To our customers,

Old Company Name in Catalogs and Other Documents

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

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

April 1st, 2010
Renesas Electronics Corporation

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

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

Notice

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

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

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

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

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

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

Application Note

78K0S/Kx1+

Sample Program
(Software UART Half-Duplex Communication)

Internal High-Speed Oscillation Clock & Calibration

This document describes an operation overview of the sample program and how to use it, as well as how to set and use UART half-duplex communication by software control. In the sample program, the baud rate is determined by performing calibration after completion of the initial settings. Afterward, 8 units of data are received as a reception test, which are transmitted as a transmission test.

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

Document No. U18916EJ2V0AN00 (2nd edition)
Date Published September 2008 NS

CONTENTS

- **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 in which UART half-duplex communication is performed by adjusting the communication timing and controlling the ports by using software, regardless of whether the microcontroller is provided with a UART function and without using the function if it is provided (hereinafter, this communication is referred to as “software UART”). Software UART is used to perform serial communication with products that are not provided with serial interface UART6 (78K0S/KU1+, 78K0S/KY1+) or increase the number of serial communication channels for products that are provided with serial interface UART6 (78K0S/KA1+, 78K0S/KB1+).

Calibration is performed after completion of the initial settings and the baud rate is determined by receiving a low level (80H) equivalent to 8 bits. Afterward, 8 units of data are received as a reception test, which are transmitted as a transmission test. Furthermore, an LED is turned on during transmission and reception.

1.1 Main Contents of the Initial Settings

The main contents of the initial settings are as follows.

- Selecting the high-speed internal oscillator as the system clock source[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

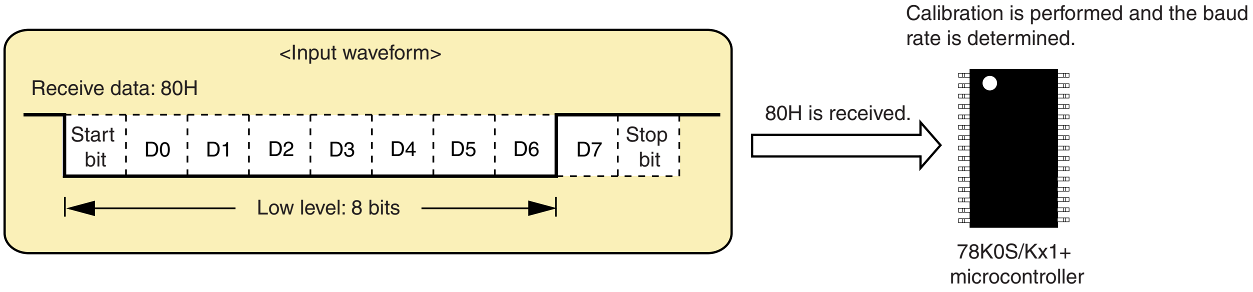
Note This is set by using the option byte.

[Column] What is half-duplex communication?
Half-duplex communication is a type of communication in which a receive operation and a transmit operation are performed alternately. In this sample program, half-duplex communication is used, which enables a receive operation and a transmit operation by using software.

1.2 Contents Following the Main Loop

Calibration is performed after completion of the initial settings and the baud rate is determined by receiving a low level (80H) equivalent to 8 bits.

● Calibration



After completion of calibration, 8 units of data are received as a reception test, which are transmitted as a transmission test. Furthermore, an LED is turned on during transmission and reception.

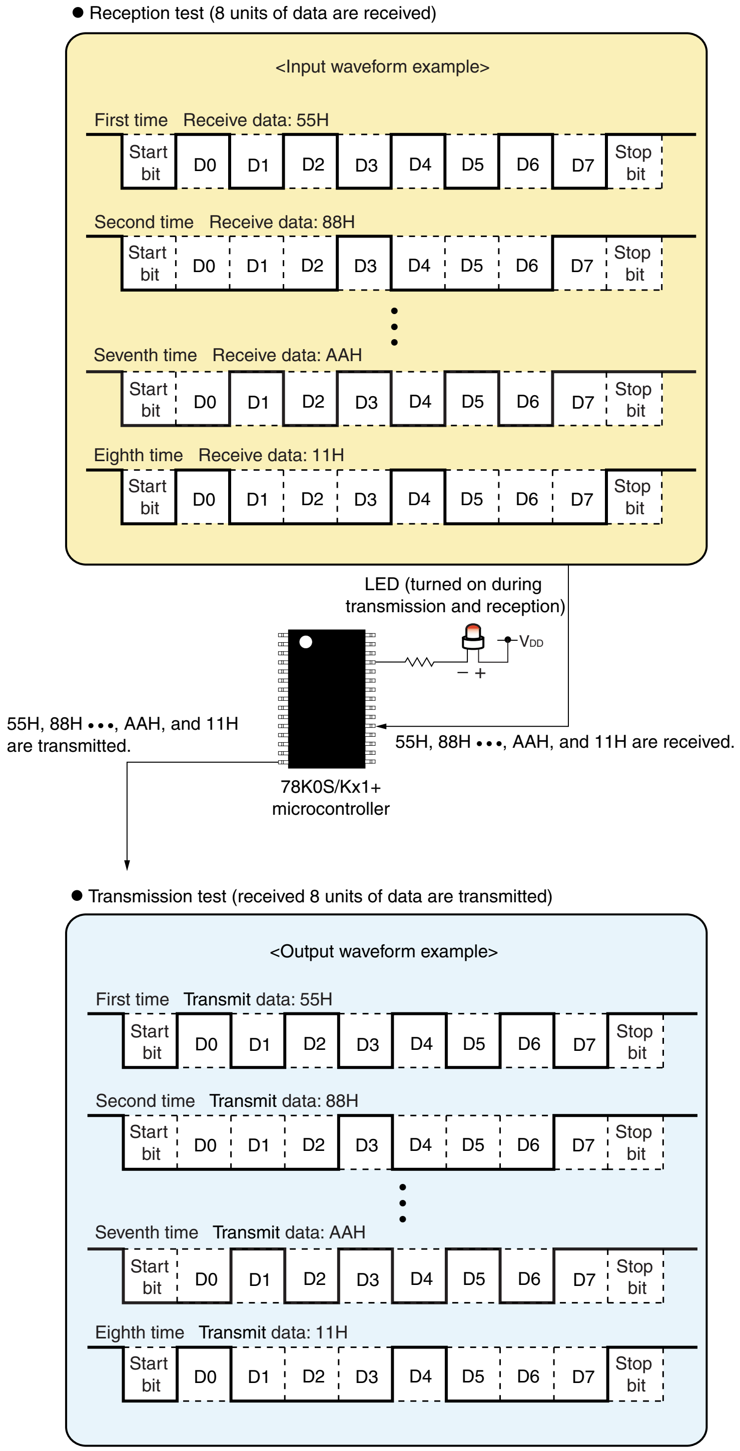
The communication protocol will be set as follows.

- Baud rate: 4,800 to 19,200 bps[Note 1]
- Data character length: 8 bits
- Parity specification: No parity
- Number of stop bits: 1 bit or 2 bits[Note 2]
- Start bit specification: LSB first

Notes 1. The baud rate value is determined by performing calibration. When not performing calibration, the baud rate value can be set by using software. The default value is 9,600 bps.

2. This can be set by using software. The default setting is 1 bit.

● Reception test → transmission test


● Reception test (8 units of data are received)
<Input waveform example>
First time Receive data: 55H
Second time Receive data: 88H
Seventh time Receive data: AAH
Eighth time Receive data: 11H
Start bit
D0
D1
D2
D3
D4
D5
D6
D7
Stop bit
LED (turned on during transmission and reception)
VDD
− +
55H, 88H •••, AAH, and 11H are transmitted.
55H, 88H •••, AAH, and 11H are received.
78K0S/Kx1+ microcontroller
● Transmission test (received 8 units of data are transmitted)
<Output waveform example>
First time Transmit data: 55H
Second time Transmit data: 88H
Seventh time Transmit data: AAH
Eighth time Transmit data: 11H

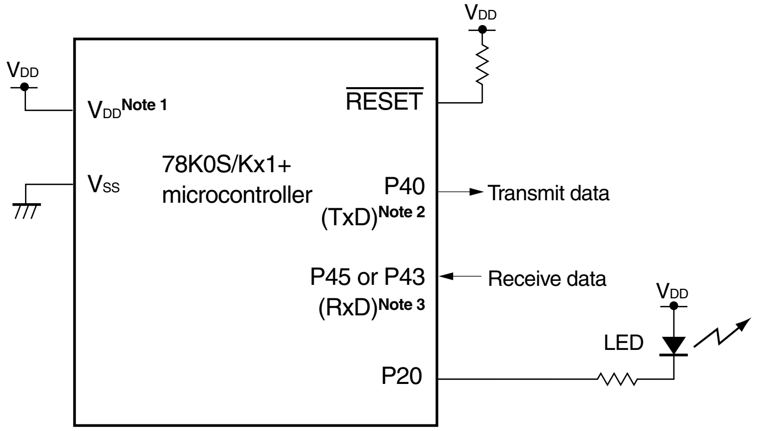
Caution For cautions when using the device, refer to the user's manual of each product (78K0S/KU1+, 78K0S/KY1+, 78K0S/KA1+, 78K0S/KB1+).

CHAPTER 2 CIRCUIT DIAGRAM

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

2.1 Circuit Diagram

A circuit diagram is shown below.



Notes 1. Use this in a voltage range of 4.5 V ≤ V_{DD} ≤ 5.5 V.

2. The P40 pin is used as the UART transmission pin.

3. The P45 pin (78K0S/KA1+ and 78K0S/KB1+ microcontrollers) or the P43 pin (78K0S/KY1+ and 78K0S/KU1+ microcontrollers) is used as the UART reception pin.

Cautions 1. Connect the AV_{REF} pin directly to V_{DD} (only for the 78K0S/KA1+ and 78K0S/KB1+ microcontrollers).

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

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

2.2 Peripheral Hardware

The peripheral hardware to be used is shown below.

- **LED**
  An LED is turned on during reception and transmission of data.

CHAPTER 3 SOFTWARE

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

3.1 File Configuration

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

File Name	Description	Compressed (*.zip) File Included		
		ZIP	PM 32	32
main.asm[Note 1]	Source file for hardware initialization processing and main processing of microcontroller	●	●	
op.asm	Assembler source file for setting the option byte (sets the system clock source)	●	●	
softuart.prw	Work space file for integrated development environment PM+		●	
softuart.prj	Project file for integrated development environment PM+		●	
softuart.pri softuart.prs softuart.prm	Project files for system simulator SM+ for 78K0S/Kx1+		●[Note 2]	
softuart0.pnl	I/O panel file for system simulator SM+ for 78K0S/Kx1+ (used for checking peripheral hardware operations)		●[Note 2]	●
softuart0.wvi	Signal data editor file for system simulator SM+ for 78K0S/Kx1+ (used for inputting external signal waveforms)		●[Note 2]	●
softuart0.wvo	Timing chart file for system simulator SM+ for 78K0S/Kx1+ (used for checking waveforms)			●

Notes 1. The software UART sample program is available only in assembly language.

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

Remark ZIP : Only the source file is included.

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

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

3.2 Internal Peripheral Functions to Be Used

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

• $V_{DD} < V_{LVI}$ detection:	Low-voltage detector (LVI)
• UART reception (RxD):	P45 or P43[Note]
• UART transmission (TxD):	P40
• LED output:	P20

Note P45: 78K0S/KA1+ and 78K0S/KB1+ microcontrollers
P43: 78K0S/KY1+ and 78K0S/KU1+ microcontrollers

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 the default baud rate value are performed.

Calibration is performed after completion of the initial settings and the baud rate is determined by receiving a low level (80H) equivalent to 8 bits. Afterward, 8 units of data are received as a reception test, which are then transmitted as a transmission test. Furthermore, an LED is turned on during transmission and reception.

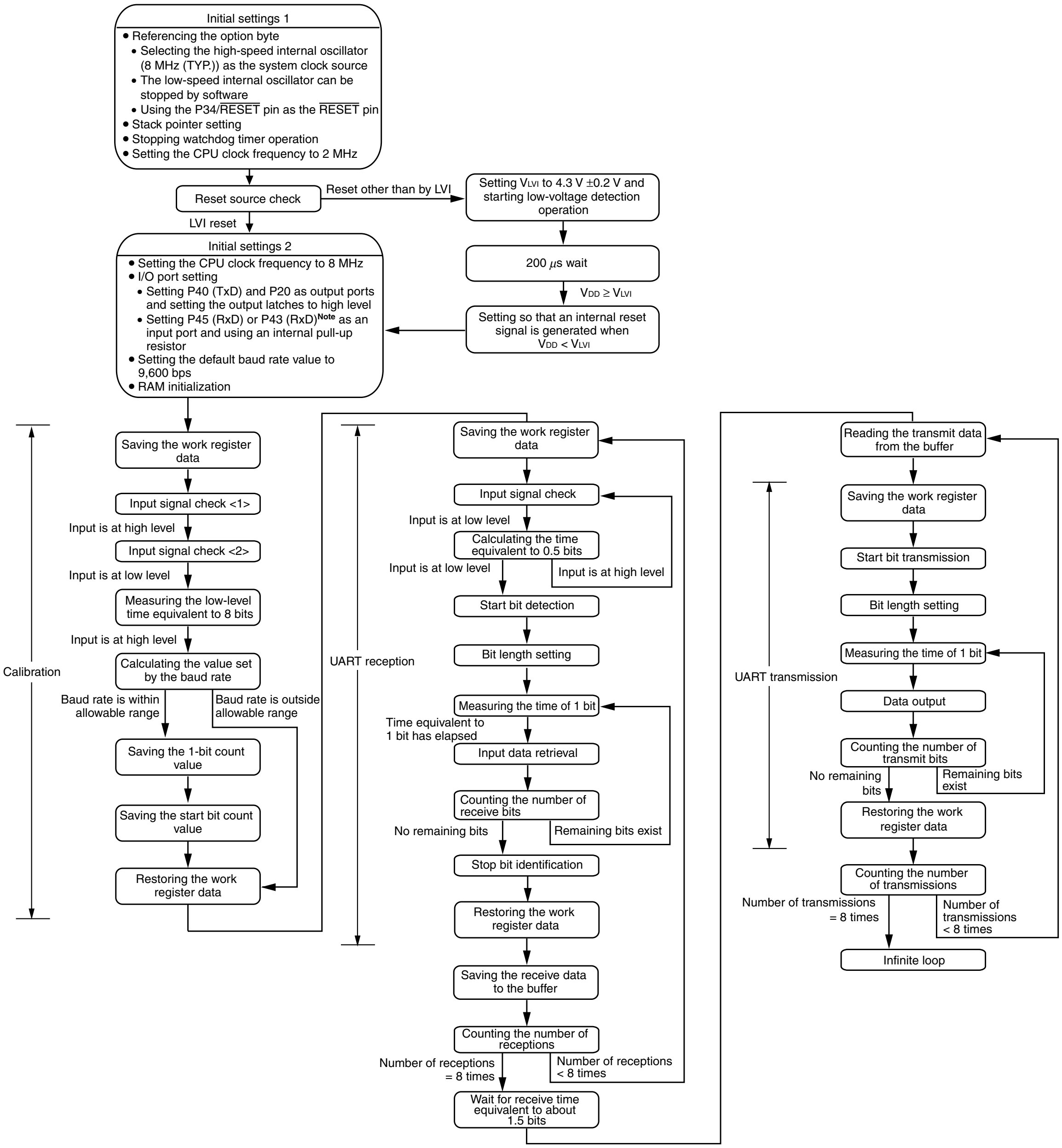
The communication protocol will be set as follows.

- Baud rate: 4,800 to 19,200 bps[Note 1]
- Data character length: 8 bits
- Parity specification: No parity
- Number of stop bits: 1 bit or 2 bits[Note 2]
- Start bit specification: LSB first

Notes 1. The baud rate value is determined by performing calibration. When not performing calibration, the baud rate value can be set by using software. The default value is 9,600 bps.

2. This can be set by using software. The default setting is 1 bit.

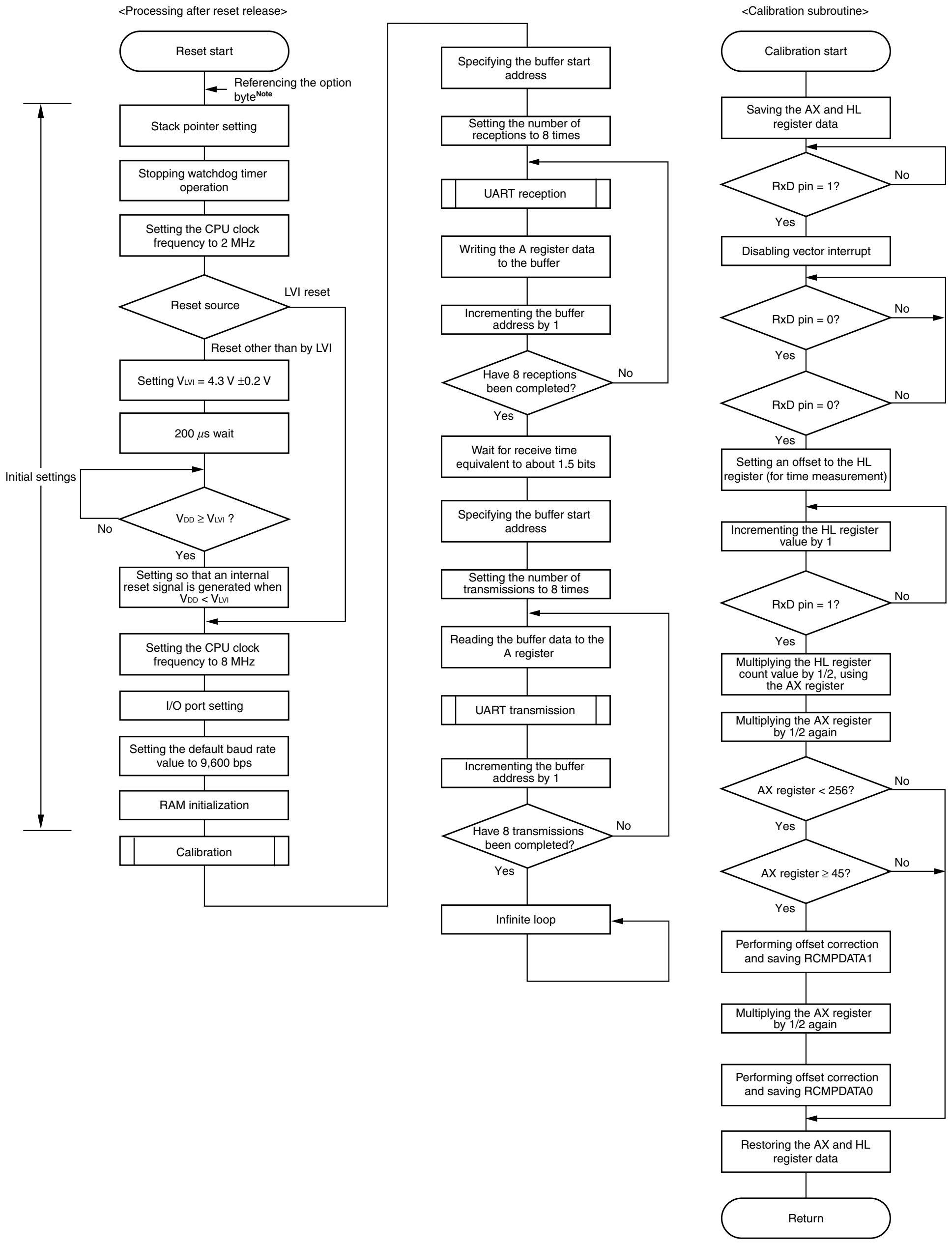
The details are described in the status transition diagram shown below.


Initial settings 1
• Referencing the option byte
• Selecting the high-speed internal oscillator (8 MHz (TYP.)) 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 V_{LVI} to 4.3 V ±0.2 V and starting low-voltage detection operation
200 μs wait
$V_{DD} \geq V_{LVI}$
Setting so that an internal reset signal is generated when $V_{DD} < V_{LVI}$
LVI reset
Initial settings 2
• Setting the CPU clock frequency to 8 MHz
• I/O port setting
• Setting P40 (TxD) and P20 as output ports and setting the output latches to high level
• Setting P45 (RxD) or P43 (RxD)Note as an input port and using an internal pull-up resistor
• Setting the default baud rate value to 9,600 bps
• RAM initialization
Calibration
Saving the work register data
Input signal check <1>
Input is at high level
Input signal check <2>
Input is at low level
Measuring the low-level time equivalent to 8 bits
Input is at high level
Calculating the value set by the baud rate
Baud rate is within allowable range
Baud rate is outside allowable range
Saving the 1-bit count value
Saving the start bit count value
Restoring the work register data
UART reception
Saving the work register data
Input signal check
Input is at low level
Calculating the time equivalent to 0.5 bits
Input is at low level
Input is at high level
Start bit detection
Bit length setting
Measuring the time of 1 bit
Time equivalent to 1 bit has elapsed
Input data retrieval
Counting the number of receive bits
No remaining bits
Remaining bits exist
Stop bit identification
Restoring the work register data
Saving the receive data to the buffer
Counting the number of receptions
Number of receptions = 8 times
Number of receptions < 8 times
Wait for receive time equivalent to about 1.5 bits
UART transmission
Reading the transmit data from the buffer
Saving the work register data
Start bit transmission
Bit length setting
Measuring the time of 1 bit
Data output
Counting the number of transmit bits
No remaining bits
Remaining bits exist
Restoring the work register data
Counting the number of transmissions
Number of transmissions = 8 times
Number of transmissions < 8 times
Infinite loop

Note P45: 78K0S/KA1+ and 78K0S/KB1+ microcontrollers
P43: 78K0S/KY1+ and 78K0S/KU1+ microcontrollers

3.4 Flow Charts

The flow charts for the sample program are shown below.

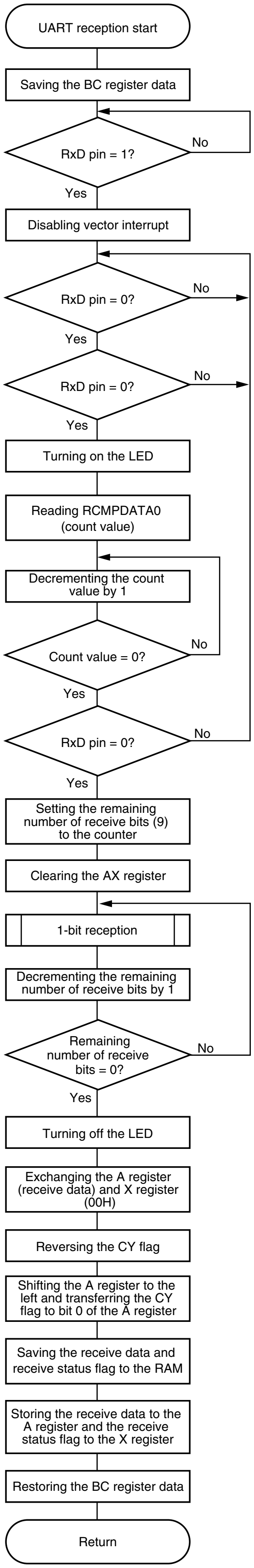

<Processing after reset release>
Reset start
Referencing the option byte[Note]
Initial settings
Stack pointer setting
Stopping watchdog timer operation
Setting the CPU clock frequency to 2 MHz
Reset source
LVI reset
Reset other than by LVI
Setting VLVI = 4.3 V ±0.2 V
200 μs wait
VDD ≥ VLVI ?
No
Yes
Setting so that an internal reset signal is generated when VDD < VLVI
Setting the CPU clock frequency to 8 MHz
I/O port setting
Setting the default baud rate value to 9,600 bps
RAM initialization
Calibration
Specifying the buffer start address
Setting the number of receptions to 8 times
UART reception
Writing the A register data to the buffer
Incrementing the buffer address by 1
Have 8 receptions been completed?
No
Yes
Wait for receive time equivalent to about 1.5 bits
Specifying the buffer start address
Setting the number of transmissions to 8 times
Reading the buffer data to the A register
UART transmission
Incrementing the buffer address by 1
Have 8 transmissions been completed?
No
Yes
Infinite loop
<Calibration subroutine>
Calibration start
Saving the AX and HL register data
RxD pin = 1?
No
Yes
Disabling vector interrupt
RxD pin = 0?
No
Yes
RxD pin = 0?
No
Yes
Setting an offset to the HL register (for time measurement)
Incrementing the HL register value by 1
RxD pin = 1?
No
Yes
Multiplying the HL register count value by 1/2, using the AX register
Multiplying the AX register by 1/2 again
AX register < 256?
No
Yes
AX register ≥ 45?
No
Yes
Performing offset correction and saving RCMPDATA1
Multiplying the AX register by 1/2 again
Performing offset correction and saving RCMPDATA0
Restoring the AX and HL register data
Return

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

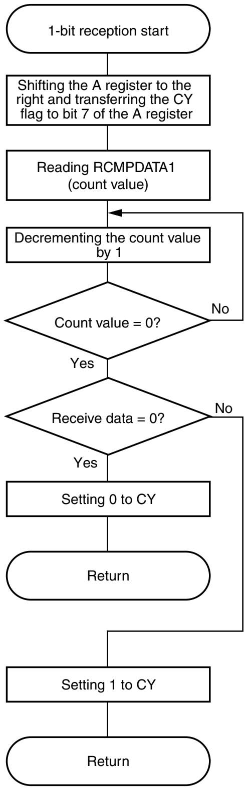
- Using the high-speed internal oscillation clock (8 MHz (TYP.)) as the system clock source
- The low-speed internal oscillator can be stopped by using software
- Using the P34/$\overline{\text{RESET}}$ pin as the $\overline{\text{RESET}}$ pin

Remark The flow charts of <UART receive subroutine>, <1-bit receive subroutine>, <UART transmit subroutine>, and <1-bit transmit subroutine> are shown on the next page.

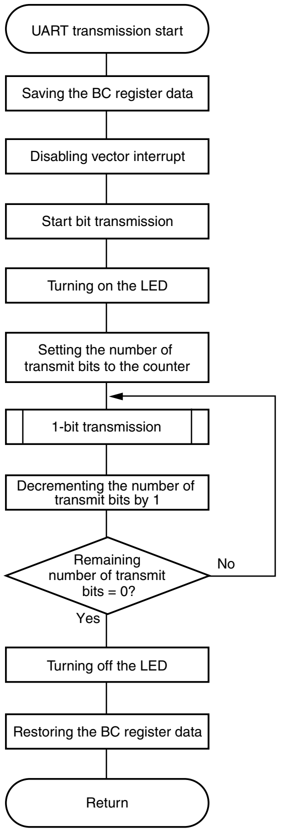
<UART receive subroutine>


UART reception start
Saving the BC register data
RxD pin = 1?
No
Yes
Disabling vector interrupt
RxD pin = 0?
No
Yes
RxD pin = 0?
No
Yes
Turning on the LED
Reading RCMPDATA0 (count value)
Decrementing the count value by 1
Count value = 0?
No
Yes
RxD pin = 0?
No
Yes
Setting the remaining number of receive bits (9) to the counter
Clearing the AX register
1-bit reception
Decrementing the remaining number of receive bits by 1
Remaining number of receive bits = 0?
No
Yes
Turning off the LED
Exchanging the A register (receive data) and X register (00H)
Reversing the CY flag
Shifting the A register to the left and transferring the CY flag to bit 0 of the A register
Saving the receive data and receive status flag to the RAM
Storing the receive data to the A register and the receive status flag to the X register
Restoring the BC register data
Return

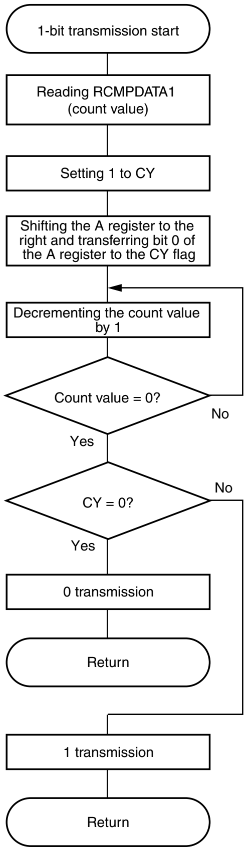
<1-bit receive subroutine>


1-bit reception start
Shifting the A register to the right and transferring the CY flag to bit 7 of the A register
Reading RCMPDATA1 (count value)
Decrementing the count value by 1
Count value = 0?
No
Yes
Receive data = 0?
No
Yes
Setting 0 to CY
Return
Setting 1 to CY
Return

<UART transmit subroutine>


UART transmission start
Saving the BC register data
Disabling vector interrupt
Start bit transmission
Turning on the LED
Setting the number of transmit bits to the counter
1-bit transmission
Decrementing the number of transmit bits by 1
Remaining number of transmit bits = 0?
No
Yes
Turning off the LED
Restoring the BC register data
Return

<1-bit transmit subroutine>


1-bit transmission start
Reading RCMPDATA1 (count value)
Setting 1 to CY
Shifting the A register to the right and transferring bit 0 of the A register to the CY flag
Decrementing the count value by 1
Count value = 0?
No
Yes
CY = 0?
No
Yes
0 transmission
Return
1 transmission
Return

CHAPTER 4 SETTING METHODS

This chapter describes the initial settings of software UART, how to use the calibration, UART receive, and UART transmit subroutines, and an operational overview of the subroutines.

For other initial settings, refer to the **78K0S/Kx1+ Sample Program (Initial Settings) LED Lighting Switch Control 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/KU1+**, **78K0S/KY1+**, **78K0S/KA1+**, **78K0S/KB1+**).

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

4.1 Initial Settings of Software UART

Set the following three items as the initial settings for using software UART communication.

- Ports to be used in software UART communication
- Default baud rate value
- Number of stop bits

4.1.1 Port setting

In this sample program, the pins to be used in software UART communication are set as follows.

- UART reception (RxD): P45 (78K0S/KA1+ and 78K0S/KB1+ microcontrollers)
  P43 (78K0S/KY1+ and 78K0S/KU1+ microcontrollers)
- UART transmission (TxD): P40

In the initial settings after reset release, the following three registers are set as shown in the table below.

- Port register: P4
- Port mode register: PM4
- Pull-up resistor option register: PU4

	P4 Register	PM4 Register	PU4 Register
78K0S/KA1+ and 78K0S/KB1+ microcontrollers	P40 = 1	PM40 = 0, PM45 = 1	PU45 = 1
78K0S/KY1+ and 78K0S/KU1+ microcontrollers	P40 = 1	PM40 = 0, PM43 = 1	PU43 = 1

In the source file, the symbols are defined as follows for frequently used port registers.

[Excerpt from this sample program source (78K0S/KA1+ and 78K0S/KB1+ microcontrollers)]

```
PTXD         EQU    P4.0          ; Pin for UART transmission (TxD pin)
PRXD         EQU    P4.5          ; Pin for UART reception (RxD pin)
```

In software UART communication, general-purpose I/O ports are used to function as the UART receive and transmit pins. Changing the setting of these ports, therefore, enables communication by using arbitrary ports.

4.1.2 Communication protocol setting

The communication protocol is as follows.

- Baud rate: 4,800 to 19,200 bps (9,600 bps by default)
- Data character length: 8 bits
- Parity specification: No parity
- Number of stop bits: 1 bit or 2 bits (1 bit by default)
- Start bit specification: LSB first

The baud rate and number of stop bits can be set by using software.
The baud rate can also be determined by performing calibration.

(1) Setting the baud rate

The following two RAM data are used to set the baud rate.

- RCMPDATA1: For a 1-bit count
- RCMPDATA0: For a start-bit count

In this sample program, the default values of these RAM data are set as follows via the initial settings.

[Excerpt from this sample program source]

```
MOV    RCMPDATA1, #CB9600  ; 1-bit count timer default value (9600 bps)
MOV    RCMPDATA0, #CHB9600 ; Start bit count timer default value (9600 bps)
```

The following three constants are provided for setting the baud rate, by defining the symbols.

Baud Rate	Symbol	
	RCMPDATA1	RCMPDATA0
4,800 bps	CB4800	CHB4800
9,600 bps (default)	CB9600 (default)	CHB9600 (default)
19,200 bps	CB19200	CHB19200

When not performing calibration or when identified as being beyond the recommended baud rate range during calibration, the RAM data that have been set via the initial settings will be applied.

(2) Setting the number of stop bits

In this sample program, the number of stop bits is set to 1 by default, by defining the symbols.

[Excerpt from this sample program source]

```
CSTOPBIT     EQU    1             ; Specify the number of stop bits
```

The number of stop bits during transmission can be set to 2 by changing “1”, shown above, to “2”.
Software UART operates with the number of stop bits always set to “1” during reception.

4.2 Calibration

4.2.1 How to use calibration

In this sample program, calibration processing is turned into a subroutine. Calibration can be executed by calling as follows.

[Example of calling a calibration subroutine]

```
CALL   !SCALIB
```

This subroutine can be used to perform calibration as many times as desired to correct the baud rate.

The following two setting values are stored to the RAM via calibration processing. These RAM data vary, depending on the baud rate.

- RCMPDATA1: For a 1-bit count
- RCMPDATA0: For a start-bit count

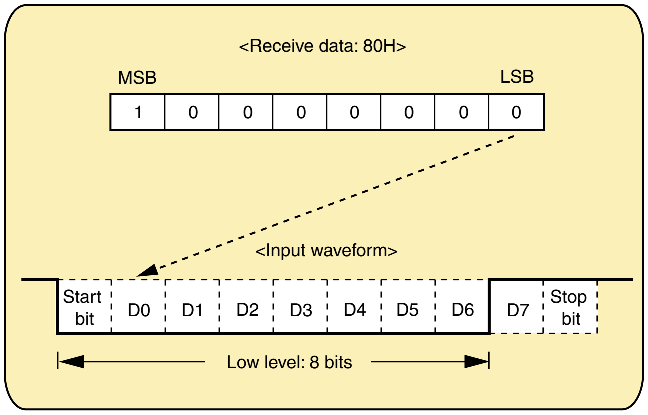
When the baud rate is significantly outside the recommended range during calibration, these RAM data will not be updated and the data with the baud rate immediately before will be retained. (In this sample program, these are the initial setting values (the values when the baud rate is 9,600 bps).) For details, refer to **4.2.2 Operational overview of calibration**.

4.2.2 Operational overview of calibration

Calibration is performed and the baud rate is determined by receiving a low level equivalent to 8 bits (receive data: 80H) from the pin (RxD) to be used as the input for serial communication.

In this sample program, calibration is performed immediately after completion of the initial settings.

The waveform of the receive data of 80H for calibration detection is shown below. The low-level width that is equivalent to 8 bits is the length from the start bit to bit 6 of the character bit.



Before starting to measure the low-level width, disable interrupt acknowledgment and set so that no interrupt servicing occurs during the measurement.

When starting to measure the low-level width, check twice that the RxD pin is at low level, in order to eliminate noise less than about 1.5 μs. After checking, start counting the HL register for measurement after setting to the HL register the correction value that has been derived from the number of execution clocks of the instruction, so that a processing time that suits the communication speed is achieved.

Increment the HL register for measuring the time by 1 every 16 clocks, based on the number of execution clocks of the instruction.

When the RxD pin has been checked to be at high level, counting the HL register ends.

[Excerpt from this sample program source (calibration)]

```
JCAL000:
        BF      PRXD,  $JCAL000      ;  If  the  RxD  pin  is  0,  wait  until  it
becomes 1
        DI                           ;  Disable vector interrupt
JCAL100:
        BT      PRXD,  $JCAL100      ;10: Wait for calibration start
        BT      PRXD,  $JCAL100      ;10: If  noise  is  present,  return  to
waiting for calibration start

        NOP                          ; 2: For time adjustment
        NOP                          ; 2: For time adjustment
        MOVW    HL,    #CCALOFFSET   ; 6: Time correction
JCAL200:
        NOP                          ; 2:
        INCW    HL                   ; 4: Time measurement
        BF      PRXD,  $JCAL200      ;10: Wait for RxD pin to become 1
```

Disabling interrupt before starting measurement

Detecting a low level twice

2 + 4 + 10 = 16 clocks

Setting the correction value

Incrementing the HL register by 1 every 16 clocks, because the NOP to BF instructions are repeated until a high level is detected

Incrementing the HL register by 1

Waiting for high-level detection

The relation between the HL register count value and the baud rate value is shown below.

The clock frequency to be used is 8 MHz; therefore, 8 MHz = 8 × 10^6 Hz.

$$\text{HL register count value} = \underbrace{\underbrace{\underbrace{\frac{1}{\text{Baud rate value}}}_{\text{(a)}} \times \boxed{8 \times 10^6}}_{\text{(b)}} \times 8}_{\text{(c)}} \times \boxed{\frac{1}{16}}$$

The HL register is incremented every 16 clocks, based on the number of execution clocks of the instruction; therefore, “Number of clocks equivalent to 8 bits × 1/16” becomes the count value.

(a): Time equivalent to 1 bit (s)
(b): Number of clocks equivalent to 1 bit
(c): Number of clocks equivalent to 8 bits

$$\rightarrow \text{Baud rate value} = \frac{4 \times 10^6}{\text{HL register count value}}$$

[Example] When the HL register count value is 416

$$\text{Baud rate value} = \frac{4 \times 10^6}{416} \cong 9{,}615 \text{ [bps]}$$

Determine from the value that is a fourth of the HL register count value whether the baud rate is within the recommended range. At this time, the allowable range of the baud rate will be as follows, based on the identified value of the count value.

$$45 \leq \frac{\text{HL register count value}}{4} < 256$$

→ Allowable range of the baud rate: 3,907 to 22,222 [bps]

(Recommended range of the baud rate: 4,800 to 19,200 [bps])

When the baud rate is within the allowable range, determine the values to be saved from the HL register count value to RCMPDATA1 and RCMPDATA0.

Remark The values of when the baud rate is 9,600 bps are set to RCMPDATA1 and RCMPDATA0 by default. When not performing calibration or when the baud rate is outside the allowable range as a result of calibration, the default values will be used for UART transmission and reception.

This is because the number of execution clocks of the instruction to be used when counting the HL register is 16 (incremented by 1 every 16 clocks and measured for 8 bits).

<1> Count value equivalent to 1 bit (RCMPDATA1)

$$= \underbrace{\underbrace{\text{HL register count value} \times \boxed{16}}_{(a)} \times \frac{1}{8}}_{(b)} \times \boxed{\frac{1}{8}} - \boxed{\text{Correction value}}$$

Correction of the processing time for counting for 1 bit

$$= \text{HL register value} \times \frac{1}{4} - \text{Correction value}$$

This is because the number of the execution clocks of the instruction to be used when counting RCMPDATA1 is 8. (The count value is decremented by 1 every 8 clocks.)

(a): Number of clocks equivalent to 8 bits

(b): Number of clocks equivalent to 1 bit

Remark The count value that is equivalent to 1 bit (RCMPDATA1) is used to determine the identification timing of the 1-bit data during UART reception and to determine the length of the 1-bit data during UART transmission.

<2>Count value for start-bit detection (RCMPDATA0)

This is because the number of the execution clocks of the instruction to be used when counting the HL register is 16 (incremented by 1 every 16 clocks and measured for 8 bits).

Correction of the processing time for start-bit detection

$$= \underbrace{\underbrace{\underbrace{\text{HL register value} \times \boxed{16}}_{(a)} \times \frac{1}{8}}_{(b)} \times \frac{1}{2}}_{(c)} \times \boxed{\frac{1}{8}} - \boxed{\text{Correction value}} = \text{HL register value} \times \frac{1}{8} - \text{Correction value}$$

This is because the number of the execution clocks of the instruction to be used when counting RCMPDATA0 is 8. (The count value is decremented by 1 every 8 clocks.)

(a): Number of clocks equivalent to 8 bits

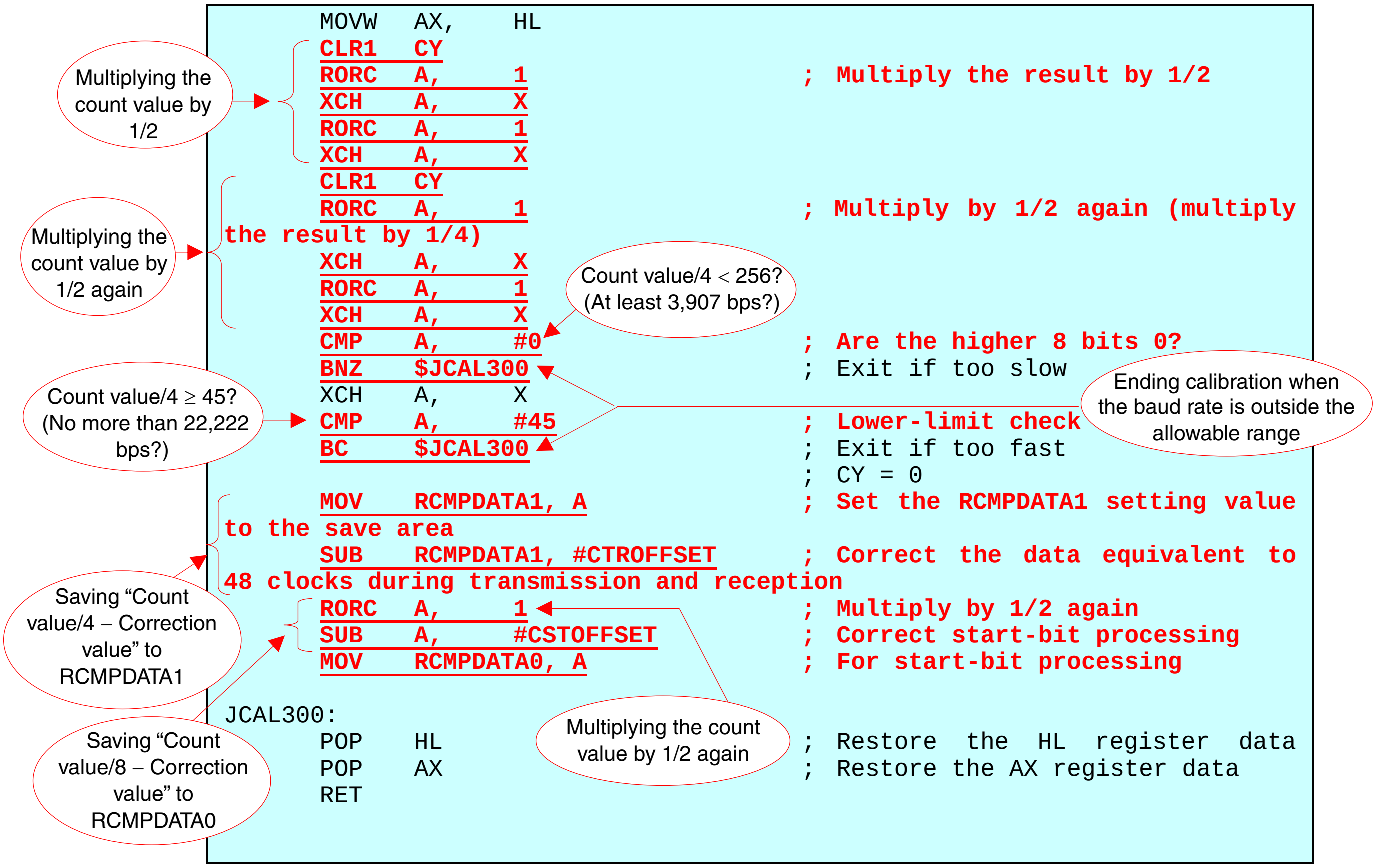
(b): Number of clocks equivalent to 1 bit

(c): Number of clocks equivalent to 0.5 bits

Remarks 1. The count value for start-bit detection (RCMPDATA0) is used to detect the start bit during UART reception. RCMPDATA0 is determined so that the start bit is detected at the point of 0.5 bits after the UART receive pin (RxD) is detected to be at low level, because 0 or 1 is identified at the center of the 1-bit data in UART reception.

2. The correction value of RCMPDATA1 and that of RCMPDATA0 are different values.

[Excerpt from this sample program source (from calibration result identification to saving of the count value)]



```
      MOVW  AX,   HL
      CLR1  CY
      RORC  A,    1                  ; Multiply the result by 1/2
      XCH   A,    X
      RORC  A,    1
      XCH   A,    X
      CLR1  CY
      RORC  A,    1                  ; Multiply by 1/2 again (multiply
the result by 1/4)
      XCH   A,    X
      RORC  A,    1
      XCH   A,    X
      CMP   A,    #0                 ; Are the higher 8 bits 0?
      BNZ   $JCAL300                 ; Exit if too slow
      XCH   A,    X
      CMP   A,    #45                ; Lower-limit check
      BC    $JCAL300                 ; Exit if too fast
                                     ; CY = 0
      MOV   RCMPDATA1, A             ; Set the RCMPDATA1 setting value
to the save area
      SUB   RCMPDATA1, #CTROFFSET    ; Correct the data equivalent to
48 clocks during transmission and reception
      RORC  A,    1                  ; Multiply by 1/2 again
      SUB   A,    #CSTOFFSET         ; Correct start-bit processing
      MOV   RCMPDATA0, A             ; For start-bit processing
JCAL300:
      POP   HL                       ; Restore the HL register data
      POP   AX                       ; Restore the AX register data
      RET
```

Multiplying the count value by 1/2
Multiplying the count value by 1/2 again
Count value/4 < 256? (At least 3,907 bps?)
Count value/4 ≥ 45? (No more than 22,222 bps?)
Ending calibration when the baud rate is outside the allowable range
Saving "Count value/4 – Correction value" to RCMPDATA1
Saving "Count value/8 – Correction value" to RCMPDATA0
Multiplying the count value by 1/2 again

4.3 UART Reception

4.3.1 How to use UART reception

In this sample program, UART receive processing is turned into a subroutine. Data receive processing can be performed, and the receive data and receive status flag can be saved by using UART receive processing, as follows.

[Example of calling the UART receive subroutine and saving the data]

```
CALLT  [ZRXDATA]             ; UART receive subroutine call
MOV    RDATA, A              ; Save the receive data to RDATA
XCH    A,     X              ; Transfer the receive status flag to the A register
MOV    SDATA, A              ; Save the receive status flag to SDATA
```

With this UART receive subroutine call, the receive data and receive status flag are stored to both a general-purpose register and a RAM area and processing is returned from the subroutine. The correspondences between the receive data and receive status flag, and the general-purpose register and RAM area are shown below.

	General-Purpose Register to Which to Be Saved	RAM Address to Which to Be Saved
Receive data	A register	RRXDATA (Lower 1 byte of RRXDATA)
Receive status flag	X register	RRXFLAG (Higher 1 byte of RRXDATA)

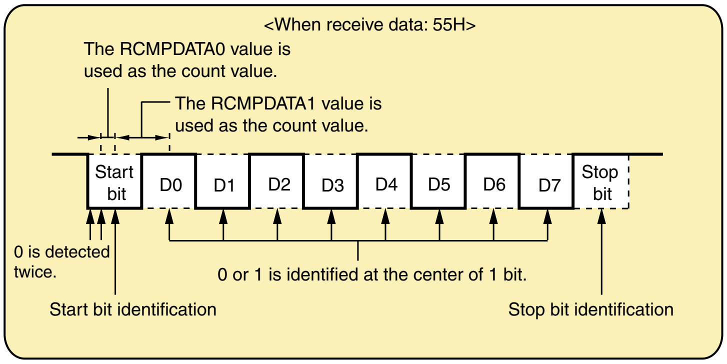
The receive status flag can only identify framing errors (stop bit is not detected).

- Receive status flag = 00H: Normal reception
- Receive status flag = 01H: Framing error

4.3.2 Operational overview of UART reception

Before starting data reception, disable interrupt acknowledgment and set so that no interrupt servicing occurs during reception.

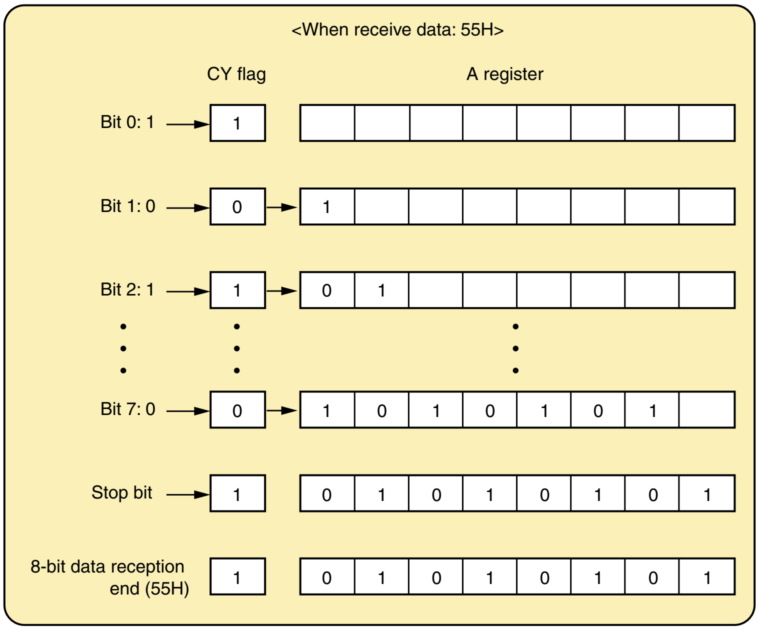
When starting data reception, check twice that the UART receive pin (RxD) pin is at low level, in order to eliminate noise less than about 1.5 μs. After checking, the start bit is identified at the center of the bit.


<When receive data: 55H>
The RCMPDATA0 value is used as the count value.
The RCMPDATA1 value is used as the count value.
Start bit
D0
D1
D2
D3
D4
D5
D6
D7
Stop bit
0 is detected twice.
0 or 1 is identified at the center of 1 bit.
Start bit identification
Stop bit identification

The identification timings are determined by using the **RCMPDATA0** value (count value equivalent to 0.5 bits) that was determined via calibration when identifying the start bit, and using the **RCMPDATA1** value (count value equivalent to 1 bit) when identifying the bits following bit 0, to identify the timing of the bits at the center of the bits.

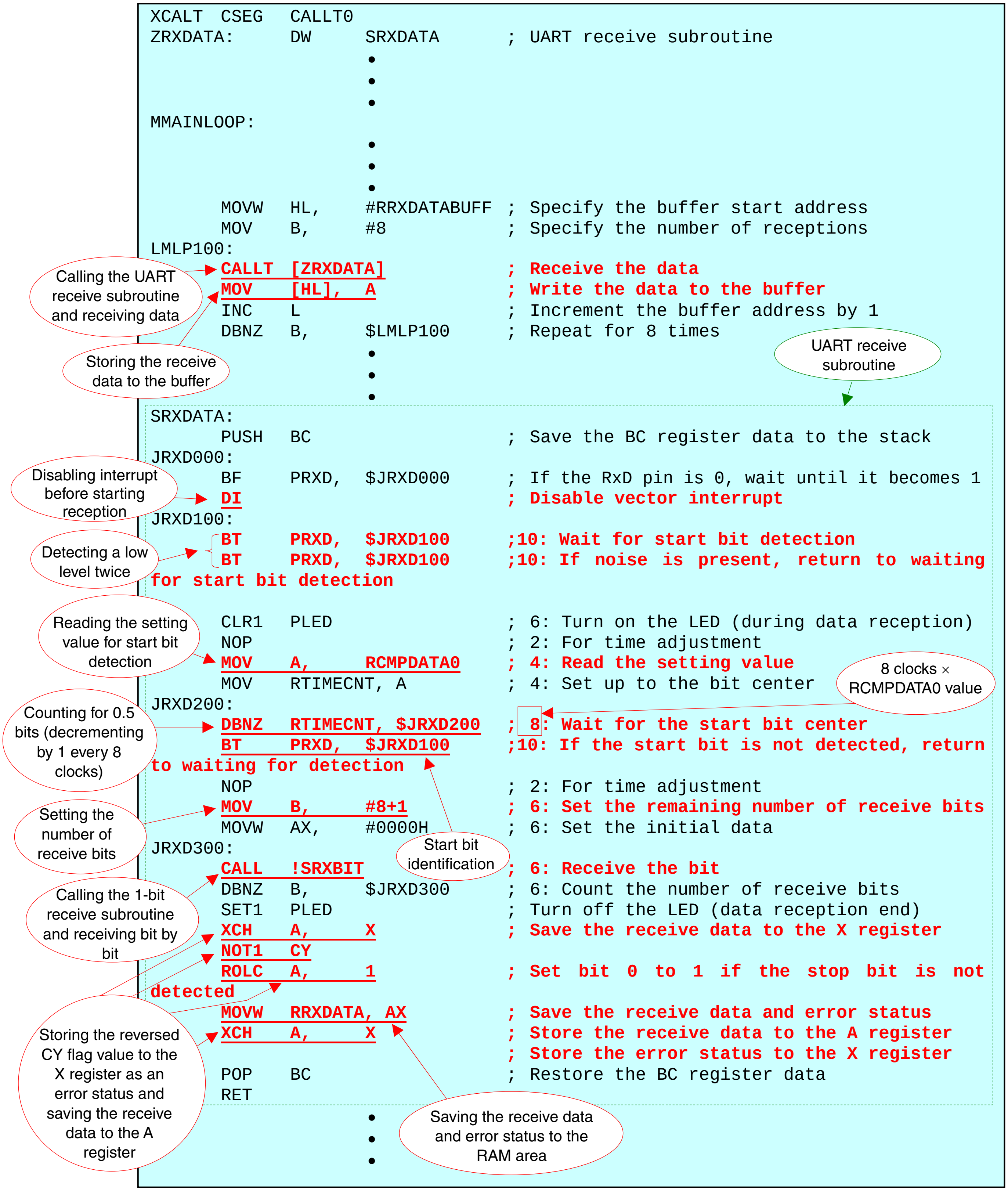
Remark The values of when the baud rate is 9,600 bps are set to RCMPDATA1 and RCMPDATA0 by default. When not performing calibration or when the baud rate is outside the allowable range as a result of calibration, the default values will be used for UART transmission and reception.

When receiving data, the A register and CY flag are used as the buffer registers for reception. The CY flag data (receive bit) will be shifted to bit 7 of the A register by setting to the CY flag the same value as the receive data every time when receiving 1 bit and shifting the A register to the right.



The receive data is saved to the A register and RRXDATA (RAM area) after 8-bit data reception ends. Furthermore, the CY flag value is reversed and saved to the X register and RRXFLAG (RAM area) as the receive status flag (0: normal reception, 1: framing error).

[Excerpt from this sample program source (UART reception <1>)]

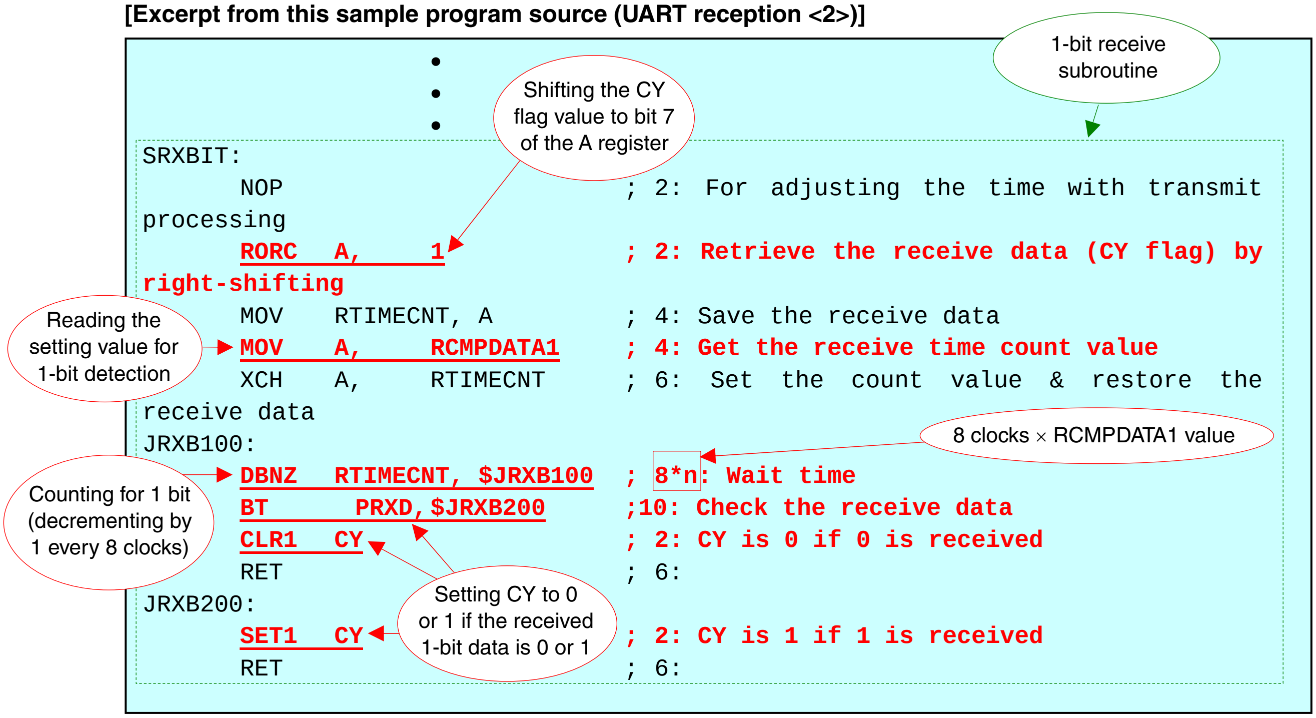


```
XCALT  CSEG   CALLT0
ZRXDATA:      DW     SRXDATA       ; UART receive subroutine
                     •
                     •
                     •
MMAINLOOP:
                     •
                     •
                     •
       MOVW   HL,    #RRXDATABUFF  ; Specify the buffer start address
       MOV    B,     #8            ; Specify the number of receptions
LMLP100:
       CALLT  [ZRXDATA]            ; Receive the data
       MOV    [HL],  A             ; Write the data to the buffer
       INC    L                    ; Increment the buffer address by 1
       DBNZ   B,     $LMLP100      ; Repeat for 8 times
                     •
                     •
                     •
SRXDATA:
       PUSH   BC                   ; Save the BC register data to the stack
JRXD000:
       BF     PRXD,  $JRXD000      ; If the RxD pin is 0, wait until it becomes 1
       DI                          ; Disable vector interrupt
JRXD100:
       BT     PRXD,  $JRXD100      ;10: Wait for start bit detection
       BT     PRXD,  $JRXD100      ;10: If noise is present, return to waiting
for start bit detection

       CLR1   PLED                 ; 6: Turn on the LED (during data reception)
       NOP                         ; 2: For time adjustment
       MOV    A,     RCMPDATA0     ; 4: Read the setting value
       MOV    RTIMECNT, A          ; 4: Set up to the bit center
JRXD200:
       DBNZ   RTIMECNT, $JRXD200   ; 8: Wait for the start bit center
       BT     PRXD,  $JRXD100      ;10: If the start bit is not detected, return
to waiting for detection
       NOP                         ; 2: For time adjustment
       MOV    B,     #8+1          ; 6: Set the remaining number of receive bits
       MOVW   AX,    #0000H        ; 6: Set the initial data
JRXD300:
       CALL   !SRXBIT              ; 6: Receive the bit
       DBNZ   B,     $JRXD300      ; 6: Count the number of receive bits
       SET1   PLED                 ; Turn off the LED (data reception end)
       XCH    A,     X             ; Save the receive data to the X register
       NOT1   CY
       ROLC   A,     1             ; Set bit 0 to 1 if the stop bit is not
detected
       MOVW   RRXDATA, AX          ; Save the receive data and error status
       XCH    A,     X             ; Store the receive data to the A register
                                   ; Store the error status to the X register
       POP    BC                   ; Restore the BC register data
       RET
                     •
                     •
                     •
```

Calling the UART receive subroutine and receiving data
Storing the receive data to the buffer
UART receive subroutine
Disabling interrupt before starting reception
Detecting a low level twice
Reading the setting value for start bit detection
8 clocks × RCMPDATA0 value
Counting for 0.5 bits (decrementing by 1 every 8 clocks)
Setting the number of receive bits
Start bit identification
Calling the 1-bit receive subroutine and receiving bit by bit
Storing the reversed CY flag value to the X register as an error status and saving the receive data to the A register
Saving the receive data and error status to the RAM area

Remark The 1-bit receive subroutine (SRXBIT) is continued on the next page.

[Excerpt from this sample program source (UART reception <2>)]



```
                    •
                    •
                    •
SRXBIT:
        NOP                             ; 2:  For  adjusting  the  time  with  transmit
processing
        RORC    A,      1               ; 2: Retrieve the receive data (CY flag) by
right-shifting
        MOV     RTIMECNT, A             ; 4: Save the receive data
        MOV     A,      RCMPDATA1       ; 4: Get the receive time count value
        XCH     A,      RTIMECNT        ; 6:  Set  the  count  value  &  restore  the
receive data
JRXB100:
        DBNZ    RTIMECNT, $JRXB100      ; 8*n: Wait time
        BT       PRXD, $JRXB200         ;10: Check the receive data
        CLR1    CY                      ; 2: CY is 0 if 0 is received
        RET                             ; 6:
JRXB200:
        SET1    CY                      ; 2: CY is 1 if 1 is received
        RET                             ; 6:
```

1-bit receive subroutine
Shifting the CY flag value to bit 7 of the A register
Reading the setting value for 1-bit detection
8 clocks × RCMPDATA1 value
Counting for 1 bit (decrementing by 1 every 8 clocks)
Setting CY to 0 or 1 if the received 1-bit data is 0 or 1

Remark This excerpt from the sample program source is continued from the previous page.

4.4 UART Transmission

4.4.1 How to use UART transmission

In this sample program, UART transmit processing is turned into a subroutine. Data transmit processing can be performed by using UART transmit processing as follows.

[Example of calling the UART transmit subroutine]

```
MOV    A,     TDATA   ; Store the data to be transmitted (TDATA) to the A register
CALLT  [ZTXDATA]      ; Transmit the data
```

4.4.2 Operational overview of UART transmission

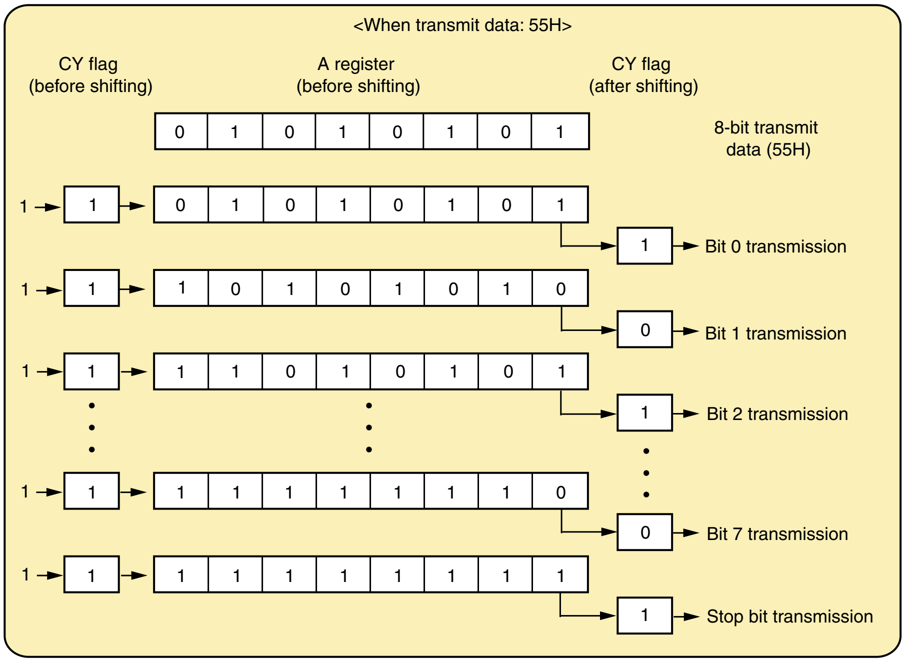
Before starting data transmission, disable interrupt acknowledgment and set so that no interrupt servicing occurs during transmission.

The transmit data must be stored to the A register before calling the subroutine, because the transmit data is read from the A register.

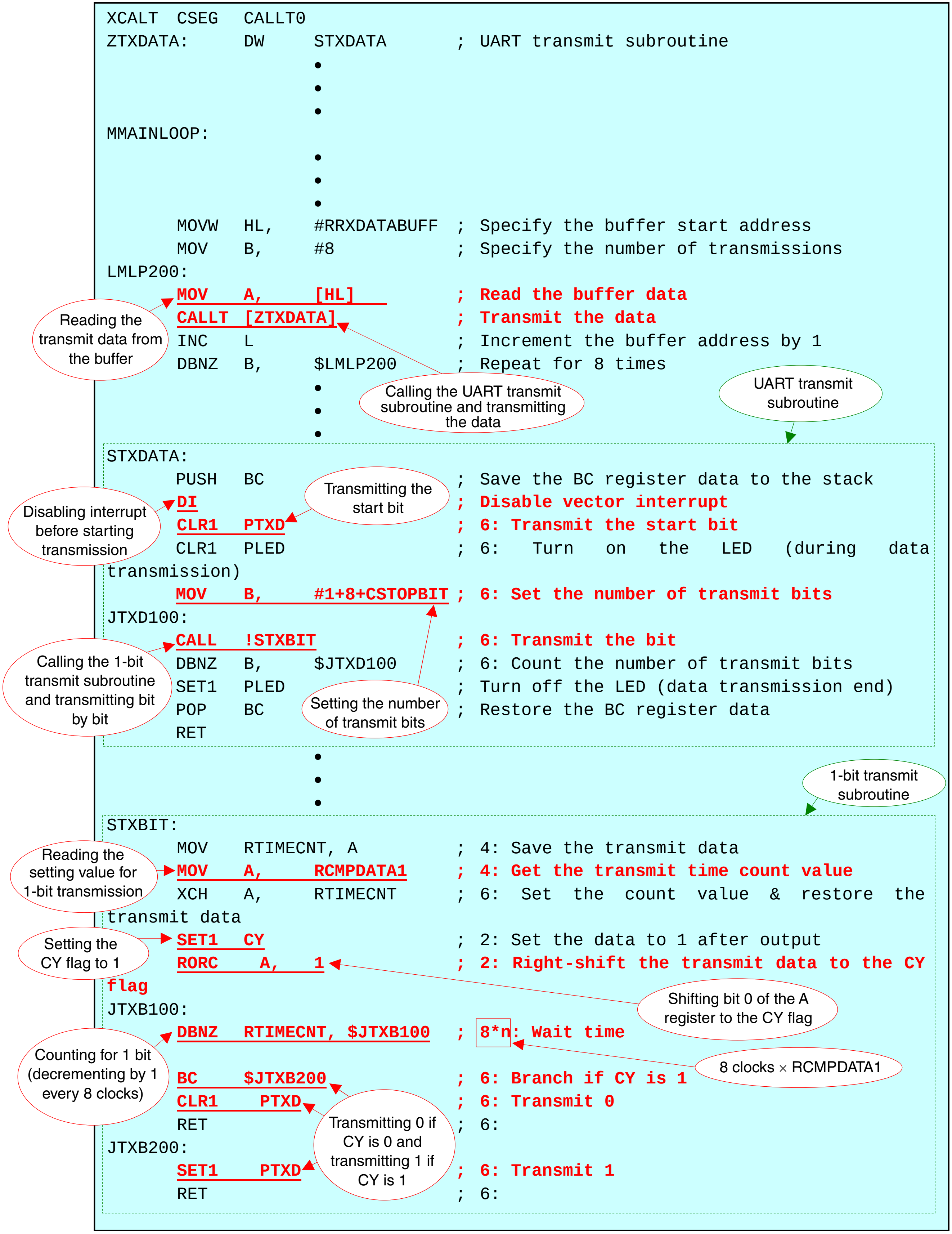
When transmitting the data, use the RCMPDATA1 value (count value equivalent to 1 bit) that was determined via calibration, in order to determine the length of 1 bit to be transmitted.

Remark The values of when the baud rate is 9,600 bps are set to RCMPDATA1 by default. When not performing calibration or when the baud rate is outside the allowable range as a result of calibration, the default value will be used for UART transmission and reception.

When transmitting data, the A register and CY flag are used as the buffer registers for transmission. The CY flag data will be shifted to bit 7 of the A register, the bit 0 data (transmit bit) of the A register will be shifted to the CY flag, and the same value as that of the CY flag after it has been shifted will be transmitted every bit, by setting 1 to the CY flag and shifting the data to the right.


<When transmit data: 55H>
CY flag (before shifting)
A register (before shifting)
CY flag (after shifting)
0 1 0 1 0 1 0 1
8-bit transmit data (55H)
1 → 1 → 0 1 0 1 0 1 0 1 → 1 → Bit 0 transmission
1 → 1 → 1 0 1 0 1 0 1 0 → 0 → Bit 1 transmission
1 → 1 → 1 1 0 1 0 1 0 1 → 1 → Bit 2 transmission
1 → 1 → 1 1 1 1 1 1 1 0 → 0 → Bit 7 transmission
1 → 1 → 1 1 1 1 1 1 1 1 → 1 → Stop bit transmission

[Excerpt from this sample program source (UART transmission)]



```
XCALT  CSEG   CALLT0
ZTXDATA:      DW     STXDATA       ; UART transmit subroutine
                     •
                     •
                     •
MMAINLOOP:
                     •
                     •
                     •
       MOVW   HL,    #RRXDATABUFF  ; Specify the buffer start address
       MOV    B,     #8            ; Specify the number of transmissions
LMLP200:
       MOV    A,     [HL]          ; Read the buffer data
       CALLT  [ZTXDATA]            ; Transmit the data
       INC    L                    ; Increment the buffer address by 1
       DBNZ   B,     $LMLP200      ; Repeat for 8 times
                     •
                     •
                     •
STXDATA:
       PUSH   BC                   ; Save the BC register data to the stack
       DI                          ; Disable vector interrupt
       CLR1   PTXD                 ; 6: Transmit the start bit
       CLR1   PLED                 ; 6:   Turn   on   the   LED   (during   data
transmission)
       MOV    B,     #1+8+CSTOPBIT ; 6: Set the number of transmit bits
JTXD100:
       CALL   !STXBIT              ; 6: Transmit the bit
       DBNZ   B,     $JTXD100      ; 6: Count the number of transmit bits
       SET1   PLED                 ; Turn off the LED (data transmission end)
       POP    BC                   ; Restore the BC register data
       RET
                     •
                     •
                     •
STXBIT:
       MOV    RTIMECNT, A          ; 4: Save the transmit data
       MOV    A,     RCMPDATA1     ; 4: Get the transmit time count value
       XCH    A,     RTIMECNT      ; 6:  Set  the  count  value  &  restore  the
transmit data
       SET1   CY                   ; 2: Set the data to 1 after output
       RORC      A,   1            ; 2: Right-shift the transmit data to the CY
flag
JTXB100:
       DBNZ   RTIMECNT, $JTXB100   ; 8*n: Wait time

       BC     $JTXB200             ; 6: Branch if CY is 1
       CLR1      PTXD              ; 6: Transmit 0
       RET                         ; 6:
JTXB200:
       SET1      PTXD              ; 6: Transmit 1
       RET                         ; 6:
```

Reading the transmit data from the buffer
Calling the UART transmit subroutine and transmitting the data
UART transmit subroutine
Transmitting the start bit
Disabling interrupt before starting transmission
Calling the 1-bit transmit subroutine and transmitting bit by bit
Setting the number of transmit bits
1-bit transmit subroutine
Reading the setting value for 1-bit transmission
Setting the CY flag to 1
Shifting bit 0 of the A register to the CY flag
8 clocks × RCMPDATA1
Counting for 1 bit (decrementing by 1 every 8 clocks)
Transmitting 0 if CY is 0 and transmitting 1 if CY is 1

CHAPTER 5 OPERATION CHECK USING SYSTEM SIMULATOR SM+

This chapter describes how the sample program operates with system simulator SM+ for 78K0S/Kx1+, by using the assembly language file (source files + project file) that has been downloaded by selecting the PM32 icon.

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

<R> 5.1 Building the Sample Program

To check the operation of the sample program by using system simulator SM+ for 78K0S/Kx1+ (hereinafter referred to as “SM+”), SM+ must be started after building the sample program. This section describes how to build a sample program by using the assembly language sample program (source program + project file) downloaded by clicking the PM32 icon. See the 78K0S/Kx1+ Sample Program Startup Guide Application Note for how to build other downloaded programs.

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

[Column] Build errors

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

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

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

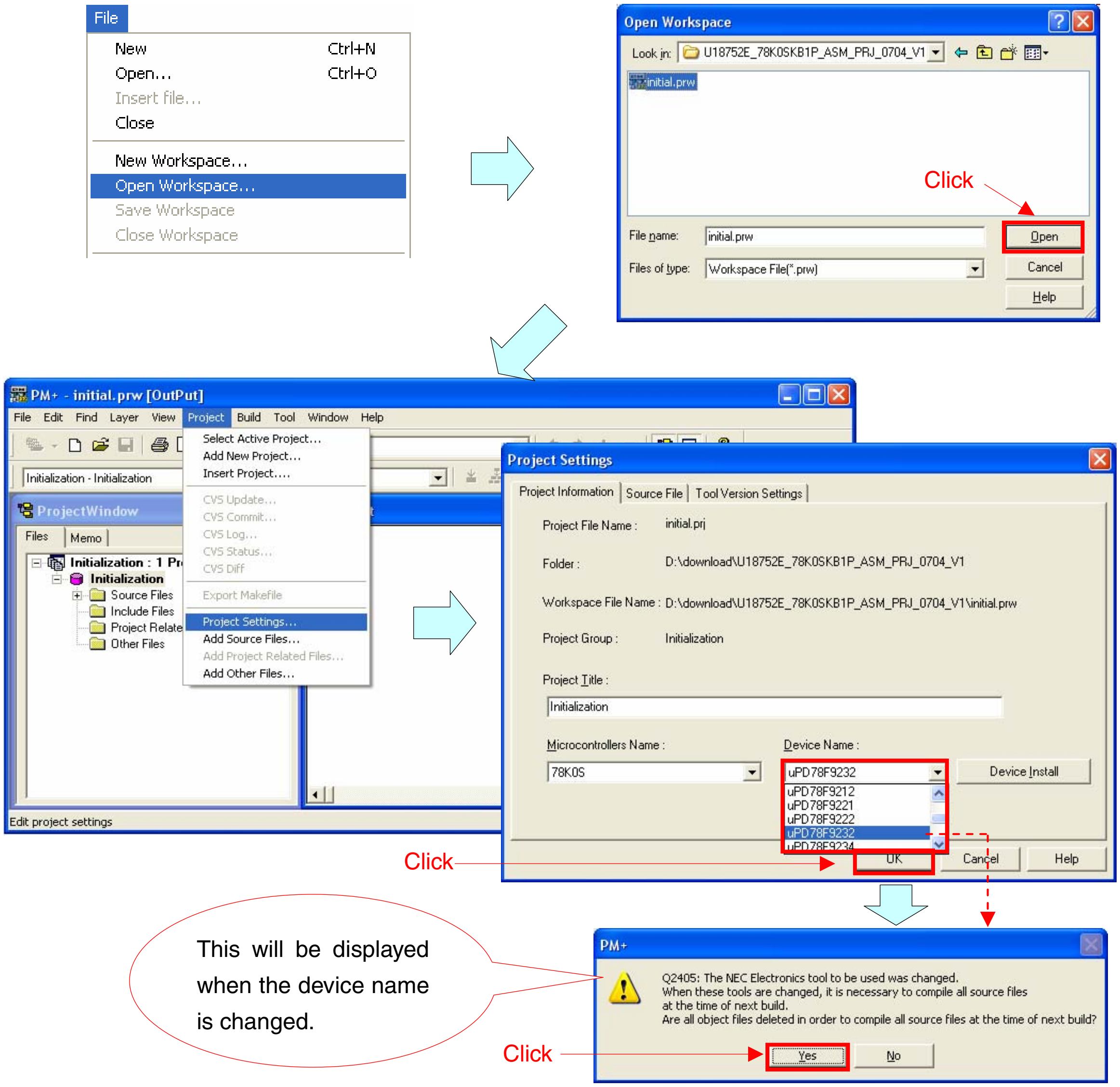
The [Using Fixed Area of Standard Library] check box is unchecked by default when the file that has been downloaded by clicking the PM32 icon is used in this sample program.

(1) Start PM+.

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

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

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



File name: initial.prw

Files of type: Workspace File(*.prw)

Project File Name : initial.prj

Folder : D:\download\U18752E_78K0SKB1P_ASM_PRJ_0704_V1

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

Project Group : Initialization

Project Title : Initialization

Microcontrollers Name : 78K0S

Device Name : uPD78F9232

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

Click

Click

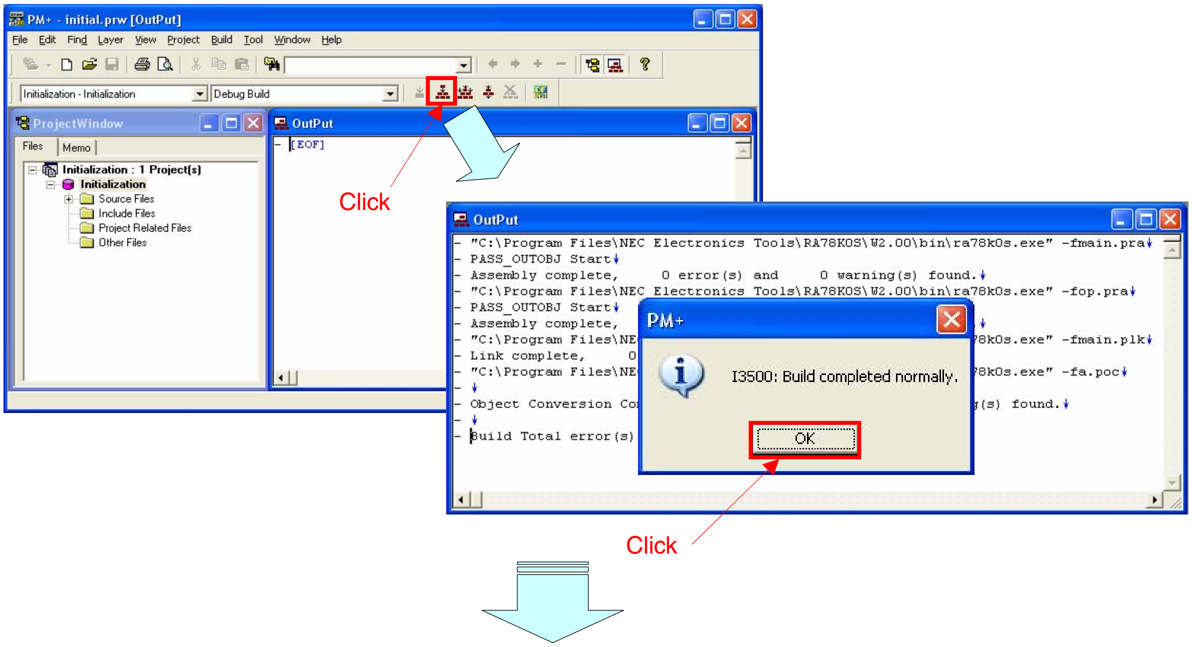
This will be displayed when the device name is changed.

Click

(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                                 (s) found.↓
- ↓
- Build Total error(s)
```

I3500: Build completed normally.

A HEX file for flash memory writing will be generated.

5.2 Operation with SM+

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

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

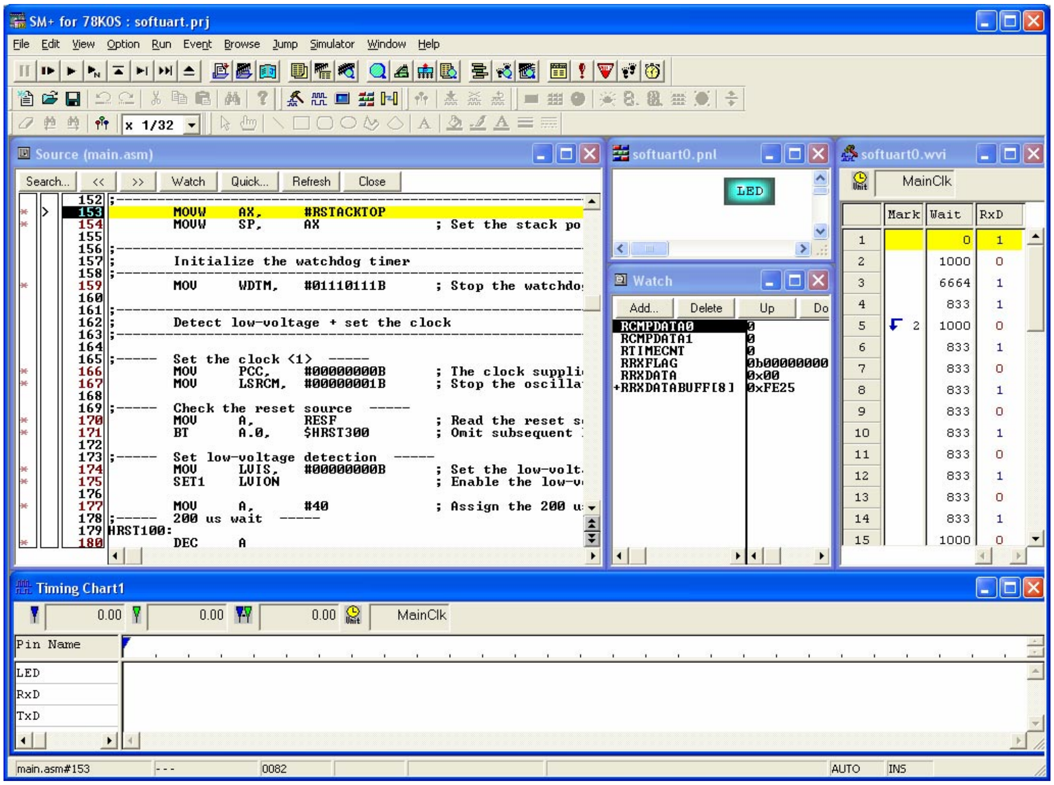
(a) When starting SM+ in PM+

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

(b) When not starting SM+ in PM+

•Start SM+ from the Windows start menu.

(2) When SM+ is started, the following screen will be displayed.


SM+ for 78K0S : softuart.prj
File Edit View Option Run Event Browse Jump Simulator Window Help
x 1/32
Source (main.asm)
Search... << >> Watch Quick... Refresh Close
softuart0.pnl
LED
Watch
Add... Delete Up Do
softuart0.wvi
MainClk
Timing Chart1
0.00 0.00 0.00 MainClk
main.asm#153 --- 0082 AUTO INS

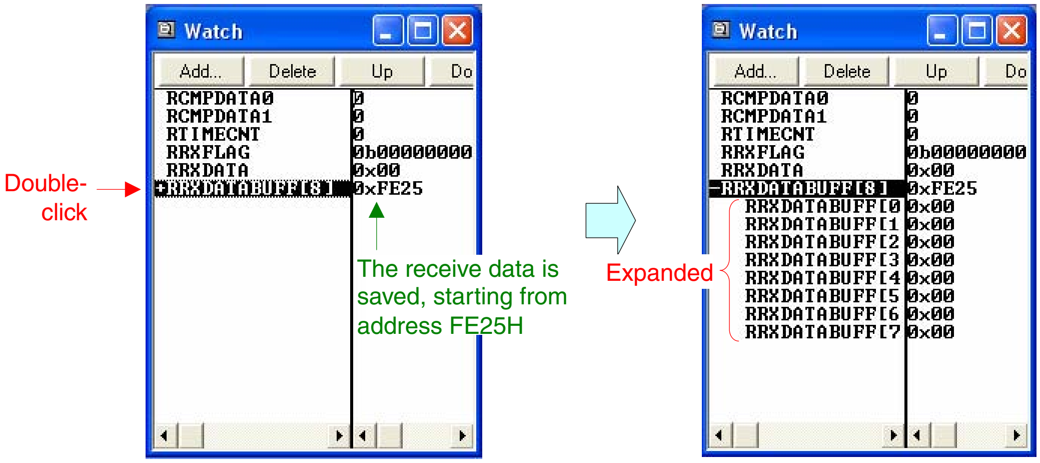
```
;-------------------------------------------------------------
        MOVW    AX,     #RSTACKTOP
        MOVW    SP,     AX                  ; Set the stack po

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

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

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

;-----  Check the reset source  -----
        MOV     A,      RESF                ; Read the reset s
        BT      A.0,    $HRST300            ; Omit subsequent 

;-----  Set low-voltage detection  -----
        MOV     LVIS,   #00000000B          ; Set the low-volt
        SET1    LVION                       ; Enable the low-v

        MOV     A,      #40                 ; Assign the 200 u
;-----  200 us wait  -----
HRST100:
        DEC     A
```

Watch	
RCMPDATA0	0
RCMPDATA1	0
RTIMECNT	0
RRXFLAG	0b00000000
RRXDATA	0x00
+RRXDATABUFF[8]	0xFE25

	Mark	Wait	RxD
1		0	1
2		1000	0
3		6664	1
4		833	1
5	2	1000	0
6		833	1
7		833	0
8		833	1
9		833	0
10		833	1
11		833	0
12		833	1
13		833	0
14		833	1
15		1000	0

Pin Name
LED
RxD
TxD

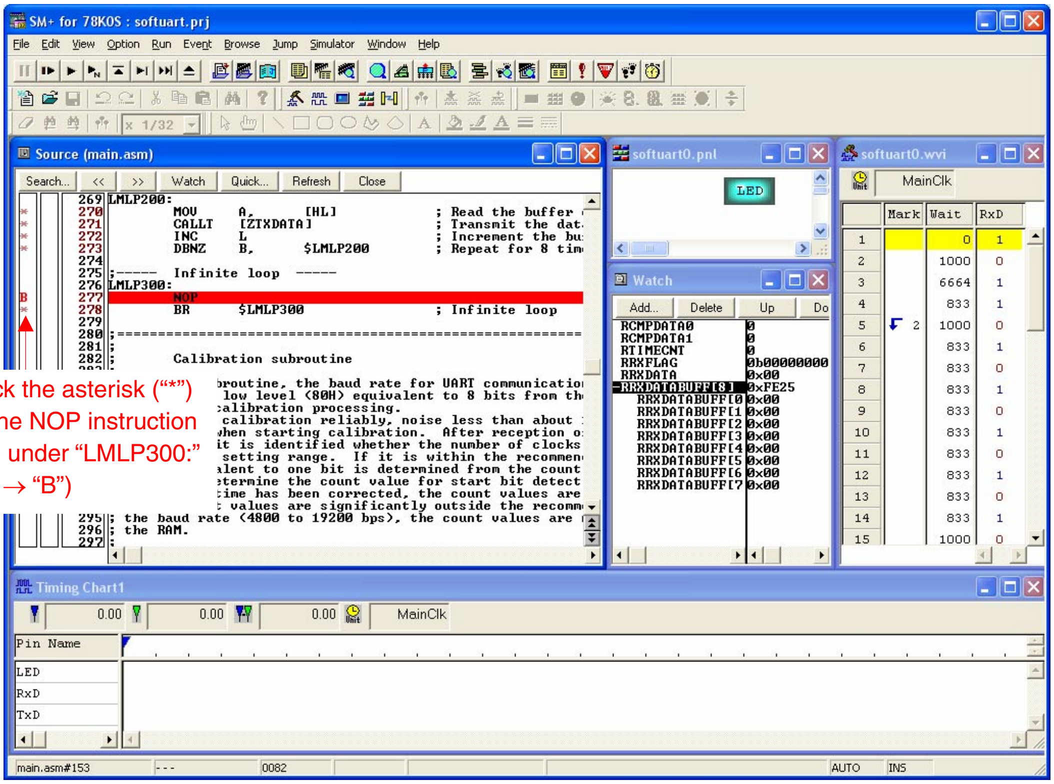
(3) The first character changes from a plus sign (“+”) to a minus sign (“–”) and the receive data to be saved will be expanded and displayed below “–RRXDATABUFF [8]”, by selecting the watch window (Watch) and double-clicking “+RRXDATABUFF [8]”.



Watch	
RCMPDATA0	0
RCMPDATA1	0
RTIMECNT	0
RRXFLAG	0b00000000
RRXDATA	0x00
+RRXDATABUFF[8]	0xFE25

Watch	
RCMPDATA0	0
RCMPDATA1	0
RTIMECNT	0
RRXFLAG	0b00000000
RRXDATA	0x00
-RRXDATABUFF[8]	0xFE25
RRXDATABUFF[0	0x00
RRXDATABUFF[1	0x00
RRXDATABUFF[2	0x00
RRXDATABUFF[3	0x00
RRXDATABUFF[4	0x00
RRXDATABUFF[5	0x00
RRXDATABUFF[6	0x00
RRXDATABUFF[7	0x00

(4) Select the source text window (Source (main.asm)) and set a break point to the NOP instruction line under label “LMLP300:” to stop simulation after completion of all processing.


SM+ for 78K0S : softuart.prj
File Edit View Option Run Event Browse Jump Simulator Window Help
Source (main.asm)
Search... << >> Watch Quick... Refresh Close
softuart0.pnl
LED
Watch
Add... Delete Up Do
softuart0.wvi
MainClk
Timing Chart1
0.00 0.00 0.00 MainClk
Pin Name
LED
RxD
TxD
main.asm#153 --- 0082 AUTO INS

```
LMLP200:
        MOV     A,      [HL]            ; Read the buffer
        CALLT   [ZTXDATA]               ; Transmit the dat
        INC     L                       ; Increment the bu
        DBNZ    B,      $LMLP200        ; Repeat for 8 tim

;-----  Infinite loop  -----
LMLP300:
        NOP
        BR      $LMLP300                ; Infinite loop

;==================================================
;
;       Calibration subroutine
```

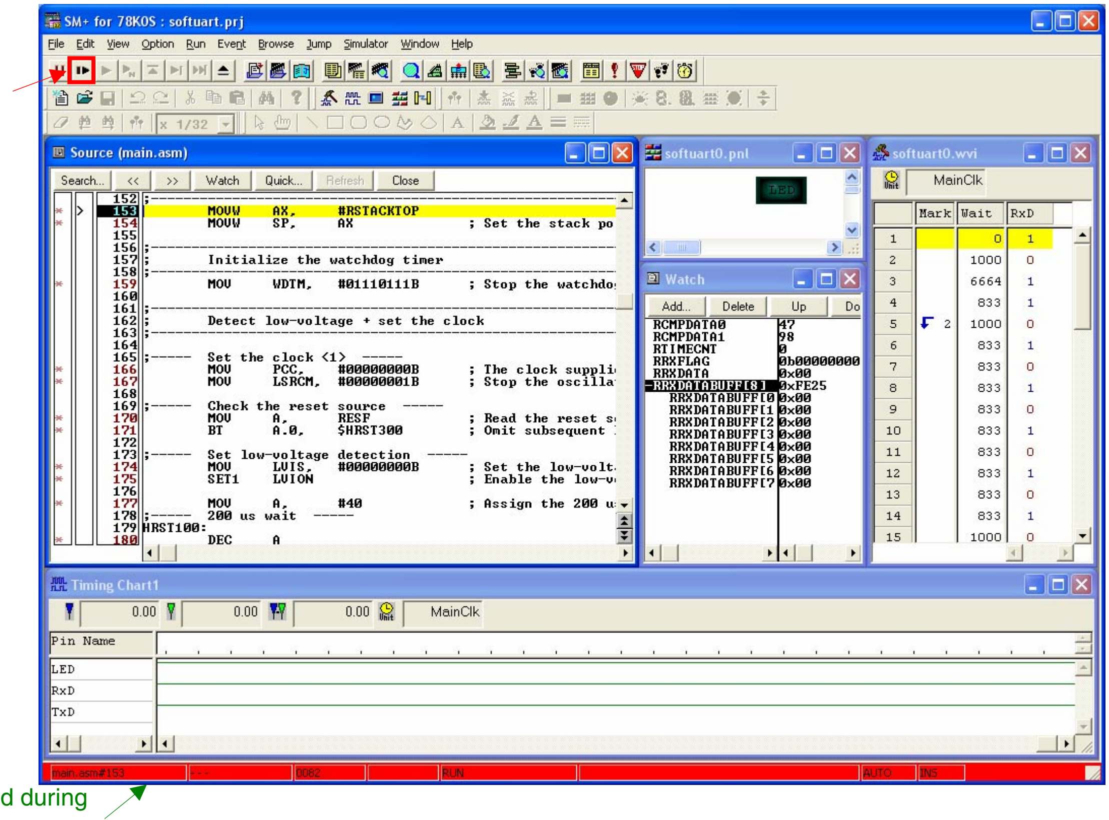
```
broutine, the baud rate for UART communicatio
 low level (80H) equivalent to 8 bits from th
calibration processing.
 calibration reliably, noise less than about
when starting calibration.  After reception o
it is identified whether the number of clocks
 setting range.  If it is within the recommen
alent to one bit is determined from the count
etermine the count value for start bit detect
time has been corrected, the count values are
t values are significantly outside the recomm
; the baud rate (4800 to 19200 bps), the count values are
; the RAM.
;
```

```
RCMPDATA0            0
RCMPDATA1            0
RTIMECNT             0
RRXFLAG              0b00000000
RRXDATA              0x00
-RRXDATABUFF[8]      0xFE25
   RRXDATABUFF[0     0x00
   RRXDATABUFF[1     0x00
   RRXDATABUFF[2     0x00
   RRXDATABUFF[3     0x00
   RRXDATABUFF[4     0x00
   RRXDATABUFF[5     0x00
   RRXDATABUFF[6     0x00
   RRXDATABUFF[7     0x00
```

	Mark	Wait	RxD
1		0	1
2		1000	0
3		6664	1
4		833	1
5	2	1000	0
6		833	1
7		833	0
8		833	1
9		833	0
10		833	1
11		833	0
12		833	1
13		833	0
14		833	1
15		1000	0

Click the asterisk (“*”) in the NOP instruction line under “LMLP300:” (“*” → “B”)

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


SM+ for 78K0S : softuart.prj
File Edit View Option Run Event Browse Jump Simulator Window Help
Source (main.asm)
Search... << >> Watch Quick... Refresh Close
softuart0.pnl
LED
Watch
Add... Delete Up Do
softuart0.wvi
MainClk
Timing Chart1
0.00 0.00 0.00 MainClk
Pin Name
LED
RxD
TxD
main.asm#153 --- 0082 RUN AUTO INS

```
;--------------------------------------------------
        MOVW    AX,     #RSTACKTOP
        MOVW    SP,     AX              ; Set the stack po

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

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

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

;-----  Check the reset source  -----
        MOV     A,      RESF            ; Read the reset s
        BT      A.0,    $HRST300        ; Omit subsequent

;-----  Set low-voltage detection  -----
        MOV     LVIS,   #00000000B      ; Set the low-volt
        SET1    LVION                   ; Enable the low-v

        MOV     A,      #40             ; Assign the 200 u
;-----  200 us wait  -----
HRST100:
        DEC     A
```

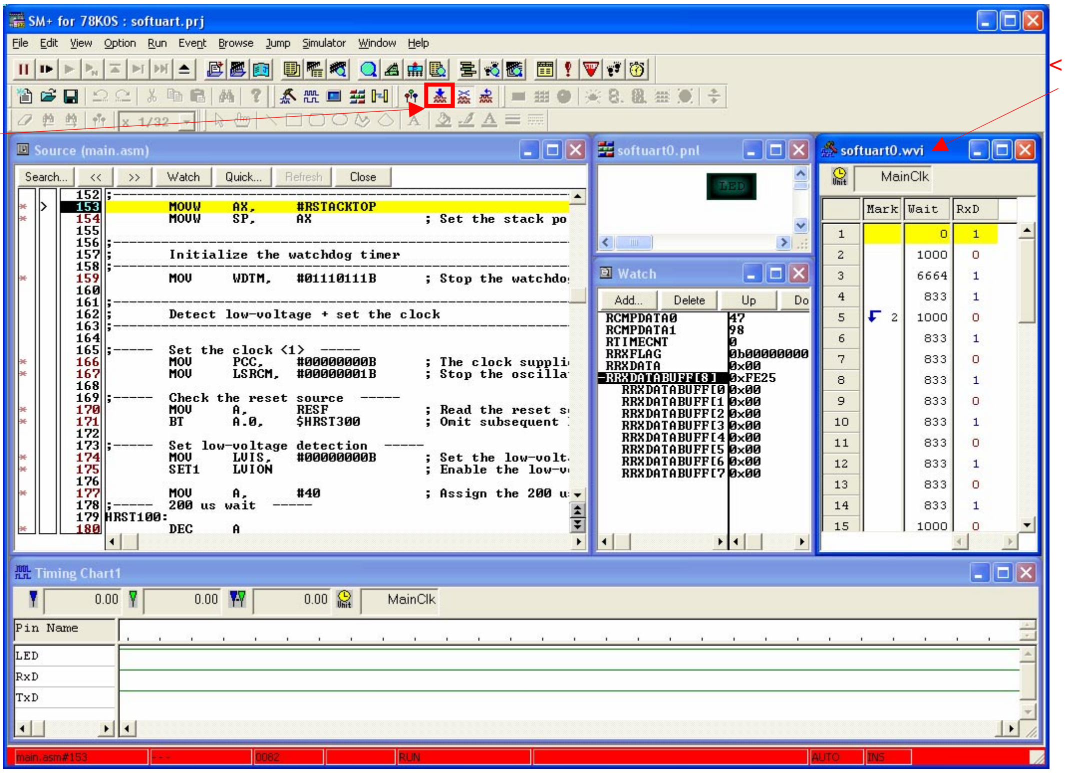
```
RCMPDATA0            47
RCMPDATA1            98
RTIMECNT             0
RRXFLAG              0b00000000
RRXDATA              0x00
-RRXDATABUFF[8]      0xFE25
   RRXDATABUFF[0     0x00
   RRXDATABUFF[1     0x00
   RRXDATABUFF[2     0x00
   RRXDATABUFF[3     0x00
   RRXDATABUFF[4     0x00
   RRXDATABUFF[5     0x00
   RRXDATABUFF[6     0x00
   RRXDATABUFF[7     0x00
```

	Mark	Wait	RxD
1		0	1
2		1000	0
3		6664	1
4		833	1
5	2	1000	0
6		833	1
7		833	0
8		833	1
9		833	0
10		833	1
11		833	0
12		833	1
13		833	0
14		833	1
15		1000	0

Click

This turns red during program execution.

(6) Select the signal data editor window (softuart0.wvi) and click (signal input start button).


SM+ for 78K0S : softuart.prj
File Edit View Option Run Event Browse Jump Simulator Window Help
x 1/32
Source (main.asm)
Search... << >> Watch Quick... Refresh Close
softuart0.pnl
LED
Watch
Add... Delete Up Do
softuart0.wvi
Unit MainClk
Timing Chart1
0.00 0.00 0.00 Unit MainClk
Pin Name
LED
RxD
TxD
main.asm#153 --- 0082 RUN AUTO INS

```
;----------------------------------------------------------------
        MOVW    AX,     #RSTACKTOP
        MOVW    SP,     AX                      ; Set the stack po

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

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

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

;-----  Check the reset source  -----
        MOV     A,      RESF                    ; Read the reset s
        BT      A.0,    $HRST300                ; Omit subsequent 

;-----  Set low-voltage detection  -----
        MOV     LVIS,   #00000000B              ; Set the low-volt
        SET1    LVION                           ; Enable the low-v

        MOV     A,      #40                     ; Assign the 200 u
;-----  200 us wait  -----
HRST100:
        DEC     A
```

```
RCMPDATA0         47
RCMPDATA1         98
RTIMECNT          0
RRXFLAG           0b00000000
RRXDATA           0x00
-RRXDATABUFF[8]   0xFE25
  RRXDATABUFF[0   0x00
  RRXDATABUFF[1   0x00
  RRXDATABUFF[2   0x00
  RRXDATABUFF[3   0x00
  RRXDATABUFF[4   0x00
  RRXDATABUFF[5   0x00
  RRXDATABUFF[6   0x00
  RRXDATABUFF[7   0x00
```

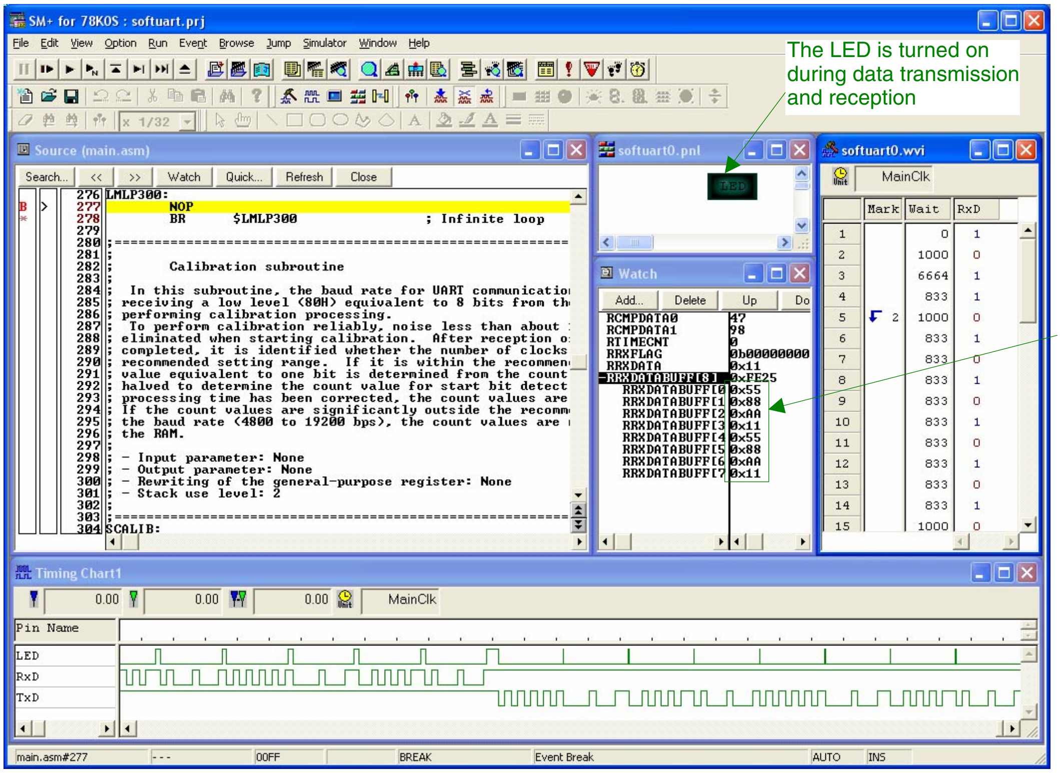
	Mark	Wait	RxD
1		0	1
2		1000	0
3		6664	1
4		833	1
5	↲ 2	1000	0
6		833	1
7		833	0
8		833	1
9		833	0
10		833	1
11		833	0
12		833	1
13		833	0
14		833	1
15		1000	0

<1> Select the window

<2> Click

Signal input

(7) Data transmission and reception will be simulated by software UART and stopped after completion of all processing.


SM+ for 78K0S : softuart.prj
File Edit View Option Run Event Browse Jump Simulator Window Help
x 1/32
Source (main.asm)
Search... << >> Watch Quick... Refresh Close
softuart0.pnl
LED
Watch
Add... Delete Up Do
softuart0.wvi
Unit MainClk
Timing Chart1
0.00 0.00 0.00 Unit MainClk
Pin Name
LED
RxD
TxD
main.asm#277 --- 00FF BREAK Event Break AUTO INS

```
LMLP300:
        NOP
        BR      $LMLP300                        ; Infinite loop

;=================================================================
;
;       Calibration subroutine
;
;  In this subroutine, the baud rate for UART communicatio
; receiving a low level (80H) equivalent to 8 bits from th
; performing calibration processing.
;  To perform calibration reliably, noise less than about 
; eliminated when starting calibration.  After reception o
; completed, it is identified whether the number of clocks
; recommended setting range.  If it is within the recommen
; value equivalent to one bit is determined from the count
; halved to determine the count value for start bit detect
; processing time has been corrected, the count values are
; If the count values are significantly outside the recomm
; the baud rate (4800 to 19200 bps), the count values are 
; the RAM.
;
; - Input parameter: None
; - Output parameter: None
; - Rewriting of the general-purpose register: None
; - Stack use level: 2
;
;=================================================================
SCALIB:
```

```
RCMPDATA0         47
RCMPDATA1         98
RTIMECNT          0
RRXFLAG           0b00000000
RRXDATA           0x11
-RRXDATABUFF[8]   0xFE25
  RRXDATABUFF[0   0x55
  RRXDATABUFF[1   0x88
  RRXDATABUFF[2   0xAA
  RRXDATABUFF[3   0x11
  RRXDATABUFF[4   0x55
  RRXDATABUFF[5   0x88
  RRXDATABUFF[6   0xAA
  RRXDATABUFF[7   0x11
```

	Mark	Wait	RxD
1		0	1
2		1000	0
3		6664	1
4		833	1
5	↲ 2	1000	0
6		833	1
7		833	0
8		833	1
9		833	0
10		833	1
11		833	0
12		833	1
13		833	0
14		833	1
15		1000	0

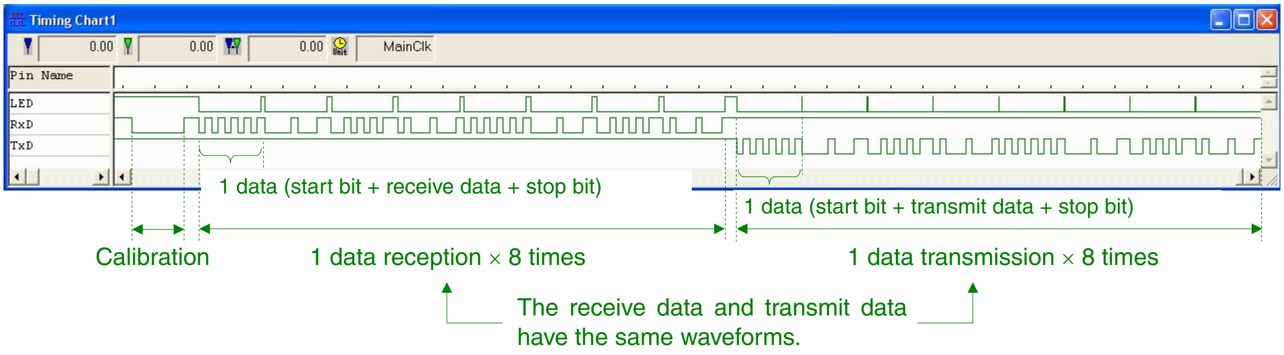
The LED is turned on during data transmission and reception

The receive data is displayed

The I/O waveforms are displayed

(8) The I/O waveforms will be displayed in the timing chart window and the received data will be displayed in the watch window as follows.

- Timing chart window



1 data (start bit + receive data + stop bit)

1 data (start bit + transmit data + stop bit)

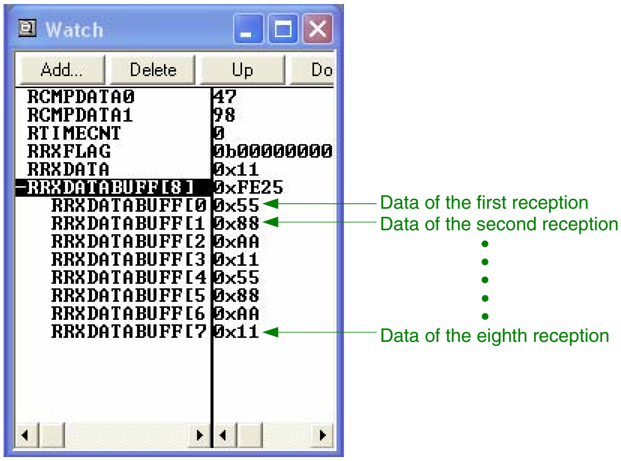
Calibration

1 data reception × 8 times

1 data transmission × 8 times

The receive data and transmit data have the same waveforms.

- Watch window



```
RCMPDATA0          47
RCMPDATA1          98
RTIMECNT           0
RRXFLAG            0b00000000
RRXDATA            0x11
-RRXDATABUFF[8]    0xFE25
   RRXDATABUFF[0   0x55
   RRXDATABUFF[1   0x88
   RRXDATABUFF[2   0xAA
   RRXDATABUFF[3   0x11
   RRXDATABUFF[4   0x55
   RRXDATABUFF[5   0x88
   RRXDATABUFF[6   0xAA
   RRXDATABUFF[7   0x11
```

Data of the first reception

Data of the second reception

•
•
•
•
•

Data of the eighth reception

CHAPTER 6 RELATED DOCUMENTS

<R>

Document Name		Japanese/English
78K0S/KU1+ User's Manual		PDF
78K0S/KY1+ User's Manual		PDF
78K0S/KA1+ User's Manual		PDF
78K0S/KB1+ User's Manual		PDF
78K/0S Series Instructions User's Manual		PDF
RA78K0S Assembler Package User's Manual	Language	PDF
	Operation	PDF
CC78K0S C Compiler User's Manual	Language	PDF
	Operation	PDF
PM+ Project Manager User's Manual		PDF
SM+ System Simulator Operation User's Manual		PDF
Flash Programming Manual (Basic) MINICUBE2 version	78K0S/KU1+	PDF
	78K0S/KY1+	PDF
	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 (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

```
;*******************************************************************************
;
;     NEC Electronics      78K0S/KB1+
;
;*******************************************************************************
;     78K0S/KB1+  Sample program
;*******************************************************************************
;     Software UART
;*******************************************************************************
;<<History>>
;     2007.9.--   Release
;*******************************************************************************
;
;<<Overview>>
;
;This sample program shows an example of UART communication by software
;control.  After completion of the initial settings, calibration is
;performed by receiving a low level (= 80H) equivalent to 8 bits and
;the baud rate is determined.  After completion of calibration, data of
;8 characters is received as a reception test, which is then transmitted
;as a transmission test.  When receiving data, the input level of the
;receive pin is observed, and calibration and data reception are started
;by using the detection of a low level as the trigger.  The baud rate is
;recommended to be set within a range of 4800 to 19200 bps and is set to
;9600 bps by default when calibration is not performed.  Furthermore, the
;LED is turned on during transmission and reception.
;
;
;  <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
;
;
;  <Serial communication protocol>
;
;  - Baud rate:               4800 to 19200 bps (9600 bps by default)
;  - Data character length:   8 bits
;  - Parity specification:     No parity
;  - Number of stop bits:      1 bit or 2 bits (1 bit by default)
;  - Start bit specification: LSB first
;
;
;  <About receive errors>
;
;  - Only framing errors are detected.
;  - Parity errors and overrun errors are not detected.
;
```

```
;
;<<I/O port settings>>
;
;  Input: P45
;  Output: P00-P03, P20-P23, P30-P33, P40-P44, P46, P47, P120-P123, P130
;  # All unused ports are set as the output mode.
;
;*****************************************************************************


;=============================================================================
;
;     Define the symbol
;
;=============================================================================
PTXD        EQU   P4.0        ; Pin for UART transmission (TxD pin)
PRXD        EQU   P4.5        ; Pin for UART reception (RxD pin)
PLED        EQU   P2.0        ; Pin for the LED displaying the transmit and
receive statuses

CSTOPBIT    EQU   1           ; Specify the number of stop bits

CCALOFFSET  EQU   (10+20-13)/16     ; For correcting the 17 clocks when
starting calibration
CTROFFSET   EQU   (6+18+18+6)/8     ; For correcting the 48 clocks during
transmission and reception
CSTOFFSET   EQU   (10+26+5)/8 ; For correcting the 41 clocks when the start
bit is detected

CB4800            EQU   202                     ; 1-bit count value at 4800
baud
CHB4800           EQU   (CB4800+CTROFFSET)/2-CSTOFFSET      ; Start bit count
value at 4800 baud
CB9600            EQU   98                      ; 1-bit count value at 9600
baud
CHB9600           EQU   (CB9600+CTROFFSET)/2-CSTOFFSET      ; Start bit count
value at 9600 baud
CB19200           EQU   46                      ; 1-bit count value at 19200
baud
CHB19200    EQU   (CB19200+CTROFFSET)/2-CSTOFFSET     ; Start bit count value
at 19200 baud

;=============================================================================
;
;     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    IRESET            ;(1C) INTSRE6
            DW    IRESET            ;(1E) INTSR6
            DW    IRESET            ;(20) INTST6

;=============================================================================
;
;     CALLT table
;
;  The instruction code of a subroutine that is frequently called can be
;  shortened by using the CALLT instruction that is a 1-byte call instruction.
;
;=============================================================================
XCALT CSEG  CALLT0
ZRXDATA:    DW    SRXDATA           ; UART receive subroutine
ZTXDATA:    DW    STXDATA           ; UART transmit subroutine

;=============================================================================
;
;     Define the RAM
;
;=============================================================================
DRAM  DSEG  SADDRP
RRXDATA:    DS    1           ; Receive data (paired with the receive status)
RRXFLAG:    DS    1           ; Receive status flag
                              ; (Framing error if bit 0 is 1)
      DSEG  SADDR
RCMPDATA0:  DS    1           ; For a start bit count
RCMPDATA1:  DS    1           ; For a 1-bit interval count
RTIMECNT:   DS    1           ; For an actual count
RRXDATABUFF:      DS    8     ; Data buffer for a transmission and reception
test

;=============================================================================
;
;     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,   #00000001B  ; Set output latches of P21-P23 as low, P20 as
high (turn off LED)
      MOV   PM2,  #11110000B  ; Set P20-P23 as output mode

;-----------------------------------------------------------------------------
;     Initialize the port 3
```

```
;-----------------------------------------------------------------------------
      MOV   P3,   #00000000B  ; Set output latches of P30-P33 as low
      MOV   PM3,  #11110000B  ; Set P30-P33 as output mode

;-----------------------------------------------------------------------------
;     Initialize the port 4
;-----------------------------------------------------------------------------
      MOV   P4,   #00000001B  ; Set output latches of P41-P47 as low, P40 as
high (set for serial transmission)
      MOV   PU4,  #00100000B  ; Connect on-chip pull-up resistor to P45
      MOV   PM4,  #00100000B  ; Set P40-P44, P46, and P47 as output mode, P45
(for serial reception) as input mode

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

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

;-----------------------------------------------------------------------------
;     Initialize the RAM
;-----------------------------------------------------------------------------
      MOV   RCMPDATA1, #CB9600     ; 1-bit count timer default value (9600
bps)
      MOV   RCMPDATA0, #CHB9600    ; Start bit count timer default value
(9600 bps)
      MOVW  AX,   #0000H
      MOVW  RRXDATA, AX       ; Initialize the receive data & receive status

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

;-----  Calibration  -----
      CALL  !SCALIB           ; Calibration processing (wait for 80H
reception)

;-----  Reception test  -----
      MOVW  HL,   #RRXDATABUFF; Specify the buffer start address
      MOV   B,    #8          ; Specify the number of receptions
LMLP100:
      CALLT [ZRXDATA]         ; Receive the data
      MOV   [HL], A           ; Write the data to the buffer
      INC   L                 ; Increment the buffer address by 1
      DBNZ  B,    $LMLP100    ; Repeat for 8 times

;-----  Wait for processing of the other party of communication  -----
      MOV   A,    RCMPDATA1   ; Read the 1-bit interval count data
LMLP150:
      NOP                     ; Set this wait time by taking the processing
time until transmission completion and
```

```
      NOP                     ; enabling of reception by the other party of
communication into consideration
      DEC   A
      BNZ   $LMLP150

;-----  Transmission test  -----
      MOVW  HL,   #RRXDATABUFF; Specify the buffer start address
      MOV   B,    #8          ; Specify the number of transmissions
LMLP200:
      MOV   A,    [HL]        ; Read the buffer data
      CALLT [ZTXDATA]         ; Transmit the data
      INC   L                 ; Increment the buffer address by 1
      DBNZ  B,    $LMLP200    ; Repeat for 8 times

;-----  Infinite loop  -----
LMLP300:
      NOP
      BR    $LMLP300          ; Infinite loop

;=============================================================================
;
;     Calibration subroutine
;
;  In this subroutine, the baud rate for UART communication is determined by
; receiving a low level (80H) equivalent to 8 bits from the RxD pin and thus
; performing calibration processing.
;  To perform calibration reliably, noise less than about 1.5 us is
; eliminated when starting calibration.  After reception of the low level is
; completed, it is identified whether the number of clocks is within the
; recommended setting range.  If it is within the recommended range, a count
; value equivalent to one bit is determined from the count value, which is
; halved to determine the count value for start bit detection.  After each
; processing time has been corrected, the count values are stored into the RAM.
; If the count values are significantly outside the recommended range for
; the baud rate (4800 to 19200 bps), the count values are not stored into
; the RAM.
;
; - Input parameter: None
; - Output parameter: None
; - Rewriting of the general-purpose register: None
; - Stack use level: 2
;
;=============================================================================
SCALIB:
      PUSH  AX                ; Save the AX register data to the stack
      PUSH  HL                ; Save the HL register data to the stack

;-----  Processing before starting calibration  -----
JCAL000:
      BF    PRXD, $JCAL000    ; If the RxD pin is 0, wait until it becomes 1
      DI                      ; Disable vector interrupt

;-----  Calibration processing  -----
JCAL100:
      BT    PRXD, $JCAL100    ;10: Wait for calibration start
      BT    PRXD, $JCAL100    ;10: If noise is present, return to waiting for
calibration start

      NOP                     ; 2: For time adjustment
      NOP                    ; 2: For time adjustment
      MOVW  HL,   #CCALOFFSET ; 6: Time correction
JCAL200:
      NOP                    ; 2:
      INCW  HL               ; 4: Time measurement
      BF    PRXD, $JCAL200   ;10: Wait for RxD pin to become 1

;-----  Identify the calibration result  -----
      MOVW  AX,   HL
      CLR1  CY
      RORC  A,    1          ; Multiply the result by 1/2
      XCH   A,    X
      RORC  A,    1
      XCH   A,    X
      CLR1  CY
      RORC  A,    1          ; Multiply by 1/2 again (multiply the result by
1/4)
      XCH   A,    X
      RORC  A,    1
      XCH   A,    X
      CMP   A,    #0         ; Are the higher 8 bits 0?
      BNZ   $JCAL300         ; Exit if too slow
      XCH   A,    X
      CMP   A,    #45        ; Lower-limit check
      BC    $JCAL300         ; Exit if too fast
                             ; CY = 0
;-----  Save to the timer count register  -----
      MOV   RCMPDATA1, A     ; Set the RCMPDATA1 setting value to the save
area
      SUB   RCMPDATA1, #CTROFFSET   ; Correct the data equivalent to 48 clocks
during transmission and reception
      RORC  A,    1          ; Multiply by 1/2 again
      SUB   A,    #CSTOFFSET ; Correct start-bit processing
      MOV   RCMPDATA0, A     ; For start-bit processing

JCAL300:
      POP   HL               ; Restore the HL register data
      POP   AX               ; Restore the AX register data
      RET

;==============================================================================
;
;     UART receive subroutine
;
;  In this subroutine, data receive processing equivalent to one character
; is performed.
;  To perform data reception reliably, noise less than about 1.5 us is
; eliminated when starting reception.  After start bit detection, 0 or 1 is
; identified at the center of the 1-bit data that is then stored.  The 1-bit
; receive subroutine is used to identify and store the 1-bit data, and the
; data is stored into RRXDATA (2 bytes) when stop bit detection has been
; completed.  At this time, the receive data is stored into the lower one
; byte (RRXDATA) and the receive status flag into the higher one byte
; (RRXFLAG).
; Furthermore, passing of the receive data can be performed both with the
; RAM and the AX register, because the receive data is stored into the A
; register and the receive status flag into the X register, and processing
; is returned from the subroutine.
;
```

```
; - Input parameter: None
; - Output parameters: A register (receive data), X register (receive status
flag)
; - Rewriting of the general-purpose register: AX register
; - Stack use level: 2
;
;=============================================================================
SRXDATA:
      PUSH  BC                ; Save the BC register data to the stack

;-----  Processing before starting reception  -----
JRXD000:
      BF    PRXD, $JRXD000    ; If the RxD pin is 0, wait until it becomes 1
      DI                      ; Disable vector interrupt


;-----  Start bit detection processing  -----
JRXD100:
      BT    PRXD, $JRXD100    ;10: Wait for start bit detection
      BT    PRXD, $JRXD100    ;10: If noise is present, return to waiting for
start bit detection

      CLR1  PLED              ; 6: Turn on the LED (during data reception)
      NOP                     ; 2: For time adjustment
      MOV   A,    RCMPDATA0   ; 4: Read the setting value
      MOV   RTIMECNT, A       ; 4: Set up to the bit center
JRXD200:
      DBNZ  RTIMECNT, $JRXD200; 8: Wait for the start bit center
      BT    PRXD, $JRXD100    ;10: If the start bit is not detected, return to
waiting for detection

      NOP                     ; 2: For time adjustment
      MOV   B,    #8+1        ; 6: Set the remaining number of receive bits
      MOVW  AX,   #0000H      ; 6: Set the initial data

;-----  Data receive processing  -----
JRXD300:
      CALL  !SRXBIT           ; 6: Receive the bit
      DBNZ  B,    $JRXD300    ; 6: Count the number of receive bits

      SET1  PLED              ; Turn off the LED (data reception end)

;-----  Receive data save processing  -----
      XCH   A,    X           ; Save the receive data to the X register
      NOT1  CY
      ROLC  A,    1           ; Set bit 0 to 1 if the stop bit is not detected
      MOVW  RRXDATA, AX       ; Save the receive data and error status
      XCH   A,    X           ; Store the receive data to the A register
                              ; Store the error status to the X register
      POP   BC                ; Restore the BC register data
      RET

;-----------------------------------------------------------------------------
;     1-bit receive subroutine
;
; - Input parameters: A register (receive data), CY flag (receive bit)
; - Output parameters: A register (receive data), CY flag (receive bit)
; - Rewriting of the general-purpose register: A register
; - Stack use level: 0
```

```
;
;-----------------------------------------------------------------------------
SRXBIT:
      NOP                     ; 2: For adjusting the time with transmit
processing
      RORC  A,    1           ; 2: Retrieve the receive data (CY flag) by
right-shifting
      MOV   RTIMECNT, A       ; 4: Save the receive data
      MOV   A,    RCMPDATA1   ; 4: Get the receive time count value
      XCH   A,    RTIMECNT    ; 6: Set the count value & restore the receive
data
JRXB100:
      DBNZ  RTIMECNT, $JRXB100      ; 8*n: Wait time

      BT      PRXD,     $JRXB200    ;10: Check the receive data
      CLR1  CY                ; 2: CY is 0 if 0 is received
      RET                     ; 6:
JRXB200:
      SET1  CY                ; 2: CY is 1 if 1 is received
      RET                     ; 6:

;=============================================================================
;
;     UART transmit subroutine
;
;  In this subroutine, data transmission equivalent to one character is
; performed.
;  The data to be transmitted is stored into the A register and this
; subroutine is called, as described in the example below.  The 1-bit
; transmit subroutine is used to transmit the data and processing is
; returned from this subroutine when stop bit transmission has been
; completed.
;
; Program example:
;     MOV   A,    #54H  ; Store 54H into the A register
;     CALLT [ZTXDATA]   ; Call the UART transmit subroutine
;
; - Input parameter: A register (transmit data)
; - Output parameter: None
; - Rewriting of the general-purpose register: A register
; - Stack use level: 2
;
;=============================================================================
STXDATA:
      PUSH  BC                ; Save the BC register data to the stack

;-----  Processing before starting transmission  -----
      DI                      ; Disable vector interrupt

;-----  Start bit transmit processing  -----
      CLR1  PTXD              ; 6: Transmit the start bit

      CLR1  PLED              ; 6: Turn on the LED (during data transmission)
      MOV   B,    #1+8+CSTOPBIT     ; 6: Set the number of transmit bits

;-----  Data transmit processing  -----
JTXD100:
      CALL  !STXBIT           ; 6: Transmit the bit
      DBNZ  B,    $JTXD100    ; 6: Count the number of transmit bits
      SET1  PLED              ; Turn off the LED (data transmission end)

      POP   BC                ; Restore the BC register data
      RET

;-----------------------------------------------------------------------------
;     1-bit transmit subroutine
;
; - Input parameter: A register (transmit data)
; - Output parameter: A register (transmit data)
; - Rewriting of the general-purpose register: A register
; - Stack use level: 0
;
;-----------------------------------------------------------------------------
STXBIT:
      MOV   RTIMECNT, A       ; 4: Save the transmit data
      MOV   A,    RCMPDATA1   ; 4: Get the transmit time count value
      XCH   A,    RTIMECNT    ; 6: Set the count value & restore the transmit
data
      SET1  CY                ; 2: Set the data to 1 after output
      RORC    A,  1           ; 2: Right-shift the transmit data to the CY
flag
JTXB100:
      DBNZ  RTIMECNT, $JTXB100; 8*n: Wait time

      BC    $JTXB200          ; 6: Branch if CY is 1
      CLR1    PTXD            ; 6: Transmit 0
      RET                     ; 6:
JTXB200:
      SET1    PTXD            ; 6: Transmit 1
      RET                     ; 6:

end
```

● op.asm

```
;=============================================================================
;
;     Option byte
;
;=============================================================================
OPBT  CSEG  AT    0080H
        DB      10011100B       ; Option byte area
;                   ||||
;                   |||+-----------Low-speed internal oscillator can be
stopped by software
;                   |++------------High-speed internal oscillation clock (8
MHz) is selected for system clock source
;                   +------------- P34/RESET pin is used as RESET pin

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

end
```

APPENDIX B REVISION HISTORY

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

Edition	Date Published	Page	Revision
1st edition	December 2007	–	–
2nd edition	September 2008	p.25	CHAPTER 5 OPERATION CHECK USING SYSTEM SIMULATOR SM+ • Modification of description in Caution ((as of September 2007) → (as of July 2008))
		pp.25 to 27	Modification of 5.1 Building the Sample Program
		p.27	5.2 Operation with SM+ • Addition of (1)
		p.32	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/