 2.

- Baud rate = $\frac{f_{XCLK6}}{2 \times k}$ [bps]

Remarks 1. The BRGC6 register can be refreshed (writing the same value) by using software during a communication operation (POWER6 = 1 and TXE6 = 1, or POWER6 = 1 and RXE6 = 1).

2. f_{XCLK6}: Frequency of the base clock selected by using TPS63 to TPS60

3. k: Value set by using MLD67 to MLD60 (k = 8, 9, 10, ..., 255)

4. ×: don't care

(3) ASIM6 register setting

This register controls the serial communication operation of serial interface UART6.

Figure 4-3. Format of Asynchronous Serial Interface Operation Mode Register 6 (ASIM6)

ASIM6

POWER6	TXE6[Note 1]	RXE6[Note 2]	PS61	PS60	CL6	SL6	ISRM6

Interrupt specification upon error occurrence

ISRM6	
0	INTSRE6
1	INTSR6 (same interrupt as that generated upon receive completion)

Specification of number of stop bits of transmit data

SL6	
0	Number of stop bits = 1
1	Number of stop bits = 2

Specification of character lengths of transmit and receive data

CL6	
0	Data character length = 7 bits
1	Data character length = 8 bits

Parity bit specification

PS61	PS60	Transmit operation	Receive operation
0	0	Does not output parity bit.	Reception without parity
0	1	0 parity output	Reception as 0 parity[Note 3]
1	0	Odd-parity output	Judges as odd parity.
1	1	Even-parity output	Judges as even parity.

Enabling or disabling receive operation

RXE6	
0	Disables receive operation (synchronously resets the receive circuit).
1	Enables receive operation.

Enabling or disabling transmit operation

TXE6	
0	Disables transmit operation (synchronously resets the transmit circuit).
1	Enables transmit operation.

Enabling or disabling internal operation clock operation

POWER6	
0[Note 4]	Disables internal operation clock operation (fixes to low level) and asynchronously resets the internal circuit[Note 5].
1[Note 6]	Enables internal operation clock operation.

Remark The ASIM6 register can be refreshed (writing the same value) by using software during a communication operation (POWER6 = 1 and TXE6 = 1, or POWER6 = 1 and RXE6 = 1).

(Notes and Cautions are given on the next page.)

Notes 1. TXE6 is synchronized by the base clock (f_{XCLK6}) set by the CKSR6 register. To re-enable transmit operation, set TXE6 to 1 after having set TXE6 to 0 and one clock of the base clock (f_{XCLK6}) has elapsed. If TXE6 is set to 1 before one clock of the base clock (f_{XCLK6}) has elapsed, the transmit circuit may not able to be initialized.

2. RXE6 is synchronized by the base clock (f_{XCLK6}) set by the CKSR6 register. To re-enable receive operation, set RXE6 to 1 after having set RXE6 to 0 and one clock of the base clock (f_{XCLK6}) has elapsed. If RXE6 is set to 1 before one clock of the base clock (f_{XCLK6}) has elapsed, the receive circuit may not be able to be initialized.

3. If "Reception as 0 parity" is selected, the parity is not judged. Therefore, the PE6 flag of the ASIS6 register is not set and no error interrupt occurs.

4. The output of the TxD6 pin goes to high level and the input from the RxD6 pin is fixed to high level when POWER6 is set to 0 during a transmission.

5. The ASIS6 register, the ASIF6 register, SBRF6 and SBRT6 of the ASICL6 register, and the RXB6 register are reset.

6. A base clock (f_{XCLK6}) is supplied as the internal operation clock when POWER6 is set to 1 and one clock of the base clock (f_{XCLK6}) has elapsed.

Cautions 1. At startup, transmit operation is started by setting TXE6 to 1 after having set POWER6 to 1, then setting the transmit data to the TXB6 register after having waited for at least one clock of the base clock (f_{XCLK6}) to elapse. To stop transmit operation, set POWER6 to 0 after having set TXE6 to 0.

2. At startup, a reception enable status is entered after having set POWER6 to 1, then setting RXE6 to 1, and one clock of the base clock (f_{XCLK6}) has elapsed. To stop receive operation, set POWER6 to 0 after having set RXE6 to 0.

3. Set POWER6 = 1 → RXE6 = 1 in a state where a high level has been input to the RxD6 pin. If POWER6 = 1 → RXE6 = 1 is set during low-level input, reception is started and correct data will not be received.

4. Clear TXE6 and RXE6 to 0 before rewriting PS61, PS60, and CL6.

5. Fix PS61 and PS60 to 0 when the interface is used in LIN communication operation.

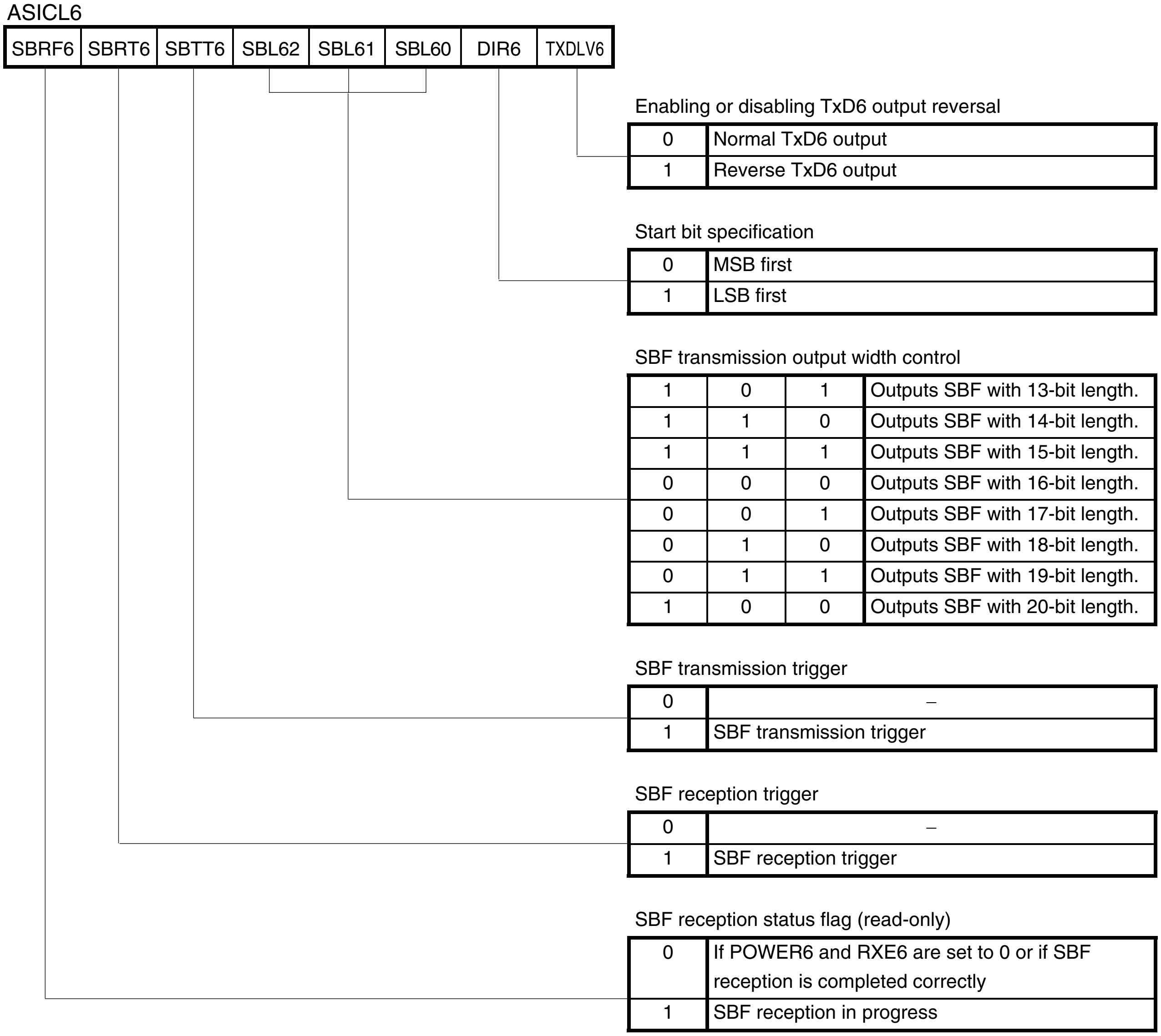
6. Rewrite SL6 after having set TXE6 to 0. Reception is not affected by the SL6 setting value, because it is always performed with "Number of stop bits = 1".

7. Rewrite ISRM6 after having set RXE6 to 0.

(4) ASICL6 register setting

This register controls the serial communication operation of serial interface UART6.

Figure 4-4. Format of Asynchronous Serial Interface Control Register 6 (ASICL6)



ASICL6

SBRF6	SBRT6	SBTT6	SBL62	SBL61	SBL60	DIR6	TXDLV6

Enabling or disabling TxD6 output reversal

0	Normal TxD6 output
1	Reverse TxD6 output

Start bit specification

0	MSB first
1	LSB first

SBF transmission output width control

1	0	1	Outputs SBF with 13-bit length.
1	1	0	Outputs SBF with 14-bit length.
1	1	1	Outputs SBF with 15-bit length.
0	0	0	Outputs SBF with 16-bit length.
0	0	1	Outputs SBF with 17-bit length.
0	1	0	Outputs SBF with 18-bit length.
0	1	1	Outputs SBF with 19-bit length.
1	0	0	Outputs SBF with 20-bit length.

SBF transmission trigger

0	–
1	SBF transmission trigger

SBF reception trigger

0	–
1	SBF reception trigger

SBF reception status flag (read-only)

0	If POWER6 and RXE6 are set to 0 or if SBF reception is completed correctly
1	SBF reception in progress

Remark The ASICL6 register can be refreshed (writing the same value) by using software during a communication operation (POWER6 = 1 and TXE6 = 1, or POWER6 = 1 and RXE6 = 1), if 0 data has been written to ASICL6 by SBRT6 and SBTT6.

(Cautions are given on the next page.)

Cautions 1. SBF transmission and SBF reception are used to use the interface in LIN communication operation. For details of SBF transmission and SBF reception, refer to the user's manual of each product (78K0S/KA1+, 78K0S/KB1+).

2. In the case of an SBF reception error, the operation is returned to the SBF reception mode again. The status of the SBRF6 flag will be held (1).

3. Set SBRT6 to 1 after having set POWER6 and RXE6 to 1. Furthermore, after having set SBRT6 to 1, do not clear SBRT6 to 0 before SBF reception ends (an interrupt request signal is generated).

4. The read value of SBRT6 is always 0. SBRT6 is automatically cleared to 0 after SBF reception has been correctly completed.

5. Set SBTT6 to 1 after having set POWER6 and TXE6 to 1. Furthermore, after having set SBTT6 to 1, do not clear SBTT6 to 0 before SBF transmission ends (an interrupt request signal is generated).

6. The read value of SBTT6 is always 0. SBTT6 is automatically cleared to 0 after SBF transmission has been completed.

7. Rewrite DIR6 and TXDLV6 after having cleared TXE6 and RXE6 to 0.

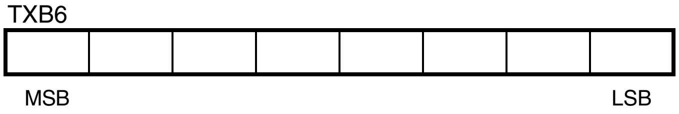
(5) TXB6 register operation

This buffer register is used to set the transmit data of serial interface UART6. Transmit operation is started by writing transmit data to the TXB6 register.

If the data length is set to 7 bits:

- In LSB-first transmission, bits 0 to 6 of TXB6 are transferred, and the MSB of TXB6 is not transmitted.
- In MSB-first transmission, bits 7 to 1 of TXB6 are transferred, and the LSB of TXB6 is not transmitted.

Figure 4-5. Format of Transmit Buffer Register 6 (TXB6)



Cautions 1. To start transmission, write transmit data to TXB6, after having set TXE6 to 1 and having waited for at least one clock of the base clock (f_{XCLK6}) to elapse.

2. Do not write data to the TXB6 register when TXBF6 of the ASIF6 register is 1.

3. Do not refresh (writing the same value) the TXB6 register by using software during a communication operation (POWER6 = 1 and TXE6 = 1, or POWER6 = 1 and RXE6 = 1). To output same values in successive transmission, be sure to confirm that TXBF6 is 0 before writing the same values to the TXB6 register.

[Column] What is LIN?

LIN stands for Local Interconnect Network and is a low-speed (1 to 20 kbps) serial communication protocol intended to aid the cost reduction of an automotive network. Serial interface UART6 is supported with a LIN-bus.

LIN communication is a single-master communication, and up to 15 slaves can be connected to one master.

The LIN slaves are used to control switches, actuators, and sensors, and these are connected to the LIN master via the LIN network.

Normally, the LIN master is connected to a network such as CAN (Controller Area Network).

In addition, the LIN bus uses a single-wire method and is connected to each node via a transceiver that complies with ISO9141.

In the LIN protocol, the master transmits a frame appended with baud rate information. A slave receives the frame and corrects the error in the baud rate with respect to the master. The baud rate can be corrected if the error among the master and slave is ±15% or less.

(6) RXB6 register operation

This buffer register is used to store the receive data of serial interface UART6.

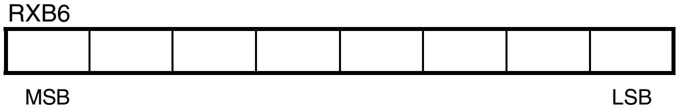
Each time 1 byte of data is received, new receive data is transferred.

If the data length is set to 7 bits:

- In LSB-first reception, the receive data is transferred to bits 0 to 6 of RXB6 and the MSB of RXB6 is always 0.
- In MSB-first reception, the receive data is transferred to bits 7 to 1 of RXB6 and the LSB of RXB6 is always 0.

If an overrun error (OVE6) occurs, the receive data is not transferred to RXB6.

Figure 4-6. Format of Receive Buffer Register 6 (RXB6)



Cautions 1. A reception enable status is entered, after having set RXE6 to 1 and one clock of the base clock (f_{XCLK6}) has elapsed.

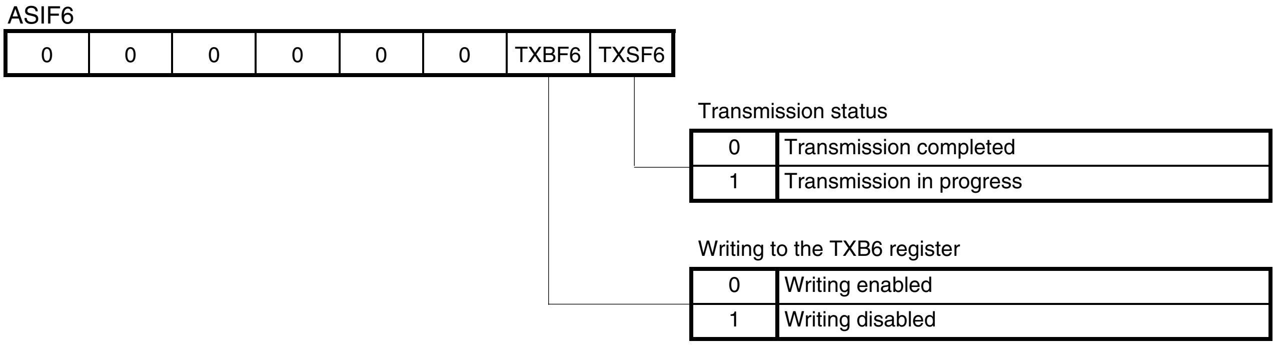
2. Be sure to read the RXB6 register, even if a reception error occurs. Otherwise, an overrun error will occur when the next data is received, and the reception error status will persist.

(7) ASIF6 register operation

This read-only register indicates the status of transmission by serial interface UART6.

Transmission can be continued without disruption even during an interrupt period, by confirming the data transfer of the TXB6 register by using the ASIF6 register, and writing the next data to the TXB6 register.

Figure 4-7. Format of Asynchronous Serial Interface Transmission Status Register 6 (ASIF6)



Cautions 1. To transmit data successively, write the first transmit data (first byte) to the TXB6 register. Be sure to write the next transmit data (second byte) to the TXB6 register after having confirmed that TXBF6 is "0". If data is written to the TXB6 register while TXBF6 is "1", the transmit data cannot be guaranteed.

2. To initialize the transmission unit upon completion of successive transmission, be sure to confirm that TXSF6 is "0" after a transmission completion interrupt has been generated. If initialization is executed while TXSF6 is "1", the transmit data cannot be guaranteed.

(8) ASIS6 register operation

This read-only register indicates an error status on completion of reception by serial interface UART6.

Figure 4-8. Format of Asynchronous Serial Interface Reception Error Status Register 6 (ASIS6)

ASIS6

0	0	0	0	0	PE6	FE6	OVE6

Status flag indicating overrun error

0	If POWER6 = 0 and RXE6 = 0, or if ASIS6 register is read
1	If receive data is set to the RXB6 register and the next receive operation is completed before the data is read (overrun error)

Status flag indicating framing error

0	If POWER6 = 0 and RXE6 = 0, or if ASIS6 register is read
1	If the stop bit is not detected on completion of reception (framing error)

Status flag indicating parity error

0	If POWER6 = 0 and RXE6 = 0, or if ASIS6 register is read
1	If the parity of transmit data does not match the parity bit on completion of reception (parity error)

Cautions 1. The operation of PE6 varies, depending on the setting values of PS61 and PS60 of the ASIM6 register.

2. Only the first stop bit of the receive data is checked, regardless of the number of stop bits.

3. If an overrun error occurs, the next receive data is discarded instead of being written to the RXB6 register.

4. Be sure to read the ASIS6 register before reading the RXB6 register.

(9) ISC register setting

This register is to be set when receiving a status signal transmitted from the master during LIN reception. Input signals from the RxD6 pin can be input to an external interrupt (INTP0) and to 16-bit timer/event counter 00 by setting the ISC register.

Figure 4-9. Format of Input Switch Control Register (ISC)

ISC

0	0	0	0	0	0	ISC1	ISC0

INTP0 input source selection

0	INTP0 (P30)
1	RxD6 (P44)

TI000 input source selection

0	TI000 (P30)
1	RxD6 (P44)

Caution For details of LIN transmission and LIN reception, refer to the user's manual of each product (78K0S/KA1+, 78K0S/KB1+).

[Example] Starting serial transmit and receive operation by setting serial interface UART6 as follows

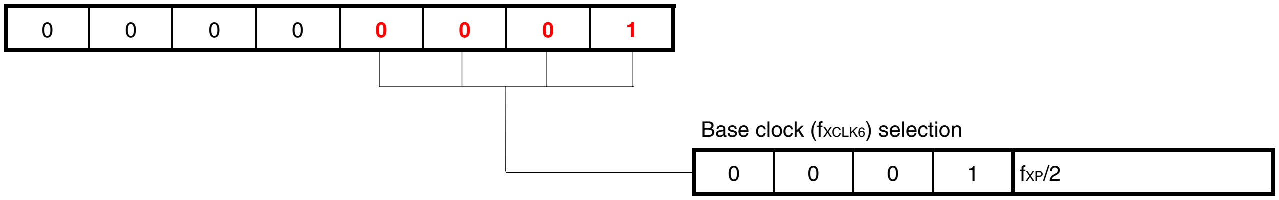
• Baud rate:	9,600 bps
• Data character length:	7 bits
• Parity bit specification:	Even parity
• Number of stop bits:	1 bit
• Start bit:	LSB first
• TxD6 output:	Normal output
• Interrupt generated upon error occurrence:	INTSRE6

(Oscillation frequency of the clock supplied to peripheral hardware (f_{XP}) = 8 MHz, not performing LIN communication)

(Same contents as in this sample program source)

(1) Register settings

<1> CKSR6



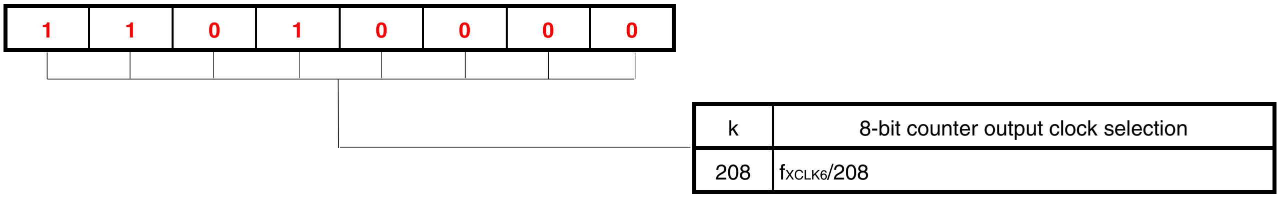
0	0	0	0	0	0	0	1

Base clock (f_{XCLK6}) selection

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

- Base clock (f_{XCLK6}) = $f_{XP}/2$ = 8 MHz/2 = 4 MHz

<2> BRGC



1	1	0	1	0	0	0	0

k	8-bit counter output clock selection
208	$f_{XCLK6}/208$

- Baud rate = $\dfrac{f_{XCLK6}}{2 \times k}$ [bps] = $\dfrac{4\ \text{MHz}}{2 \times 208}$ [bps] = $\dfrac{4{,}000{,}000}{2 \times 208}$ [bps] $\cong$ 9,600 [bps]

<3> ASIM6

POWER6	TXE6	RXE6	1	1	0	0	0

Interrupt specification upon error occurrence

0	INTSRE6

Specification of number of stop bits of transmit data

0	Number of stop bits = 1

Specification of character lengths of transmit and receive data

0	Data character length = 7 bits

Parity bit specification

		Transmit operation	Receive operation
1	1	Even-parity output	Judges as even parity.

Enabling or disabling receive operation

0	Disables receive operation (synchronously resets the receive circuit).
1	Enables receive operation.

Enabling or disabling transmit operation

0	Disables transmit operation (synchronously resets the transmit circuit).
1	Enables transmit operation.

Enabling or disabling internal operation clock operation

0	Disables internal operation clock operation (fixes to low level) and asynchronously resets the internal circuit.
1	Enables internal operation clock operation.

<4> ASICL6: Used as set by default (start bit: LSB first, TxD6 output: normal output)

<5> PMx, Px

PM43	P43	PM44	P44	UART6 Operation	Pin Function	
					TxD6/INTP1/P43	RxD6/P44
0	1	1	×	Transmission and reception	TxD6	RxD6

Remark ×: don’t care

(2) Sample program

<1> Assembly language

```
SET1   P4.3
CLR1   PM4.3
SET1   PM4.4
MOV    CKSR6  #1
MOV    BRGC6  #208
MOV    ASIM6, #00011000B
SET1   POWER6
SET1   TXE6
SET1   RXE6
```

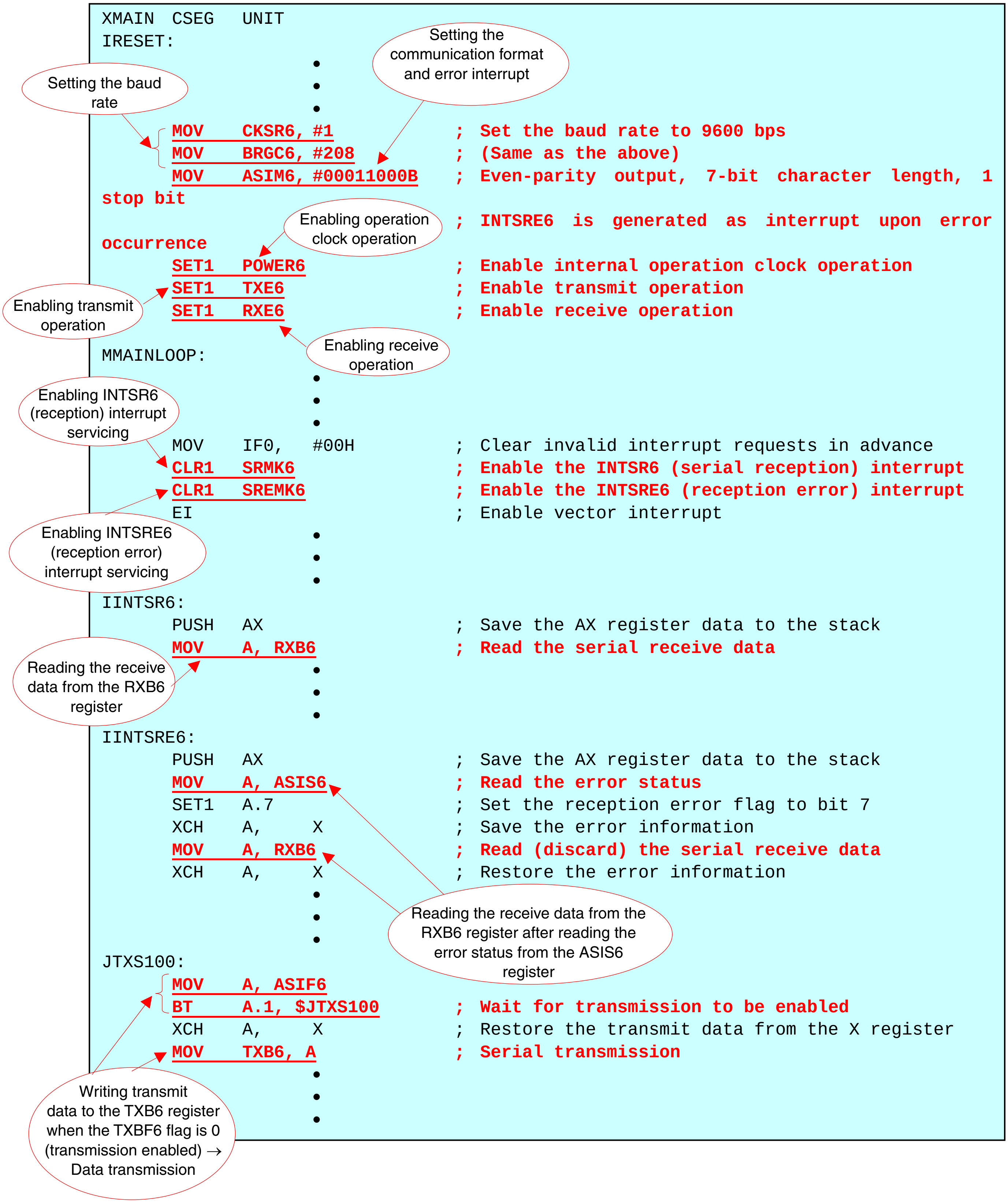
<2> C language

```
P4.3 = 1;
PM4.3 = 0;
PM4.4 = 1;
CKSR6 = 1;
BRGC6 = 208;
ASIM6 = 0b00011000;
POWER6 = 1;
TXE6 = 1;
RXE6 = 1;
```

[Excerpt from this sample program source]

An excerpt from APPENDIX A PROGRAM LIST, which is related to the serial interface UART6 function, is shown below (same contents as in [Example] mentioned above).

(1) Assembly language



```
XMAIN  CSEG   UNIT
IRESET:
                     •
                     •
                     •
       MOV    CKSR6, #1              ; Set the baud rate to 9600 bps
       MOV    BRGC6, #208            ; (Same as the above)
       MOV    ASIM6, #00011000B      ; Even-parity output, 7-bit character length, 1
stop bit
                                     ; INTSRE6 is generated as interrupt upon error
occurrence
       SET1   POWER6                 ; Enable internal operation clock operation
       SET1   TXE6                   ; Enable transmit operation
       SET1   RXE6                   ; Enable receive operation

MMAINLOOP:
                     •
                     •
                     •
       MOV    IF0,   #00H            ; Clear invalid interrupt requests in advance
       CLR1   SRMK6                  ; Enable the INTSR6 (serial reception) interrupt
       CLR1   SREMK6                 ; Enable the INTSRE6 (reception error) interrupt
       EI                            ; Enable vector interrupt
                     •
                     •
                     •
IINTSR6:
       PUSH   AX                     ; Save the AX register data to the stack
       MOV    A, RXB6                ; Read the serial receive data
                     •
                     •
                     •
IINTSRE6:
       PUSH   AX                     ; Save the AX register data to the stack
       MOV    A, ASIS6               ; Read the error status
       SET1   A.7                    ; Set the reception error flag to bit 7
       XCH    A,     X               ; Save the error information
       MOV    A, RXB6                ; Read (discard) the serial receive data
       XCH    A,     X               ; Restore the error information
                     •
                     •
                     •
JTXS100:
       MOV    A, ASIF6
       BT     A.1, $JTXS100          ; Wait for transmission to be enabled
       XCH    A,     X               ; Restore the transmit data from the X register
       MOV    TXB6, A                ; Serial transmission
                     •
                     •
                     •
```

Setting the communication format and error interrupt
Setting the baud rate
Enabling operation clock operation
Enabling transmit operation
Enabling receive operation
Enabling INTSR6 (reception) interrupt servicing
Enabling INTSRE6 (reception error) interrupt servicing
Reading the receive data from the RXB6 register
Reading the receive data from the RXB6 register after reading the error status from the ASIS6 register
Writing transmit data to the TXB6 register when the TXBF6 flag is 0 (transmission enabled) → Data transmission

(2) C language

```
void hdwinit(void){
      unsigned char ucCnt200us; /* 8-bit variable for 200 us wait */
                    •
                    •
                    •
      CKSR6 = 1;                /* Set the baud rate to 9600 bps */
      BRGC6 = 208;              /* (Same as the above) */
      ASIM6 = 0b00011000;       /* Even-parity output, 7-bit character length, 1 stop bit
*/
                                /* INTSRE6 is generated as interrupt upon error occurrence
*/
      POWER6 = 1;               /* Enable internal operation clock operation */
      TXE6   =  1;              /* Enable transmit operation */
      RXE6   =  1;              /* Enable receive operation */
      return;
}

void main(void)
{
                    •
                    •
                    •
      IF0 = 0x00;               /* Clear invalid interrupt requests in advance */
      SRMK6 = 0;                /* Enable the INTSR6 (serial reception) interrupt */
      SREMK6 = 0;               /* Enable the INTSRE6 (reception error) interrupt */
      EI();                     /* Enable vector interrupt */
                    •
                    •
                    •
}

__interrupt void fn_intsr6()
{
   unsigned char ucData;
   ucData = RXB6;               /* Read the serial receive data */
                    •
                    •
                    •
}

__interrupt void fn_intsre6()
{
   unsigned char ucData;
   unsigned char ucTemp;
   ucData = ASIS6 | 0b10000000; /* Store the error information by setting the error
flag to bit 7 */
   ucTemp = RXB6;                 /* Read (discard) the serial receive data */
                    •
                    •
                    •
}

void fn_uart_send(unsigned char ucTxData)
{
   g_ucAsif6 = ASIF6;              /* Read the transmission status */
         while (g_ucAsif6.1)       /* Wait for transmission to be enabled */
   {
         g_ucAsif6 = ASIF6;        /* Read the transmission status */
   }
   TXB6 = ucTxData;                /* Serial transmission */
   return;
}
```

Setting the baud rate

Setting the communication format and error interrupt

Enabling operation clock operation

Enabling transmit operation

Enabling receive operation

Enabling INTSR6 (reception) interrupt servicing

Enabling INTSRE6 (reception error) interrupt servicing

Reading the receive data from the RXB6 register

Reading the receive data from the RXB6 register after reading the error status from the ASIS6 register

Writing transmit data to the TXB6 register when the TXBF6 flag is 0 (transmission enabled) → Data transmission

4.2 Receive Data or Reception Error Content and Transmit Data

In this sample program, serial communication is performed by using serial interface UART6 and data is transmitted in accordance with the data received or the reception error content. The receive data assumes a 1-character ASCII code and the transmit data assumes a 4-character ASCII code.

When reception has been completed normally, the bits of the receive data are sequentially transferred to bit 0 of the receive buffer register (RXB6), from the start bit and up to bit 6, because the data character length was set to 7 bits and the start bit to LSB first. At this time, bit 7 (MSB) is always 0. The receive data of the RXB6 register is stored into the buffer during interrupt (INTSR6) servicing.

When a reception error occurs, the error status of the ASIS6 register is stored into bits 0 to 6 of the buffer and bit 7 is set to 1 during interrupt (INTSRE6) servicing.

Consequently, the contents of the data of bits 0 to 6 are identified after the data of bits 0 to 6 are identified by bit 7 as being receive data or error statuses when the buffer is read.

The transmit data corresponding to the receive data read or the reception error content are as follows.

● Normal reception (bit 7 is 0)

1-Character Receive Data (Hexadecimal Data)	4-Character Transmit Data (Hexadecimal Data)			
T (54H)	O (4FH)	K (4BH)	“CR” (0DH)	“LF” (0AH)
t (74H)	o (6FH)	k (6BH)	“CR” (0DH)	“LF” (0AH)
Other than the above	U (55H)	C (43H)	“CR” (0DH)	“LF” (0AH)

● Error reception (bit 7 is 1)

Reception Error Content (ASIS6 Register Flag Value)	4-Character Transmit Data (Hexadecimal Data)			
Parity error (PE6 is 1)	P (50H)	E (45H)	“CR” (0DH)	“LF” (0AH)
Framing error (FE6 is 1)	F (46H)	E (45H)	“CR” (0DH)	“LF” (0AH)
Overrun error (OVE6 is 1)	O (4FH)	E (45H)	“CR” (0DH)	“LF” (0AH)

Remark “CR” and “LF” are control characters. Combining “CR” and “LF” forms a line feed code.

[Column] Macro and subroutine

In this assembly language sample program, a macro and a subroutine are used to simplify descriptions. The maintenance of the program can be facilitated by using a macro and a subroutine, because only one revision has to be made in the processing program and the program can be easily understood.

<1> Macro

Using a macro facilitates simplifying descriptions by defining names to processing programs that are frequently used and using these names. The program execution time can be shortened, because the program codes that have been defined by the macro are expanded into the macro references without being interrupted by redundant instructions, such as CALL, CALLT, and RET instructions. Furthermore, similar processing programs can be easily defined by simply changing the parameters, because the parameters can be defined.

<2> Subroutine

Using a subroutine helps to simplify descriptions by cutting out a unified processing program so that it can be referenced from the main routine.

<Excerpt from this sample program source>

```
_SEROUT     MACRO  RTXDATA
   PUSH     AX                  ; Save the AX register data to the stack
   MOV      A,     RTXDATA      ; Store provisional parameter RTXDATA to
                                ; the A register
   CALLT    [ZTXSUB]            ; Execute the UART transmission subroutine
   POP      AX                  ; Restore the AX register data
   ENDM
            •
            •
            •
XCALT       CSEG   CALLT0
ZTXSUB:     DW     STXSUB       ; UART transmission subroutine
            •
            •
            •
MMAINLOOP:
            •
            •
            •
LMLP200:
   CMP      A, #'T'             ; Compare the receive data with T (54H)
   BNZ      $LMLP210            ; Branch if the receive data is not T (54H)
   _SEROUT  #'O'                ; Transmit O (4FH)
   _SEROUT  #'K'                ; Transmit K (4BH)
   _SEROUT  #0DH                ; Transmit line feed code “CR”
   _SEROUT  #0AH                ; Transmit line feed code “LF”
   BR       !LMLP400            ; Branch to LMLP400
            •
            •
            •
STXSUB:
     XCH    A, X                ; Save the transmit data to the X register
JTXS100:
     MOV    A, ASIF6
     BT     A.1, $JTXS100       ; Wait for transmission to be enabled
     XCH    A, X                ; Restore the transmit data from the X
                                ; register
     MOV    TXB6, A             ; Serial transmission
     RET                        ; Return from the subroutine
```

Macro definition of macro name “_SEROUT” and provisional parameter “RTXDATA”

Macro execution contents

Referencing the subroutine

Ending macro definition

Registering address “ZTXSUB” to the CALLT instruction table area so that subroutine “STXSUB” can be called by using the CALLT instruction

Actual parameters

Referencing the macro (The above-mentioned four lines are expanded.)

Subroutine

Subroutine processing contents

Returning to the main routine

CHAPTER 5 OPERATION CHECK USING THE DEVICE

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

5.1 Building the Sample Program

This section describes how to build sample programs, using the sample program (source files + project file) downloaded by clicking the icon. For how to build other downloaded programs, refer to the **78K0S/Kx1+ Sample Program Startup Guide Application Note**.

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

[Column] Build errors

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

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

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

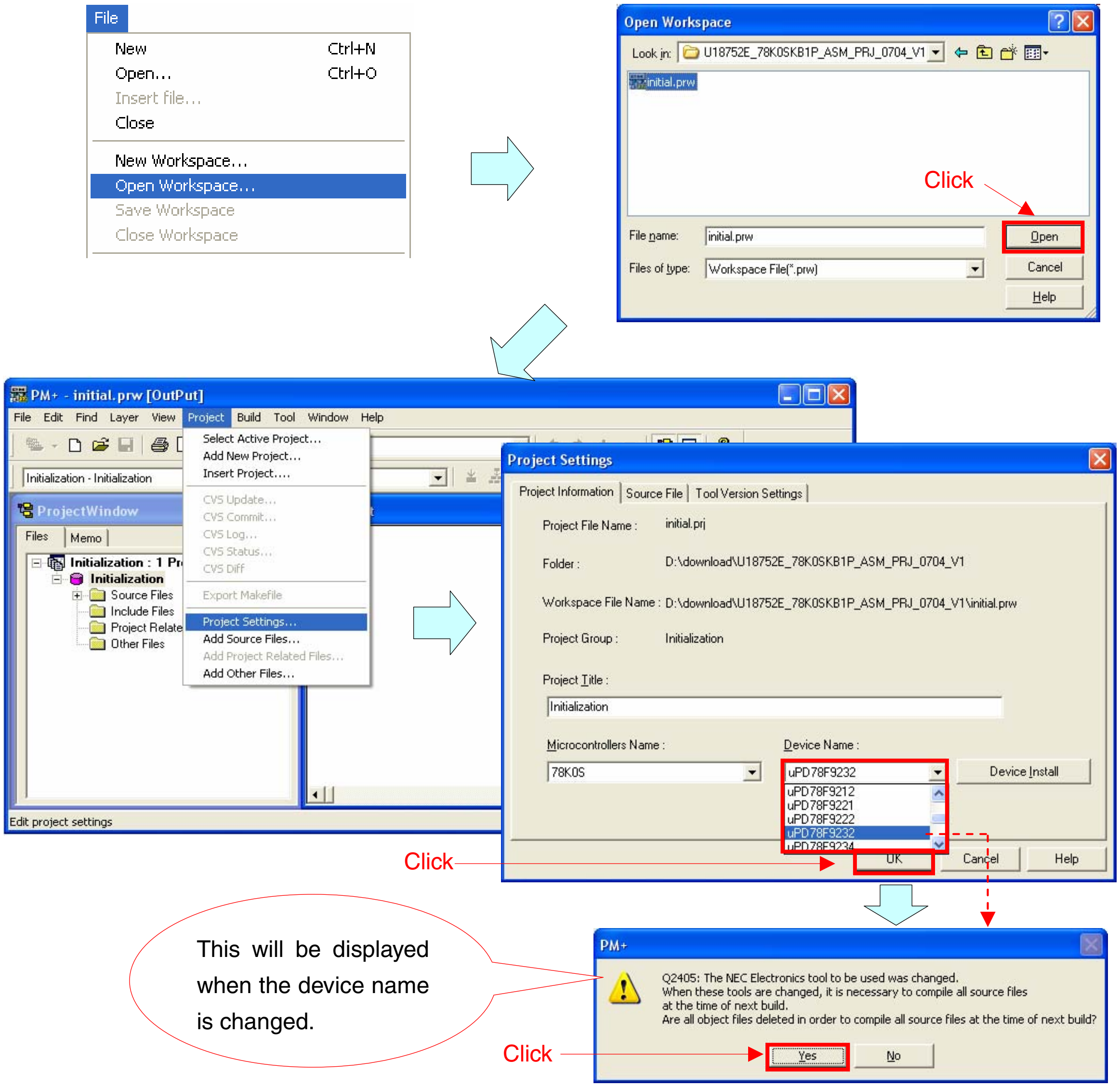
The [Using Fixed Area of Standard Library] check box is unchecked by default when the file that has been downloaded by clicking the icon is used in this sample program.

(1) Start PM+.

(2) Select “uart6.prw” by clicking [Open Workspace] from the [File] menu and click [Open]. A workspace into which the source file will be automatically read will be created.

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

Remark Screenshots of the Sample Program (Initial Settings) LED Lighting Switch Control are shown below.



Look in: U18752E_78K0SKB1P_ASM_PRJ_0704_V1

File name: initial.prw

Files of type: Workspace File(*.prw)

Project File Name : initial.prj

Folder : D:\download\U18752E_78K0SKB1P_ASM_PRJ_0704_V1

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

Project Group : Initialization

Project Title : Initialization

Microcontrollers Name : 78K0S

Device Name : uPD78F9232

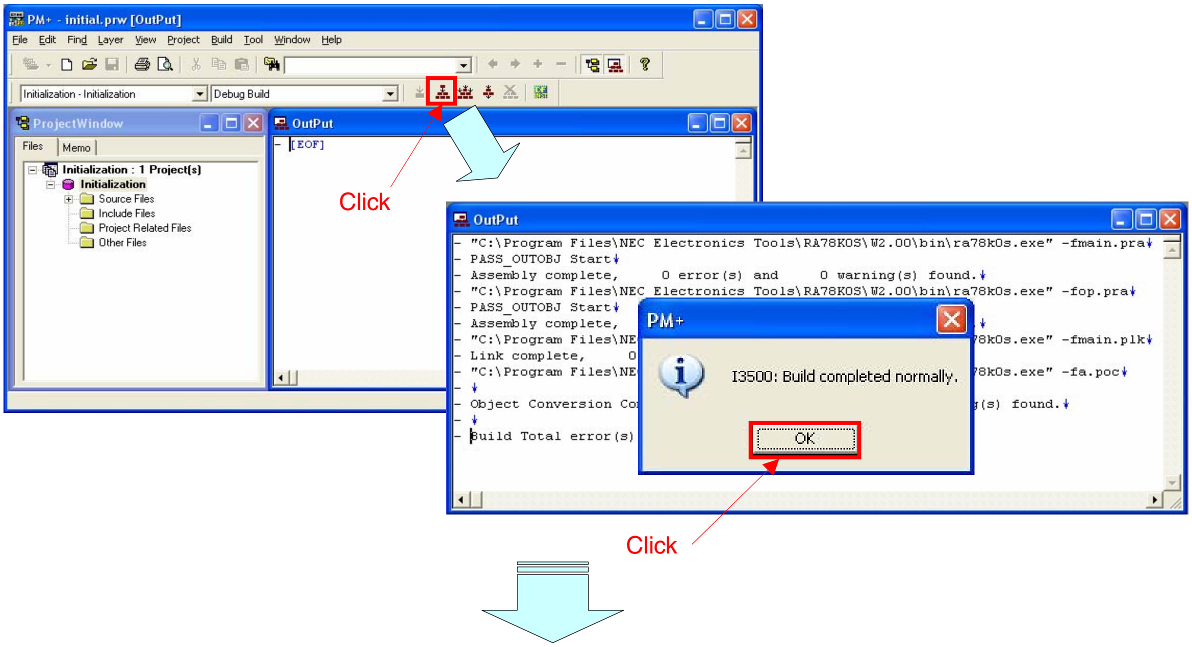
Q2405: The NEC Electronics tool to be used was changed.
When these tools are changed, it is necessary to compile all source files at the time of next build.
Are all object files deleted in order to compile all source files at the time of next build?

This will be displayed when the device name is changed.

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

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

Remark Screenshots of the Sample Program (Initial Settings) LED Lighting Switch Control are shown below.



PM+ - initial.prw [OutPut]
File Edit Find Layer View Project Build Tool Window Help
Initialization - Initialization
Debug Build
ProjectWindow
Files
Memo
Initialization : 1 Project(s)
Initialization
Source Files
Include Files
Project Related Files
Other Files
OutPut
[EOF]
Click
OutPut
PM+
OK
Click

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

I3500: Build completed normally.

A HEX file for flash memory writing will be generated.

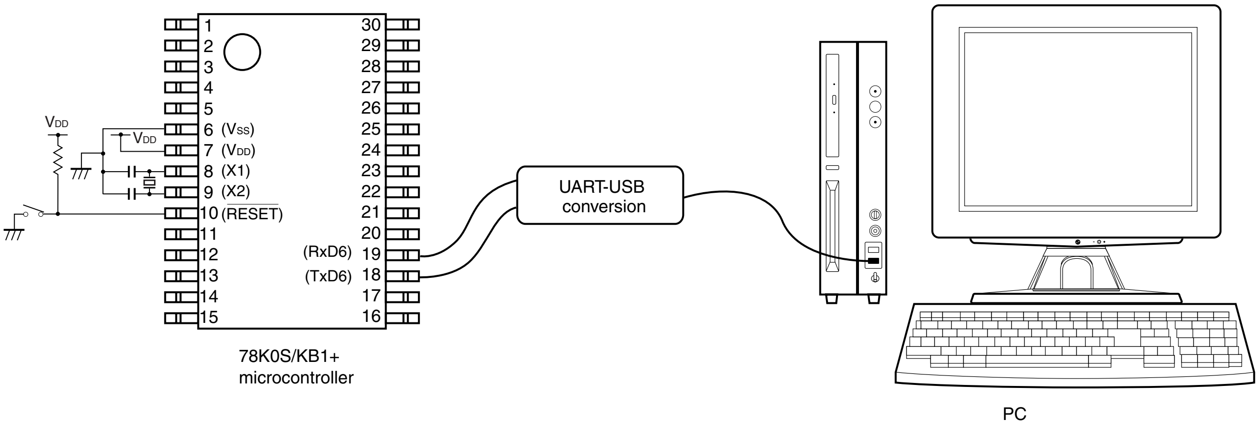
5.2 Operation with the Device

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

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

For how to write to the flash memory of the device, refer to the **Flash Programming Manual (Basic) MINICUBE2 version** of each product **(78K0S/KA1+, 78K0S/KB1+)**.

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



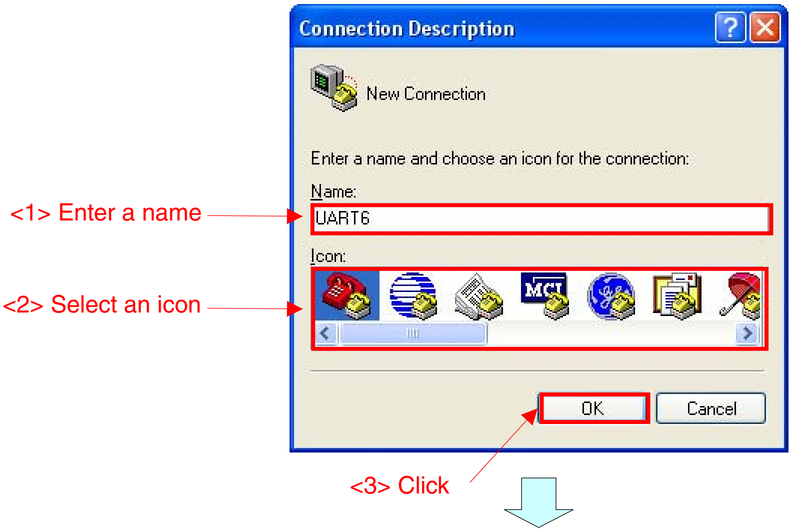
An operation example of when the device to which this sample program has been written is connected as shown above, and Hyper Terminal which is a standard tool provided with Windows™ 2000 and Windows XP™ is used is shown next.

(1) Connect the device to which this sample program has been written as shown above and start Hyper Terminal by using the following procedure.

- Windows 2000: Select in the order of [Start], [Programs], [Accessories], [Communications], and [Hyper Terminal].

- Windows XP: Select in the order of [Start], [All programs], [Accessories], [Communications], and [Hyper Terminal].

(2) The [Connection Description] dialog box will be opened. Enter an arbitrary name (“UART6” in the example below), select an icon (the leftmost icon in the example below), and click the [OK] button.



Connection Description

New Connection

Enter a name and choose an icon for the connection:

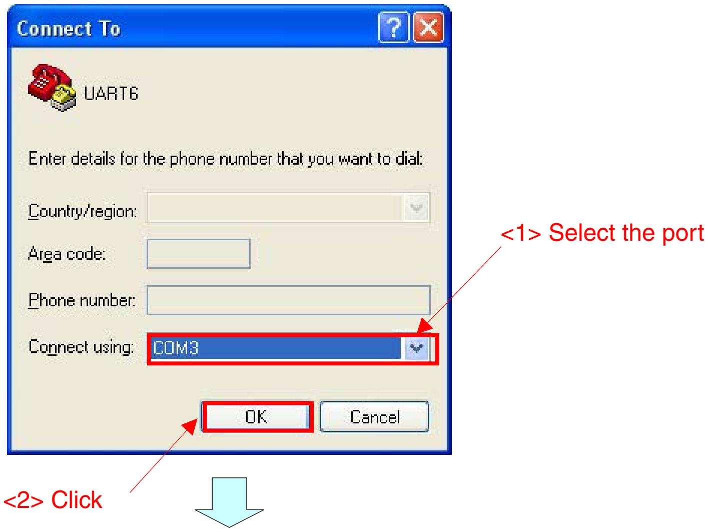
Name: UART6

<1> Enter a name

<2> Select an icon

<3> Click

(3) A new dialog box will be opened. Select the port to which the USB cable is connected (“COM3” in the example below) for the connection method and click the [OK] button.



Connect To

UART6

Enter details for the phone number that you want to dial:

Country/region:

Area code:

Phone number:

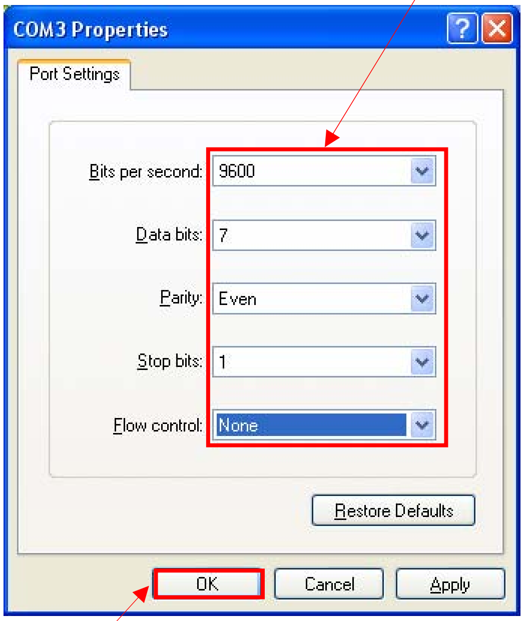
Connect using: COM3

<1> Select the port

<2> Click

(4) The [xxxx Properties] dialog box (xxxx: Port name set in step (3), [COM3 Properties] in the example below) will be opened. Set the communication protocol of the port as shown below and click the [OK] button.

<1> Set the communication protocol of the port



Bits per second: 9600

Data bits: 7

Parity: Even

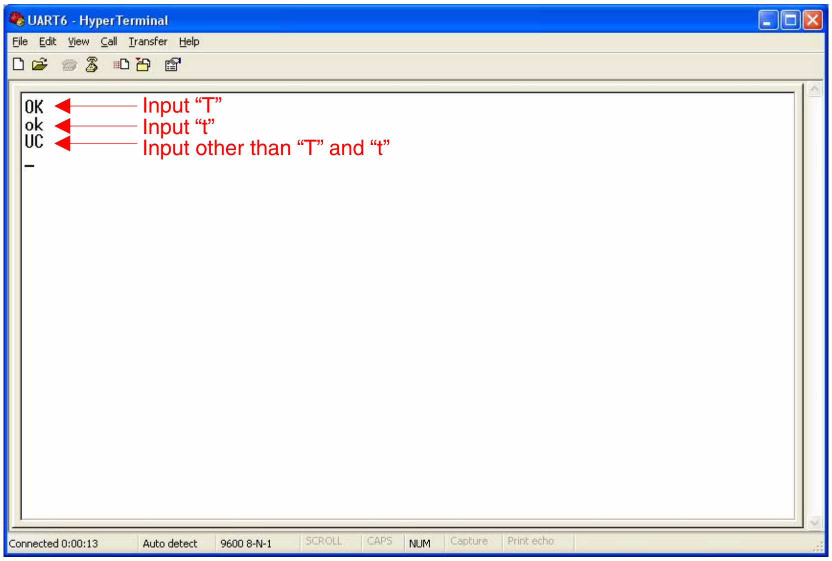
Stop bits: 1

Flow control: None

<2> Click

(5) The [yyyy - Hyper Terminal] window (yyyy: Name of the communication set in step (2), [UART6 - Hyper Terminal] in the example below) will be opened. The characters that will be displayed on the window are as follows, depending on the characters entered by using the keyboard.

Keyboard Entry	Display
T	OK + “line feed”
t	ok + “line feed”
Other than the above	UC + “line feed”



```
OK
ok
UC
_
```

Input “T”

Input “t”

Input other than “T” and “t”

Connected 0:00:13 | Auto detect | 9600 8-N-1 | SCROLL | CAPS | NUM | Capture | Print echo

CHAPTER 6 RELATED DOCUMENTS

Document Name		Japanese/English
78K0S/KA1+ User’s Manual		PDF
78K0S/KB1+ User’s Manual		PDF
78K/0S Series Instructions User’s Manual		PDF
RA78K0S Assembler Package User’s Manual	Language	PDF
	Operation	PDF
CC78K0S C Compiler User’s Manual	Language	PDF
	Operation	PDF
PM+ Project Manager User’s Manual		PDF
SM+ System Simulator Operation User’s Manual		PDF
<R> Flash Programming Manual (Basic) MINICUBE2 version	78K0S/KA1+	PDF
	78K0S/KB1+	PDF
78K0S/Kx1+ Application Note	Sample Program Startup Guide	PDF
	Sample Program (Initial Settings) LED Lighting Switch Control	PDF
	Sample Program (Interrupt) External Interrupt Generated by Switch Input	PDF
	Sample Program (Low-Voltage Detection) Reset Generation During Detection at Less than 2.7 V	PDF

APPENDIX A PROGRAM LIST

As a program list example, the 78K0S/KB1+ microcontroller source program is shown below.

● main.asm (Assembly language version)

```
;******************************************************************************
;
;     NEC Electronics     78K0S/KB1+
;
;******************************************************************************
;     78K0S/KB1+  Sample program
;******************************************************************************
;     Serial interface UART6
;******************************************************************************
;<<History>>
;     2007.8.--   Release
;******************************************************************************
;
;<<Overview>>
;
;This sample program presents an example of using serial interface UART6.
;Serial communication at a baud rate of 9600 bps is performed by using a
;crystal or ceramic oscillation clock of 8 MHz as the system clock source.
;Transmission and reception of ASCII codes are assumed and 4-character data
;is transmitted in accordance with the reception of 1-character data.
;When a reception error occurs, 4-character data is transmitted in
;correspondence with the error, also.
;
;
;  <Principal setting contents>
;
;  - Stop the watchdog timer operation
;  - Set the low-voltage detection voltage (VLVI) to 4.3 V +-0.2 V
;  - Generate an internal reset signal (low-voltage detector) when VDD < VLVI
after VDD >= VLVI
;  - Set the CPU clock to 8 MHz
;  - Set the clock supplied to the peripheral hardware to 8 MHz
;  - Set serial interface UART6
;  - Use the DE and HL registers for interrupt servicing (similarly as a
global variable)
;
;
;  <Serial communication protocol>
;
;  - Baud rate:               9600 bps
;  - Data character length:   7 bits
;  - Parity specification:    Even parity
;  - Number of stop bits:     1 bit
;  - Start bit specification: LSB first
;
;
;  <Receive data>
;
;  The data character length is set to 7 bits and LSB first is set, because
;  the reception of ASCII codes is assumed.  The receive data is therefore
;  transferred to bits 0 to 6 of the RXB6 register and bit 7 (MSB) is always
;  0.  Furthermore, when a reception error occurs, bit 7 of the reception
;  error information is set to 1 and stored into the buffer into which the
;  receive data is also stored.  As a result, the data is identified by bit
;  7 whether it is receive data or reception error information, when the
;  buffer is read.
;
;
;  <Successive reception>
;
;  Successive reception can be performed, because the receive data is stored
;  into the buffer by using an interrupt, and the receive data is sequentially
;  accumulated from the start of the buffer.  Furthermore, the buffer is
;  configured as a ring buffer and the receive data is stored from the start
;  of the buffer again after the end of the buffers has been reached.  At
;  this time, a buffer that has been read will store the receive data, but
;  an unread buffer (when the unread data has reached the buffer size) will
;  discard the receive data instead of storing it.  The receive data will be
;  stored as soon as the buffer data is read.  The buffer size is defined by
;  CBUFFSIZE and is 50 bytes by default.
;
;
;  <Command specifications>
;
;  - Normal reception
;  +---------------------------------------------------+
;  |  Receive Data  |     4-Character Transmit Data     |
;  |   (Hex Data)   |            (Hex Data)             |
;  |---------------------------------------------------|
;  |        T       |   O   |   K   |  "CR"  |  "LF"  |
;  |      (54H)     | (4FH) | (4BH) |  (0DH) |  (0AH) |
;  |---------------------------------------------------|
;  |        t       |   o   |   k   |  "CR"  |  "LF"  |
;  |      (74H)     | (6FH) | (6BH) |  (0DH) |  (0AH) |
;  |---------------------------------------------------|
;  |   Other data   |   U   |   C   |  "CR"  |  "LF"  |
;  |                | (55H) | (43H) |  (0DH) |  (0AH) |
;  +---------------------------------------------------+
;  # "CR" + "LF" is a line feed code.
;
;  - Error reception
;  +---------------------------------------------------+
;  | Error Reception|     4-Character Transmit Data     |
;  |   Information  |            (Hex Data)             |
;  |---------------------------------------------------|
;  | Parity error   |   P   |   E   |  "CR"  |  "LF"  |
;  |                | (50H) | (45H) |  (0DH) |  (0AH) |
;  |---------------------------------------------------|
;  | Framing error  |   F   |   E   |  "CR"  |  "LF"  |
;  |                | (46H) | (45H) |  (0DH) |  (0AH) |
;  |---------------------------------------------------|
;  | Overrun error  |   O   |   E   |  "CR"  |  "LF"  |
;  |                | (4FH) | (45H) |  (0DH) |  (0AH) |
;  +---------------------------------------------------+
;  # "CR" + "LF" is a line feed code.
;
;
;<<I/O port settings>>
;
;  Input: P44
;  Output: P00-P03, P20-P23, P30-P33, P40-P43, P45-P47, P120-P123, P130
;  # All unused ports are set as the output mode.
;
;*****************************************************************************


;=============================================================================
;
;     Define the symbol
;
;=============================================================================
CBUFFSIZE   EQU   50          ; Buffer size of the receive data

;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
;
;     Define the macro
;
; UART transmission processing defines a macro to simplify descriptions.
;
;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
_SEROUT     MACRO RTXDATA

      PUSH  AX                ; Save the AX register data to the stack
      MOV   A,    RTXDATA     ; Store provisional parameter RTXDATA to the A
register
      CALLT [ZTXSUB]          ; Execute the UART transmission subroutine
      POP   AX                ; Restore the AX register data

      ENDM

;=============================================================================
;
;     Vector table
;
;=============================================================================
XVCT  CSEG  AT    0000H
      DW    IRESET            ;(00) RESET
      DW    IRESET            ;(02) --
      DW    IRESET            ;(04) --
      DW    IRESET            ;(06) INTLVI
      DW    IRESET            ;(08) INTP0
      DW    IRESET            ;(0A) INTP1
      DW    IRESET            ;(0C) INTTMH1
      DW    IRESET            ;(0E) INTTM000
      DW    IRESET            ;(10) INTTM010
      DW    IRESET            ;(12) INTAD
      DW    IRESET            ;(14) --
      DW    IRESET            ;(16) INTP2
      DW    IRESET            ;(18) INTP3
      DW    IRESET            ;(1A) INTTM80
      DW    IINTSRE6          ;(1C) INTSRE6
      DW    IINTSR6           ;(1E) INTSR6
      DW    IRESET            ;(20) INTST6

;=============================================================================
;
;     CALLT table
;
;  The instruction code of a frequently called subroutine can be shortened
```

```
;by using the CALLT instruction that is a 1-byte call instruction.  Here,
;an address is registered to the table by using the DW pseudo instruction,
;so that UART transmission subroutine STXSUB can be used by using the CALLT
;instruction.  The use of the subroutine will be enabled by using the address
;(ZTXSUB) and describing CALLT [ZTXSUB].
;
;=============================================================================
XCALT CSEG  CALLT0
ZTXSUB:           DW    STXSUB      ; UART transmission subroutine

;=============================================================================
;
;     Define the RAM
;
;=============================================================================
DRAM1 DSEG  UNIT
RRXBUFTOP:  DS    CBUFFSIZE   ; Receive data buffer
RRXBUFEND:

DRAM2 DSEG  SADDR
RRXCNT:           DS    1     ; Reception count variable

;=============================================================================
;
;     Define the memory stack area
;
;=============================================================================
DSTK  DSEG  AT    0FEE0H
RSTACKEND:  DS    20H         ; Memory stack area = 32 bytes
RSTACKTOP:                    ; Start address of the memory stack area = FF00H

;*****************************************************************************
;
;     Initialization after RESET
;
;*****************************************************************************
XMAIN CSEG  UNIT
IRESET:
;-----------------------------------------------------------------------------
;     Initialize the stack pointer
;-----------------------------------------------------------------------------
      MOVW  AX,   #RSTACKTOP
      MOVW  SP,   AX          ; Set the stack pointer

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

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

;-----  Set the clock <1>  -----
      MOV   PCC,  #00000000B  ; The clock supplied to the CPU (fcpu) = fxp (=
fx/4 = 2 MHz)
      MOV   LSRCM,      #00000001B  ; Stop the oscillation of the low-speed
internal oscillator
```

```
;-----  Check the reset source  -----
      MOV   A,    RESF        ; Read the reset source
      BT    A.0,  $HRST300    ; Omit subsequent LVI-related processing and go
to SET_CLOCK during LVI reset

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

      MOV   A,    #40         ; Assign the 200 us wait count value
;-----  200 us wait  -----
HRST100:
      DEC   A
      BNZ   $HRST100          ; 0.5[us/clk] x 10[clk] x 40[count] = 200[us]

;-----  VDD >= VLVI wait processing  -----
HRST200:
      NOP
      BT    LVIF, $HRST200    ; Branch if VDD < VLVI

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

;-----  Set the clock <2>  -----
HRST300:
      MOV   PPCC, #00000000B  ; The clock supplied to the peripheral hardware
(fxp) = fx (= 8 MHz)
                              ; -> The clock supplied to the CPU (fcpu) = fxp
= 8 MHz

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

;-----------------------------------------------------------------------------
;     Initialize the port 2
;-----------------------------------------------------------------------------
      MOV   P2,   #00000000B  ; Set output latches of P20-P23 as low
      MOV   PM2,  #11110000B  ; Set P20-P23 as output mode

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

;-----------------------------------------------------------------------------
;     Initialize the port 4
;-----------------------------------------------------------------------------
      MOV   P4,   #00001000B  ; Set output latches of P40-P42 and P44-P47 as
low, output latch of P43 as high (setting for serial transmission)
      MOV   PM4,  #00010000B  ; Set P40-P43 and P45-P47 as output mode, P44 as
input mode

;-----------------------------------------------------------------------------
;     Initialize the port 12
```

```
;-----------------------------------------------------------------------------
      MOV   P12,  #00000000B  ; Set output latches of P120-P123 as low
      MOV   PM12, #11110000B  ; Set P120-P123 as output mode

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

;-----------------------------------------------------------------------------
;     Set UART6
;-----------------------------------------------------------------------------
      MOV   CKSR6,      #1    ; Set the baud rate to 9600 bps
      MOV   BRGC6,      #208  ; (Same as the above)
      MOV   ASIM6,      #00011000B  ; Even-parity output, 7-bit character
length, 1 stop bit
                              ; INTSRE6 is generated as interrupt upon error
occurrence
      SET1  POWER6            ; Enable internal operation clock operation
      SET1  TXE6              ; Enable transmit operation
      SET1  RXE6              ; Enable receive operation

;*****************************************************************************
;
;     Main loop
;
;*****************************************************************************
MMAINLOOP:

;-----  Initialize the RAM and general-purpose register  -----
      MOV   RRXCNT,     #0    ; Reception count = 0
      MOVW  HL,   #RRXBUFTOP  ; Initialize the write address to the buffer
start
      MOVW  DE,   #RRXBUFTOP  ; Initialize the read address to the buffer
start

;-----  Set the interrupts  -----
      MOV   IF0,  #00H        ; Clear invalid interrupt requests in advance
      CLR1  SRMK6             ; Enable the INTSR6 (serial reception) interrupt
      CLR1  SREMK6            ; Enable the INTSRE6 (reception error) interrupt
      EI                      ; Enable vector interrupt

;-----  Wait for a reception interrupt  -----
LMLP100:
      CMP   RRXCNT,     #0
      BZ    $LMLP100          ; Branch if the reception count is 0

      MOV   A, [DE]           ; Read the data
      DEC   RRXCNT            ; Decrement the reception count by 1

      BT    A.7,  $LMLP300    ; Branch to processing when a reception error
occurs if bit 7 is 1

;-----  Processing during normal reception  -----
LMLP200:
      CMP   A, #'T'           ; Compare the receive data with T (54H)
      BNZ   $LMLP210          ; Branch if the receive data is not T (54H)
      _SEROUT     #'O'        ; Transmit O (4FH)
      _SEROUT     #'K'        ; Transmit K (4BH)
      _SEROUT     #0DH        ; Transmit line feed code "CR"
      _SEROUT     #0AH        ; Transmit line feed code "LF"
      BR    !LMLP400          ; Branch to LMLP400

LMLP210:
      CMP   A, #'t'           ; Compare the receive data with t (74H)
      BNZ   $LMLP220          ; Branch if the receive data is not t (74H)
      _SEROUT     #'o'        ; Transmit o (6FH)
      _SEROUT     #'k'        ; Transmit k (6BH)
      _SEROUT     #0DH        ; Transmit line feed code "CR"
      _SEROUT     #0AH        ; Transmit line feed code "LF"
      BR    !LMLP400          ; Branch to LMLP400

LMLP220:
      _SEROUT     #'U'        ; Transmit U (55H)
      _SEROUT     #'C'        ; Transmit C (43H)
      _SEROUT     #0DH        ; Transmit line feed code "CR"
      _SEROUT     #0AH        ; Transmit line feed code "LF"
      BR    !LMLP400          ; Branch to LMLP400

;-----  Processing when a reception error occurs  -----
LMLP300:
      BF    A.2, $LMLP310     ; Branch if not a parity error
      _SEROUT     #'P'        ; Transmit P (50H)
      _SEROUT     #'E'        ; Transmit E (45H)
      _SEROUT     #0DH        ; Transmit line feed code "CR"
      _SEROUT     #0AH        ; Transmit line feed code "LF"

LMLP310:
      BF    A.1, $LMLP320     ; Branch if not a framing error
      _SEROUT     #'F'        ; Transmit F (46H)
      _SEROUT     #'E'        ; Transmit E (45H)
      _SEROUT     #0DH        ; Transmit line feed code "CR"
      _SEROUT     #0AH        ; Transmit line feed code "LF"

LMLP320:
      BF    A.0, $LMLP400     ; Branch if not an overrun error
      _SEROUT     #'O'        ; Transmit O (4FH)
      _SEROUT     #'E'        ; Transmit E (45H)
      _SEROUT     #0DH        ; Transmit line feed code "CR"
      _SEROUT     #0AH        ; Transmit line feed code "LF"

;-----  Update the read address  -----
LMLP400:
      INCW  DE                ; Increment the read address by 1
      MOVW  AX,   DE
      CMPW  AX,   #RRXBUFEND
      BC    $LMLP450          ; Branch if the read address is within the
buffer
      MOVW  DE,   #RRXBUFTOP  ; Initialize the read address to the buffer
start
LMLP450:
      BR    !LMLP100          ; Branch to LMLP100

;*****************************************************************************
;
;     Serial reception interrupt INTSR6
;
;*****************************************************************************
```

```
IINTSR6:
      PUSH  AX                ; Save the AX register data to the stack

;-----  Read the receive data  -----
      MOV   A, RXB6           ; Read the serial receive data

;-----  Check the free buffer space  -----
      CMP   RRXCNT,     #CBUFFSIZE  ; Compare the reception count with the
buffer size
      BNC   $HSR100           ; Do not store the data if no free space is
available in the buffer
      INC   RRXCNT            ; Increment the reception count by 1

;-----  Save the data and update the write address  -----
      MOV  [HL], A            ; Store the receive data
      INCW  HL                ; Increment the write address by 1
      MOVW  AX,   HL
      CMPW  AX,   #RRXBUFEND
      BC    $HSR100           ; Branch if the write address is within the
buffer
      MOVW  HL,   #RRXBUFTOP  ; Initialize the write address to the buffer
start

HSR100:
      POP   AX                ; Restore the AX register data
      RETI                    ; Return from interrupt servicing

;*****************************************************************************
;
;     Reception error interrupt INTSRE6
;
;*****************************************************************************
IINTSRE6:
      PUSH  AX                ; Save the AX register data to the stack

;-----  Read the error status  -----
      MOV   A, ASIS6          ; Read the error status
      SET1  A.7               ; Set the reception error flag to bit 7
      XCH   A,    X           ; Save the error information
      MOV   A, RXB6           ; Read (discard) the serial receive data
      XCH   A,    X           ; Restore the error information

;-----  Check the free buffer space  -----
      CMP   RRXCNT,     #CBUFFSIZE  ; Compare the reception count with the
buffer size
      BNC   $HSRE100          ; Do not store the data if no free space is
available in the buffer
      INC   RRXCNT            ; Increment the reception count by 1

;-----  Save the data and update the write address  -----
      MOV  [HL], A            ; Store the error status
      INCW  HL                ; Increment the write address by 1
      MOVW  AX,   HL
      CMPW  AX,   #RRXBUFEND
      BC    $HSR100           ; Branch if the write address is within the
buffer
      MOVW  HL,   #RRXBUFTOP  ; Initialize the write address to the buffer
start
```

```
HSRE100:
      POP   AX          ; Restore the AX register data
      RETI

;=============================================================================
;     Subroutine for serial data transmission
;
; This subroutine is used to transmit serial data.
; 1-byte data indicated with #DATA can be transmitted by setting the
; subroutine as follows.
;
;     MOV   A,    #DATA ; Store DATA to the A register
;     CALL  !STXSUB     ; Transmit DATA
;=============================================================================
STXSUB:
      XCH   A,    X     ; Save the transmit data to the X register

;-----  Wait for transmission to be enabled  -----
JTXS100:
      MOV   A, ASIF6
      BT    A.1, $JTXS100     ; Wait for transmission to be enabled

;-----  Transmit the data  -----
      XCH   A,    X     ; Restore the transmit data from the X register
      MOV   TXB6, A     ; Serial transmission

      RET               ; Return from the subroutine

end
```

● main.c (C language version)

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

      NEC Electronics     78K0S/KB1+

******************************************************************************
      78K0S/KB1+  Sample program
******************************************************************************
      Serial interface UART6
******************************************************************************
<<History>>
      2007.8.--   Release
******************************************************************************

<<Overview>>

This sample program presents an example of using serial interface UART6.
Serial communication at a baud rate of 9600 bps is performed by using a
crystal or ceramic oscillation clock of 8 MHz as the system clock source.
Transmission and reception of ASCII codes are assumed and 4-character data
is transmitted in accordance with the reception of 1-character data.
When a reception error occurs, 4-character data is transmitted in
correspondence with the error, also.


  <Principal setting contents>

  - Declare a function run by an interrupt: INTSR6 -> fn_intsr6()
  - Declare a function run by an interrupt: INTSRE6 -> fn_intsre6()
  - Stop the watchdog timer operation
  - Set the low-voltage detection voltage (VLVI) to 4.3 V +-0.2 V
  - Generate an internal reset signal (low-voltage detector) when VDD < VLVI
after VDD >= VLVI
  - Set the CPU clock to 8 MHz
  - Set the clock supplied to the peripheral hardware to 8 MHz
  - Set serial interface UART6


  <Serial communication protocol>

  - Baud rate:                9600 bps
  - Data character length:    7 bits
  - Parity specification:     Even parity
  - Number of stop bits:      1 bit
  - Start bit specification:  LSB first


  <Receive data>

  The data character length is set to 7 bits and LSB first is set, because
  the reception of ASCII codes is assumed.  The receive data is therefore
  transferred to bits 0 to 6 of the RXB6 register and bit 7 (MSB) is always
  0.  Furthermore, when a reception error occurs, bit 7 of the reception
  error information is set to 1 and stored into the buffer into which the
  receive data is also stored.  As a result, the data is identified by bit
  7 whether it is receive data or reception error information, when the
  buffer is read.
  <Successive reception>

  Successive reception can be performed, because the receive data is stored
  into the buffer by using an interrupt, and the receive data is sequentially
  accumulated from the start of the buffer.  Furthermore, the buffer is
  configured as a ring buffer and the receive data is stored from the start
  of the buffer again after the end of the buffers has been reached.  At
  this time, a buffer that has been read will store the receive data, but
  an unread buffer (when the unread data has reached the buffer size) will
  discard the receive data instead of storing it.  The receive data will be
  stored as soon as the buffer data is read.  The buffer size is defined by
  BUFF_SIZE and is 50 bytes by default.


  <Command specifications>

  - Normal reception
  +----------------------------------------------------+
  |  Receive Data  |     4-Character Transmit Data     |
  |   (Hex Data)   |            (Hex Data)             |
  |----------------------------------------------------|
  |        T       |    O   |    K   |  "CR"  |  "LF"  |
  |      (54H)     |  (4FH) |  (4BH) |  (0DH) |  (0AH) |
  |----------------------------------------------------|
  |        t       |    o   |    k   |  "CR"  |  "LF"  |
  |      (74H)     |  (6FH) |  (6BH) |  (0DH) |  (0AH) |
  |----------------------------------------------------|
  |   Other data   |    U   |    C   |  "CR"  |  "LF"  |
  |                |  (55H) |  (43H) |  (0DH) |  (0AH) |
  +----------------------------------------------------+
  # "CR" + "LF" is a line feed code.

  - Error reception
  +----------------------------------------------------+
  | Error Reception|     4-Character Transmit Data     |
  |   Information  |            (Hex Data)             |
  |----------------------------------------------------|
  | Parity error   |    P   |    E   |  "CR"  |  "LF"  |
  |                |  (50H) |  (45H) |  (0DH) |  (0AH) |
  |----------------------------------------------------|
  | Framing error  |    F   |    E   |  "CR"  |  "LF"  |
  |                |  (46H) |  (45H) |  (0DH) |  (0AH) |
  |----------------------------------------------------|
  | Overrun error  |    O   |    E   |  "CR"  |  "LF"  |
  |                |  (4FH) |  (45H) |  (0DH) |  (0AH) |
  +----------------------------------------------------+
  # "CR" + "LF" is a line feed code.


<<I/O port settings>>

  Input: P44
  Output: P00-P03, P20-P23, P30-P33, P40-P43, P45-P47, P120-P123, P130
  # All unused ports are set as the output mode.

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

```
/*============================================================================

      Preprocessing directive (#pragma)

============================================================================*/
#pragma     SFR                     /* SFR names can be described at the C
source level */
#pragma     EI                      /* EI instructions can be described at the
C source level */
#pragma     NOP                     /* NOP instructions can be described at
the C source level */
#pragma interrupt INTSR6 fn_intsr6  /* Interrupt function declaration:INTSR6
*/
#pragma interrupt INTSRE6 fn_intsre6/* Interrupt function declaration:INTSRE6
*/

#define BUFF_SIZE 50                /* Buffer size of the receive data */

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

      Function prototype declaration

============================================================================*/
void fn_uart_send(unsigned char ucTxData);      /* Function for serial data
transmission */

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

      Define the global variables

============================================================================*/
static unsigned char g_ucRxBuff[BUFF_SIZE];     /* Receive data buffer table
*/
sreg unsigned char g_ucRxCnt;       /* 8-bit variable for reception count */
sreg unsigned char g_ucStoreAddr;   /* 8-bit variable for write address */
sreg unsigned char g_ucReadAddr;    /* 8-bit variable for read address */
sreg unsigned char g_ucRxData;      /* 8-bit variable for identifying the
receive data */
sreg unsigned char g_ucAsif6;       /* 8-bit variable for identifying the
transmission status */

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

      Initialization after RESET

*****************************************************************************/
void hdwinit(void){
      unsigned char ucCnt200us;     /* 8-bit variable for 200 us wait */

/*----------------------------------------------------------------------------
      Initialize the watchdog timer + detect low-voltage + set the clock
----------------------------------------------------------------------------*/
      /* Initialize the watchdog timer */
      WDTM  = 0b01110111;           /* Stop the watchdog timer operation */

      /* Set the clock <1> */
      PCC   = 0b00000000;           /* The clock supplied to the CPU (fcpu) =
fxp (= fx/4 = 2 MHz) */
```

```
      LSRCM = 0b00000001;     /* Stop the oscillation of the low-speed
internal oscillator */

      /* Check the reset source */
      if (!(RESF & 0b00000001)){    /* Omit subsequent LVI-related processing
during LVI reset */

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

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

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

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

      /* Set the clock <2> */
      PPCC  = 0b00000000;     /* The clock supplied to the peripheral hardware
(fxp) = fx (= 8 MHz)
                                 -> The clock supplied to the CPU (fcpu) = fxp
= 8 MHz */

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

/*------------------------------------------------------------------------------
      Initialize the port 2
--------------------------------------------------------------------------------
*/
      P2    = 0b00000000;     /* Set output latches of P20-P23 as low */
      PM2   = 0b11110000;     /* Set P20-P23 as output mode */

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

/*------------------------------------------------------------------------------
      Initialize the port 4
------------------------------------------------------------------------------*/
      P4    = 0b00001000;     /* Set output latches of P40-P42 and P44-P47 as
low, output latch of P43 as high (setting for serial transmission) */
      PM4   = 0b00010000;     /* Set P40-P43 and P45-P47 as output mode, P44
as input mode */

/*------------------------------------------------------------------------------
```

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

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

/*----------------------------------------------------------------------------
      Set UART6
----------------------------------------------------------------------------*/
      CKSR6 = 1;              /* Set the baud rate to 9600 bps */
      BRGC6 = 208;            /* (Same as the above) */
      ASIM6 = 0b00011000;     /* Even-parity output, 7-bit character length, 1
stop bit */
                              /* INTSRE6 is generated as interrupt upon error
occurrence */
      POWER6 = 1;             /* Enable internal operation clock operation */
      TXE6  =     1;          /* Enable transmit operation */
      RXE6  =     1;          /* Enable receive operation */

      return;
}

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

      Main loop

*****************************************************************************/
void main(void)
{
      g_ucRxCnt = 0;          /* Reception count = 0 */
      g_ucStoreAddr = 0;      /* Initialize the write address to the buffer
start */
      g_ucReadAddr = 0;       /* Initialize the read address to the buffer
start */
      IF0 = 0x00;             /* Clear invalid interrupt requests in advance
*/
      SRMK6 = 0;              /* Enable the INTSR6 (serial reception)
interrupt */
      SREMK6 = 0;             /* Enable the INTSRE6 (reception error)
interrupt */
      EI();                   /* Enable vector interrupt */

      while (1)
      {
            while (g_ucRxCnt == 0)  /* Wait for a reception interrupt */
            {
                  NOP();
            }

            while (g_ucRxCnt > 0)   /* Processing when the reception count > 0
*/
            {
                  g_ucRxData = g_ucRxBuff[g_ucReadAddr];    /* Read the data
*/
```

```
                  g_ucRxCnt -= 1;         /* Decrement the reception count by
1 */

                  if (!g_ucRxData.7)      /* Processing during normal
reception */
                  {
                        switch (g_ucRxData)
                        {
                              case 'T' :  /* When receiving T (54H) */

                                    fn_uart_send('O');      /* Transmit O
(4FH) */
                                    fn_uart_send('K');      /* Transmit K
(4BH) */
                                    fn_uart_send(0x0D);     /* Transmit line
feed code "CR" */
                                    fn_uart_send(0x0A);     /* Transmit line
feed code "LF" */
                                    break;

                              case 't' :  /* When receiving t (74H) */

                                    fn_uart_send('o');      /* Transmit o
(6FH) */
                                    fn_uart_send('k');      /* Transmit k
(6BH) */
                                    fn_uart_send(0x0D);     /* Transmit line
feed code "CR" */
                                    fn_uart_send(0x0A);     /* Transmit line
feed code "LF" */
                                    break;

                              default  :  /* When receiving other data */

                                    fn_uart_send('U');      /* Transmit U
(55H) */
                                    fn_uart_send('C');      /* Transmit C
(43H) */
                                    fn_uart_send(0x0D);     /* Transmit line
feed code "CR" */
                                    fn_uart_send(0x0A);     /* Transmit line
feed code "LF" */
                                    break;
                        }
                  }

                  else              /* Processing when receiving an error */
                  {
                        if (g_ucRxData.2) /* When a parity error occurs */
                        {
                              fn_uart_send('P');      /* Transmit P (50H) */
                              fn_uart_send('E');      /* Transmit E (45H) */
                              fn_uart_send(0x0D);     /* Transmit line feed
code "CR" */
                              fn_uart_send(0x0A);     /* Transmit line feed
code "LF" */
                        }

                        if (g_ucRxData.1) /* When a framing error occurs */
                        {
                              fn_uart_send('F');      /* Transmit F (46H) */
                              fn_uart_send('E');      /* Transmit E (45H) */
                              fn_uart_send(0x0D);     /* Transmit line feed
code "CR" */
                              fn_uart_send(0x0A);     /* Transmit line feed
code "LF" */
                        }

                        if (g_ucRxData.0) /* When an overrun error occurs */
                        {
                              fn_uart_send('O');      /* Transmit O (4FH) */
                              fn_uart_send('E');      /* Transmit E (45H) */
                              fn_uart_send(0x0D);     /* Transmit line feed
code "CR" */
                              fn_uart_send(0x0A);     /* Transmit line feed
code "LF" */
                        }
                  }
                  g_ucReadAddr += 1;            /* Increment the read address
by 1 */

                  if (g_ucReadAddr >= BUFF_SIZE)      /* When the read address
is outside the buffer */
                  {
                        g_ucReadAddr = 0;       /* Initialize the read address
to the buffer start */
                  }
            }
      }
}

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

      Serial reception interrupt INTSR6

******************************************************************************/
__interrupt void fn_intsr6()
{
      unsigned char ucData;

      ucData = RXB6;                /* Read the serial receive data */

      if (g_ucRxCnt < BUFF_SIZE)    /* When the write address is within the
buffer */
      {
            g_ucRxCnt += 1;                     /* Increment the reception
count by 1 */
            g_ucRxBuff[g_ucStoreAddr] = ucData; /* Save the receive data */

            g_ucStoreAddr += 1;                 /* Increment the write address
by 1 */

            if (g_ucStoreAddr >= BUFF_SIZE)     /* When the write address is
outside the buffer */
            {
                  g_ucStoreAddr = 0;            /* Initialize the write
address to the buffer start */
            }
      }

      return;
}

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

      Reception error interrupt INTSRE6

*****************************************************************************/
__interrupt void fn_intsre6()
{
      unsigned char ucData;
      unsigned char ucTemp;

      ucData = ASIS6 | 0b10000000;  /* Store the error information by setting
the error flag to bit 7 */
      ucTemp = RXB6;                /* Read (discard) the serial receive data
*/

      if (g_ucRxCnt < BUFF_SIZE)    /* When the write address is within the
buffer */
      {
            g_ucRxCnt += 1;                     /* Increment the reception
count by 1 */
            g_ucRxBuff[g_ucStoreAddr] = ucData; /* Save the receive data */

            g_ucStoreAddr += 1;                 /* Increment the write address
by 1 */

            if (g_ucStoreAddr >= BUFF_SIZE)     /* When the write address is
outside the buffer */
            {
                  g_ucStoreAddr = 0;            /* Initialize the write
address to the buffer start */
            }
      }

      return;
}

/*============================================================================
      Function for serial data transmission

 This function is used to transmit serial data.
 1-byte data indicated with Data can be transmitted by using the function
 as follows.

      fn_uart_send(Data);
============================================================================*/
void fn_uart_send(unsigned char ucTxData)
{
      g_ucAsif6 = ASIF6;            /* Read the transmission status */

      while (g_ucAsif6.1)           /* Wait for transmission to be enabled */
      {
            g_ucAsif6 = ASIF6;      /* Read the transmission status */
      }
      TXB6 = ucTxData;              /* Serial transmission */
      return;
}
```

● op.asm (Common to assembly language and C language versions)

```
;==============================================================================
;
;     Option byte
;
;==============================================================================
OPBT  CSEG  AT    0080H
        DB      10011000B       ; Option byte area
;                || ||||
;                || |||+---------- Low-speed internal oscillator can be
stopped by software
;                || |++----------- Crystal or ceramic oscillation clock is
used
;                || +------------- P34/RESET pin is used as RESET pin
;                ++--------------- Oscillation stabilization time when turning
power on or after reset release = 2^10/fx

        DB      11111111B       ; Protect byte area (for the self programming
mode)
;               ||||||||
;               ++++++++---------- All blocks can be written or erased

end
```

APPENDIX B REVISION HISTORY

The mark “<R>” shows major revised points. The revised points can be easily searched by copying an “<R>” in the PDF file and specifying it in the “Find what.” field.

Edition	Date Published	Page	Revision
1st edition	December 2007	–	–
2nd edition	September 2008	pp.27 to 29	Modification of 5.1 Building the Sample Program
		p.33	CHAPTER 6 RELATED DOCUMENTS • Addition of Flash Programming Manual (Basic) MINICUBE2 version

For further information,
please contact:

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

[America]

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

[Europe]

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

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

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

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

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

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

Sucursal en España
Juan Esplandiu, 15
28007 Madrid, Spain
Tel: 091-504-2787

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

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

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

[Asia & Oceania]

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

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

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

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

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

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

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