Introduction

This application note explains how to implement the software handshake communication in UART communication through the serial array unit (SAU).

Target Device

RL78/G10 microcomputer

10-pin: ROM = 2 KB, 4 KB (R5F10Y16, R5F10Y17)
16-pin: ROM = 2 KB, 4 KB (R5F10Y46, R5F10Y47)

When applying the sample program covered in this application note to another microcomputer, modify the program according to the specifications for the target microcomputer and conduct an extensive evaluation of the modified program.

Contents

1. Specifications ................................................................................................................................... 3
   1.1 Handshake Communication Overview .............................................................................................. 6
   1.2 Ring Buffer Structure ......................................................................................................................... 8
   1.3 Baud Rate Correction ......................................................................................................................... 9

2. Operation Check Conditions ........................................................................................................ 11

3. Related Application Notes ............................................................................................................ 11

4. Description of Hardware .............................................................................................................. 12
   4.1 Hardware Configuration Example ..................................................................................................... 12
   4.2 List of Pins to be Used .................................................................................................................... 13

5. Description of Software ................................................................................................................ 14
   5.1 Operation Overview ......................................................................................................................... 14
   5.2 List of Option Byte Settings ........................................................................................................... 15
   5.3 List of Constants ............................................................................................................................. 16
   5.4 List of Variables ............................................................................................................................. 17
   5.5 List of Functions (Subroutines) ......................................................................................................... 18
   5.6 Function (Subroutine) Specifications ............................................................................................... 18
   5.7 Flowcharts ...................................................................................................................................... 22
      5.7.1 Overall Flow ............................................................................................................................... 22
      5.7.2 I/O Port Settings ......................................................................................................................... 23
      5.7.3 Setting of Clock Generation Circuit .......................................................................................... 24
      5.7.4 SAU Setting ............................................................................................................................. 25
5.7.5 TAU Setting ................................................................................................................... 27
5.7.6 Main Processing............................................................................................................. 28
5.7.7 Main Initialization Processing ...................................................................................... 31
5.7.8 Wait Processing (millisecond units) ........................................................................... 32
5.7.9 UART0 Transmission Enabling Processing ............................................................... 33
5.7.10 UART0 Reception Enabling Processing ................................................................. 34
5.7.11 UART0 Operation Stop Processing ......................................................................... 35
5.7.12 UART0 Data Transmission Processing ................................................................. 36
5.7.13 UART0 Control Code Transmission Processing ................................................... 37
5.7.14 Ring Buffer Store Processing .................................................................................. 38
5.7.15 Ring Buffer Store and XOFF Transmission Processing ........................................ 39
5.7.16 Ring Buffer Read Processing ................................................................................. 40
5.7.17 Ring Buffer Read and XON Transmission Processing ........................................ 41
5.7.18 UART0 Reception End Interrupt Processing ....................................................... 42
5.7.19 UART0 Reception Error Interrupt Processing .................................................... 44
5.7.20 UART0 Transmission End Interrupt Processing ............................................... 45
5.7.21 INTTM01 Interrupt Processing ............................................................................ 46
5.7.22 INT00 Interrupt Processing ................................................................................. 48
5.7.23 INTP1 Interrupt Processing ................................................................................. 49

6. Operation Verification ........................................................................................................ 50
   6.1 Transmission Waveform Verification .......................................................................... 50
   6.2 Baud Rate Correction Operation Verification .......................................................... 51
   6.3 Example of Baud Rate Correction Failure ................................................................. 54
   6.4 Transitions between Operating Modes ..................................................................... 55
   6.5 Software Handshake Communication ....................................................................... 56

7. Sample Code ....................................................................................................................... 57

8. Reference Documents ....................................................................................................... 57
1. Specifications

This application note describes how to perform software handshake communication in UART communication through the serial array unit (SAU). In this communication, XON (11H) is used as a transmission enable code and XOFF (13H) is used as a transmission disable code, and the received data is stored in a 16-byte ring buffer. When the free space in the buffer is three bytes or less, XOFF (13H) is issued to disable transmission. When the buffer is read and the number of bytes of data stored in the received data area in the buffer is three bytes or less, XON (11H) is issued to resume transmission.

This sample program performs UART communication between two evaluation boards. Each of the evaluation boards has a LED1 and a LED2 to show the status of the handshake communication. One of the LEDs operates as the data transmission side and the other as the data reception side, therefore the flow of data is unidirectional from the data transmission side to the data reception side. However, since it is necessary for the data reception side to send “XON (transmission enabled)/XOFF (transmission disabled)” to the data transmission side, these control codes are transmitted from the data reception side to the data transmission side. The two evaluation boards have the same specification and the same program, and LED1 is used to distinguish between the data transmission side and the data reception side. (Data transmission side: LED1 is lit. Data reception side: LED1 is not lit.)
lists the operating modes. The operating modes are distinguished by LED1. After reset is released, both two evaluation boards operate in initialization mode. In initialization mode LED1 flashes, and in this state the board whose switch is pressed becomes the data transmission side, and the other board becomes the data reception side.

<table>
<thead>
<tr>
<th>Table 1.1 List of Operating Modes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating Mode</td>
</tr>
<tr>
<td>Initialization mode</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Correction mode</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Transmission mode</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Reception mode</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

Table 1.2 shows the on-chip functions to be used and their usage.

<table>
<thead>
<tr>
<th>Table 1.2 On-chip Functions to be Used and their Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>On-chip Function</td>
</tr>
<tr>
<td>------------------</td>
</tr>
<tr>
<td>Serial array unit 0</td>
</tr>
<tr>
<td>Timer array unit channel 1</td>
</tr>
<tr>
<td>External interrupt</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Port function</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
shows the state transition diagram (baud rate correction enabled).
1.1 Handshake Communication Overview

As the communication speed increases, the processing on the reception side may not be able to keep up, and data may be lost. In handshake communication, when it does not seem that the processing on the reception side will be able to keep up, data loss is avoided by disabling transmission until the processing is completed. Handshake communication methods may be implemented in hardware or software, and an overview of each of them is given below.

1) Handshake communication using hardware

Figure 1.1 shows an example of handshake communication that is used relatively often in microcomputers. The RTS and CTS signals are paired, where RTS is used as a transmission request signal (output), and CTS as a transmission enable signal (input). This method can communicate at high speed compared to a method implemented in software, but it is necessary to add these two signal lines.

![Figure 1.1 Operation of Handshake Communication using Hardware](image-url)
(1) Handshake communication using software

This performs handshake communication by defining control codes for enabling transmission and disabling transmission, and transmitting them as shown in Figure 1.2. (This sample program uses XON and XOFF, which are defined as control signals in the ASCII code.) It is not necessary to add signal lines, but processing of the control codes takes time, so the communication speed is slower than with a method implemented in hardware. Also, when binary data is transmitted, it is necessary to implement processing to distinguish between the control codes and the data.

![Figure 1.2 Operation of Handshake Communication Using Software](image-url)
1.2 Ring Buffer Structure

Figure 1.3 shows the structure of the ring buffer. This sample program has a 16-byte ring buffer, and when the free space in the buffer is three bytes or less, the data reception side transmits XOFF (transmission disabled) to the data transmission side, thereby preventing the loss of received data. The data reception side reads the reception buffer, and when the number of bytes of data stored in the buffer is three bytes or less, the data reception side transmits XON (transmission enabled) to the data transmission side to resume communication. In order to perform stable handshake communication using software, in consideration of the time required to transmit the control codes, the data reception side transmits XOFF (transmission disabled) to the data transmission side at the point when the free space in the buffer is three bytes.
1.3 Baud Rate Correction

In the sample program included in this application note, it is possible to change whether baud rate correction is enabled or disabled by using a constant definition. 55H is transmitted from the data transmission side, and the data reception side performs pulse width measurement. The baud rate is calculated and set from the measured value. The range of the baud rate value that can be corrected is from 1200 bps to 312500 bps. (If the baud rate exceeds 312500 bps, it is impossible to measure the baud rate because the INTTM01 interrupt processing will not be completed before the next INTTM01 interrupt is generated.) For details on the baud rate correction, refer to the related application note (Serial Array Unit (Baud Rate Correction) CC-RL (R01AN3083EJ)). Switching enable/disable of baud rate correction and the initial baud rate are set in an include file.

(1) Switching enable/disable of baud rate correction

Figure 1.4, if the constant CFLAGBRC in SETTING.inc is set to CBRCENABLE then baud rate correction is enabled, and if it is set to CBRCDISABLE then baud rate correction is disabled. By default, baud rate correction is enabled.

(2) Setting of Initial baud rate

As shown in Figure 1.4, the initial baud rate is set by setting the value of SET control instruction that corresponds to the baud rate to 1. The default baud rate is 9600 bps.

Figure 1.4 shows a setting example of initial baud rate and correction operation.

In this example, the baud rate correction is enabled and the initial baud rate is 9600 bps.

```
CBRCENABLE    .EQU 0x01
CBRCDISABLE   .EQU 0x00

:******************************************
: baud rate correction
:******************************************
CFLAGBRC      .EQU CBRCENABLE
:CFLAGBRC     .EQU CBRCDISABLE

:******************************************
: baud rate select
:******************************************
BAUDRATE1200  SET 0 : 1200bps
BAUDRATE9600  SET 1 : 9600bps
BAUDRATE38400 SET 0 : 38400bps
BAUDRATE115200 SET 0 : 115200bps
BAUDRATE312500 SET 0 : 312500bps
```

Figure 1.4 Setting Example of Initial Baud Rate and Correction Operation
Figure 1.5 shows the state transition diagram when baud rate correction is disabled.

Figure 1.5 State Transition Diagram (Baud Rate Correction Disabled)
2. Operation Check Conditions

The sample code contained in this application note has been checked under the conditions listed in the table below.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microcontroller used</td>
<td>RL78/G10 (R5F10Y47)</td>
</tr>
<tr>
<td>Operating frequency</td>
<td>High-speed on-chip oscillator (HOCO) clock: 10 MHz</td>
</tr>
<tr>
<td>Operation voltage</td>
<td>5.0V (Operation is possible over a voltage range of 2.9V to 5.5V.)</td>
</tr>
<tr>
<td></td>
<td>SPOR setting: Rising edge: 2.90 V</td>
</tr>
<tr>
<td></td>
<td>Falling edge: 2.84 V</td>
</tr>
<tr>
<td>Integrated development environment</td>
<td>CS+ for CC V3.02.00 from Renesas Electronics Corp.</td>
</tr>
<tr>
<td>Assembler</td>
<td>CC-RL V1.02.00 from Renesas Electronics Corp.</td>
</tr>
<tr>
<td>Board to be used</td>
<td>RL78/G10 target board (RTE510Y470TGB00000R)</td>
</tr>
<tr>
<td>Component for expanding circuit</td>
<td>Breadboard, LED, resistor, and tact switch</td>
</tr>
</tbody>
</table>

3. Related Application Notes

The application notes related to this application note are listed below for reference.

RL78/G10 Initialization CC-RL (R01AN2668EJ) Application Note
RL78/G10 Serial Array Unit (UART Communication) CC-RL (R01AN2918EJ) Application Note
RL78/G10 Serial Array Unit (Baud Rate Correction) CC-RL(R01AN3083EJ) Application Note
4. Description of Hardware

4.1 Hardware Configuration Example

Figure 4.1 shows an example of hardware configuration that is used in this application note.

Cautions:

1. The purpose of this circuit is only to provide the connection outline and the circuit is simplified accordingly. When designing and implementing an actual circuit, provide a proper pin treatment and make sure that the hardware’s electrical specifications are met (connect the input-only ports separately to $V_{DD}$ or $V_{SS}$ via a resistor).

2. $V_{DD}$ must be held at not lower than the reset release voltage ($V_{SPOR}$) that is specified as SPOR.
### 4.2 List of Pins to be Used

Table 4.1 lists the pins to be used and their functions.

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>I/O</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>P00/TXD0</td>
<td>Output</td>
<td>UART transmission</td>
</tr>
<tr>
<td>P01/RXD0</td>
<td>Input</td>
<td>Can also be used for one of the following functions by setting the ISC register.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- UART reception</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- TI01 input</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- INTP0 input</td>
</tr>
<tr>
<td>P04</td>
<td>Output</td>
<td>LED control</td>
</tr>
<tr>
<td>P03/(INTP1)</td>
<td>Input</td>
<td>Switch input</td>
</tr>
<tr>
<td>P02</td>
<td>Output</td>
<td>LED control</td>
</tr>
</tbody>
</table>
5. Description of Software

5.1 Operation Overview

The sample code included in this application note performs a handshake communication using the control signals shown in Table 5.1.

<table>
<thead>
<tr>
<th>Table 5.1 Control Signals Used for Handshake Communication</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Transmission Data</strong></td>
</tr>
<tr>
<td>XON (11H)</td>
</tr>
<tr>
<td>XOFF (13H)</td>
</tr>
</tbody>
</table>

In the sample program included in this application note, UART communication is performed between two evaluation boards. One of them operates as the data transmission side and the other as the data reception side, so data flow is unidirectional from the data transmission side to the data reception side. However, since it is necessary for the data reception side to send “XON (transmission enabled) / XOFF (transmission disabled)” to the data transmission side, these control codes are transmitted from the data reception side to the data transmission side.

For the software operation, four operating modes are defined as the internal state, and processing is performed based on the respective operating modes. The operating modes are distinguished by LED1. After reset is released, both two evaluation boards operate in initialization mode. In initialization mode LED1 flashes, and in this state the board whose switch is pressed becomes the data transmission side, and the other board becomes the data reception side.

Table 1.1 lists the operating modes. After reset is released, both boards operate in initialization mode. In initialization mode LED1 flashes, and in this state the board whose switch is pressed becomes the data transmission side and the other one becomes the data reception side.

The specific processing flow is as follows.

(1) Perform the initial setting of functions to be used.

   <Setting conditions>  
   - Initial setting of SAU (Perform the UART0 setting.)
   - Initial setting of TAU (Set pulse interval measurement mode for TAU01.)
   - Set the falling edge detection for INTP0/INTP1 interrupts.

(2) While flashing LED1 with an interval of approximately 500 ms, wait for the generation of either the INTP0 (UART communication start) or INTP1 (switch pressed) interrupt.

(3) If the INTP0 interrupt occurs then this board will become the data reception side, so start the TAU01 operation to measure the baud rate.

   If the INTP1 interrupt occurs then this board will become the data transmission side, so transmit the correction code 55H through the UART.

(4) The data reception side measures the pulse interval of the correction code 55H and calculates the baud rate. After applying the calculated baud rate, it transmits XON to inform the data transmission side that the transmission is currently enabled.

   The data transmission side waits for XON to be transmitted from the data reception side.

(5) The data reception side stores received data in the ring buffer, and when the switch is pressed it reads one byte of the received data in the buffer. When the free space in the buffer is three bytes or less it transmits XOFF to set the transmission disable state, and when the number of bytes of data stored in the reception buffer is three bytes or less it transmits XON to set transmission enable state.

   In transmission enable state the data transmission side transmits data when the switch is pressed. In transmission disable state it will not start a transmission operation even if the switch is pressed. If XOFF is received during a transmission operation, then the transmission end interrupt is prohibited so that the next transmission data is not set.
## 5.2 List of Option Byte Settings

Table 5.2 shows the settings of the option bytes.

### Table 5.2 Option Byte Settings

<table>
<thead>
<tr>
<th>Address</th>
<th>Setting Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>000C0H</td>
<td>11101110B</td>
<td>Disables the watchdog timer. (Stops counting after release from the reset state.)</td>
</tr>
<tr>
<td>000C1H</td>
<td>11110111B</td>
<td>SPOR detection voltage Rising edge: 2.90 V Falling edge: 2.84 V</td>
</tr>
<tr>
<td>000C2H</td>
<td>11110101B</td>
<td>HOCO: 10 MHz</td>
</tr>
<tr>
<td>000C3H</td>
<td>10000101B</td>
<td>Enables the on-chip debugger.</td>
</tr>
</tbody>
</table>
5.3 List of Constants

Table 5.3 lists the constants that are used in the sample program.

<table>
<thead>
<tr>
<th>Constant Name</th>
<th>Setting Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCODEXON</td>
<td>11H&lt;sup&gt;Note1&lt;/sup&gt;</td>
<td>Transmission enable code: XON</td>
</tr>
<tr>
<td>CCODEXOFF</td>
<td>13H&lt;sup&gt;Note1&lt;/sup&gt;</td>
<td>Transmission disable code: XOFF</td>
</tr>
<tr>
<td>CCODEBRC</td>
<td>55H</td>
<td>Value to be transmitted for baud rate correction: 55H</td>
</tr>
<tr>
<td>CMODETX</td>
<td>‘T’</td>
<td>Operating mode: Transmission mode</td>
</tr>
<tr>
<td>CMODERX</td>
<td>‘R’</td>
<td>Operating mode: Reception mode</td>
</tr>
<tr>
<td>CMODEBRC</td>
<td>‘B’</td>
<td>Operating mode: Correction mode</td>
</tr>
<tr>
<td>CMODEINI</td>
<td>‘I’</td>
<td>Operating mode: Initialization mode</td>
</tr>
<tr>
<td>CSTATEXON</td>
<td>01H</td>
<td>Communication state: Transmission enabled (XON)</td>
</tr>
<tr>
<td>CSTATEXOFF</td>
<td>00H</td>
<td>Communication state: Transmission disabled (XOFF)</td>
</tr>
<tr>
<td>CFLAGBRC</td>
<td>CBRCENABLE&lt;sup&gt;Note2&lt;/sup&gt;</td>
<td>Whether baud rate correction is enabled/disabled</td>
</tr>
<tr>
<td>CBRCENABLE</td>
<td>01H</td>
<td>Baud rate correction enabled</td>
</tr>
<tr>
<td>CBRCDISABLE</td>
<td>00H</td>
<td>Baud rate correction disabled</td>
</tr>
<tr>
<td>CSENDDATASIZE</td>
<td>03H</td>
<td>Number of bytes of transmission data</td>
</tr>
<tr>
<td>CSENDDATA</td>
<td>“ABC”&lt;sup&gt;Note3&lt;/sup&gt;</td>
<td>Transmission data</td>
</tr>
<tr>
<td>SW</td>
<td>P0.3</td>
<td>Switch</td>
</tr>
<tr>
<td>LED1</td>
<td>P0.2</td>
<td>LED1</td>
</tr>
<tr>
<td>LED2</td>
<td>P0.4</td>
<td>LED2</td>
</tr>
</tbody>
</table>

Notes:
1. The setting value can be arbitrarily changed, but CCODEXON cannot be the same value as CCODEXOFF.
2. Set CBRCENABLE or CBRCDISABLE in the include file to switch whether baud rate correction is enabled or disabled. The default value is CBRCENABLE (01H) and baud rate correction is enabled.
3. Data transmitted using the SUART0SENDDATA function. Although it can be set to an arbitrary value, always set the number of bytes to the same value as specified by the constant CDATASIZE.
## 5.4 List of Variables

Table 5.4 lists the global variables that are used in the sample program.

<table>
<thead>
<tr>
<th>Type</th>
<th>Variable Name</th>
<th>Content</th>
<th>Function Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 bits</td>
<td>COUNT</td>
<td>Used for 500 ms wait processing</td>
<td>main</td>
</tr>
<tr>
<td>8 bits</td>
<td>UASTT</td>
<td>Transmission status flag (counter)</td>
<td>SUART0SENDDATA, IINTST0</td>
</tr>
<tr>
<td>16 bits</td>
<td>SENDPNT</td>
<td>Pointer to UART transmission data</td>
<td>SUART0SENDDATA</td>
</tr>
<tr>
<td>8 bits</td>
<td>DATABUF</td>
<td>Ring buffer for storing UART received data</td>
<td>SPUTDATA, SGETDATA</td>
</tr>
<tr>
<td>8 bits</td>
<td>SETPNT</td>
<td>Pointer for storing data in ring buffer</td>
<td>SINIMAIN, SPUTDATA</td>
</tr>
<tr>
<td>8 bits</td>
<td>GETPNT</td>
<td>Pointer for reading data from ring buffer</td>
<td>SINIMAIN, SGETDATA</td>
</tr>
<tr>
<td>8 bits</td>
<td>DATACNT</td>
<td>Number of bytes of data in ring buffer</td>
<td>SINIMAIN, SPUTDATA, SPUTDATASENDXOFF, SGETDATA, SGETDATASENDXON, IINTSR0, IINTSRE0</td>
</tr>
<tr>
<td>8 bits</td>
<td>MODE</td>
<td>Indicates operating mode (initialization, correction, transmission, or reception).</td>
<td>Main, SINIMAIN, IINTSR0, IINTTM01, IINTP0, IINTP1</td>
</tr>
<tr>
<td>8 bits</td>
<td>CMCSTATE</td>
<td>Indicates communication state (XON or XOFF).</td>
<td>Main, SINIMAIN, SUART0SENDDATA, SPUTDATASENDXOFF, SGETDATA, SGETDATASENDXON, IINTSR0, IINTSRE0</td>
</tr>
<tr>
<td>8 bits</td>
<td>FLAGSWON</td>
<td>Indicates that switch is ON.</td>
<td>Main, SINIMAIN, IINTP0, IINTP1</td>
</tr>
<tr>
<td>8 bits</td>
<td>FLAGSENDING</td>
<td>Indicates that data is being transmitted.</td>
<td>SUART0SENDDATA, IINTST0</td>
</tr>
<tr>
<td>8 bits</td>
<td>SPSDATA</td>
<td>Stores the setting value for SPS0 register.</td>
<td>Main, IINTTM01</td>
</tr>
<tr>
<td>8 bits</td>
<td>DIVDATA</td>
<td>Stores the setting value for SDR0nH register.</td>
<td>Main, IINTTM01</td>
</tr>
<tr>
<td>16 bits</td>
<td>CAPTUREL</td>
<td>Variable for accumulating capture values (lower 16 bits)</td>
<td>IINTTM01, IINTP0</td>
</tr>
<tr>
<td>8 bits</td>
<td>CAPTUREH</td>
<td>Variable for accumulating capture values (upper 8 bits)</td>
<td>IINTTM01, IINTP0</td>
</tr>
<tr>
<td>8 bits</td>
<td>LPCOUNT</td>
<td>Used for counting the number of captures</td>
<td>Main, SINIMAIN, IINTTM01</td>
</tr>
</tbody>
</table>
5.5 List of Functions (Subroutines)

Table 5.5 lists the functions (subroutines) that are used in the sample program.

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Outline</th>
</tr>
</thead>
<tbody>
<tr>
<td>SINIMAIN</td>
<td>Initialization processing using the main function</td>
</tr>
<tr>
<td>SWAITMS</td>
<td>Wait processing (millisecond units)</td>
</tr>
<tr>
<td>SSTARTUART0TX</td>
<td>UART0 transmission enabling processing</td>
</tr>
<tr>
<td>SSTARTUART0RX</td>
<td>UART0 reception enabling processing</td>
</tr>
<tr>
<td>STOPUART0</td>
<td>UART0 operation stop processing</td>
</tr>
<tr>
<td>SUART0SENDDATA</td>
<td>UART0 data transmission processing</td>
</tr>
<tr>
<td>SUART0SENDCODE</td>
<td>UART0 control code transmission processing</td>
</tr>
<tr>
<td>SPUTDATA</td>
<td>Ring buffer store processing</td>
</tr>
<tr>
<td>SPUTDATASENDXOFF</td>
<td>Ring buffer store and XOFF transmission processing</td>
</tr>
<tr>
<td>SGETDATA</td>
<td>Ring buffer read processing</td>
</tr>
<tr>
<td>SGETDATASENDXON</td>
<td>Ring buffer read and XON transmission processing</td>
</tr>
<tr>
<td>IINTSR0</td>
<td>UART0 reception end interrupt processing</td>
</tr>
<tr>
<td>IINTSRE0</td>
<td>UART0 reception error interrupt processing</td>
</tr>
<tr>
<td>IINTST0</td>
<td>UART0 transmission end interrupt processing</td>
</tr>
<tr>
<td>IINTTM01</td>
<td>IINTTM01 interrupt processing</td>
</tr>
<tr>
<td>IINTP0</td>
<td>IINTP0 interrupt processing</td>
</tr>
<tr>
<td>IINTP1</td>
<td>IINTP1 interrupt processing</td>
</tr>
</tbody>
</table>

5.6 Function (Subroutine) Specifications

This section describes the specifications for the functions (subroutines) that are used in the sample program.

[Function name] SINIMAIN

| Outline | Initialization processing using the main function |
| Explaination | Initializes the variables and peripherals. |
| Argument | None |
| Return value | None |
| Remarks | None |

[Function name] SWAITMS

| Outline | Wait processing (millisecond units) |
| Explaination | Performs wait processing for the time specified by the AX register (millisecond units). |
| Argument | AX : [Time to be waited (ms)] |
| Return value | None |
| Remarks | Since wait is implemented by using loops, the wait time is not precise. |
### Function SSTARTUART0TX

**Outline**
UART0 transmission enabling processing

**Explanation**
Starts operation of channel 0 of serial array unit 0 to make the system enter a transmission wait state. Disables transmission end interrupt.

**Argument**
None

**Return value**
None

**Remarks**
None

### Function SSTARTUART0RX

**Outline**
UART0 reception enabling processing

**Explanation**
Starts operation of channel 1 of serial array unit 0 to make the system enter a reception wait state. Enables reception end interrupt and reception error interrupt.

**Argument**
None

**Return value**
None

**Remarks**
None

### Function SSTOPUART0

**Outline**
UART0 operation stop processing

**Explanation**
Stops operation of UART0.

**Argument**
None

**Return value**
None

**Remarks**
This function is used to reconfigure the baud rate.

### Function SUART0SENDDATA

**Outline**
UART0 data transmission processing

**Explanation**
Transmits data from UART0. Transmission processing is performed on an interrupt basis.

**Argument**
HL : [Address for storing transmission data]

**Return value**
None

**Remarks**
None

### Function SUART0SENDCODE

**Outline**
UART0 control code transmission processing

**Explanation**
Transmits the control codes from UART0.

**Argument**
A : [Transmission data (control codes)]

**Return value**
None

**Remarks**
None

### Function SPUTDATA

**Outline**
Ring buffer store processing

**Explanation**
Stores data if there is free space in the ring buffer.

**Argument**
A : [Data to be stored]

**Return value**
CY : [1: Buffer full, 0: Store operation completed]

**Remarks**
None
[Function name] SPUTDATASENDXOFF

Outline   Ring buffer store and XOFF transmission processing.

Explanation Stores data if there is free space in the ring buffer. As a result of storing, if free space in the buffer is three bytes or less then transmit XOFF.

Argument   A : [Data to be stored]

Return value CY : [1: Buffer full, 0: Store operation completed]

Remarks   None

[Function name] SGETDATA

Outline   Ring buffer read processing

Explanation Reads the oldest data from the ring buffer if there is any data in the buffer.

Argument   None

Return value CY : [1: No data, 0: Reading completed]

Remarks   None

[Function name] SGETDATASENDXON

Outline   Ring buffer read XON transmission processing

Explanation Reads the oldest data from the ring buffer if there is any data stored in the buffer. As a result of reading, if the number of bytes of data remaining in the buffer is three bytes or less then transmit XON.

Argument   None

Return value CY : [1: No data, 0: Reading completed]

Remarks   None

[Function name] IINTSR0

Outline   UART0 reception end interrupt processing

Explanation Performs processing based on the operating modes.

Initialization mode: Changes the mode setting to reception mode and performs the same processing as in reception mode.

Reception mode: Stores the received data in the ring buffer.

Other modes: Processes the received data as the control codes.

Argument   None

Return value None

Remarks   None

[Function name] IINTSRE0

Outline   UART0 reception error interrupt processing

Explanation Clears the reception end interrupt request. In reception mode, if there is free space in the ring buffer then store the UART0 reception error status (the lower three bits of the SSR01 register) in the ring buffer. Does nothing if the ring buffer is full.

Argument   None

Return value None

Remarks   None

Bit 0 of the SSR01 register is the overrun error detection flag, bit 1 is the parity error detection flag, and bit 2 is the framing error detection flag. (In this sample program the UART operation is set to “no parity”, so parity error will not be generated.)
## IINTST0

**Outline**  
UART0 transmission end interrupt processing

**Explanation**  
Decrements the number of bytes of transmission data (UASTT) by 1. Transmits the next data if there is any data remaining in the buffer. Disables transmission end interrupts if there is no data remaining.

**Argument**  
None

**Return value**  
None (UASTT indicates the number of remaining data bytes. 0 indicates that the transmission is completed.

**Remarks**  
None

## IINTTM01

**Outline**  
INTTM01 interrupt processing

**Explanation**  
Performs processing according to the number of times an interrupt has occurred.
First time: Counts down the number of captures (LPCOUNT).
Second to fourth times: Adds the capture value to the variable for accumulation.
Fifth time: Adds the capture value to the variable for accumulation and stops TM01.
From the accumulated value, calculates the values to set in the SPS0 and SDR0nH registers, and stores them into variables.

**Argument**  
None

**Return value**  
None

**Remarks**  
Stores the value to set in the SPS0 register into the SPSDATA variable, and the value to set in the SDR0nH register into the DIVDATA variable.

## IINTP0

**Outline**  
INTP0 interrupt processing (Generates an interrupt when the UART start bit is detected)

**Explanation**  
Prepares for baud rate measurement and enables TM01 operation. Disables an INTP0 interrupt.

**Argument**  
None

**Return value**  
None

**Remarks**  
None

## IINTP1

**Outline**  
INTP1 interrupt processing (Generates an interrupt when the switch is pressed)

**Explanation**  
Waits for 10 ms to avoid chattering, then checks again that the switch is pressed, and sets the switch flag. Only when this function is invoked in initialization mode, changes the operating mode to correction mode or transmission mode.

**Argument**  
None

**Return value**  
None

**Remarks**  
None
5.7 Flowcharts

"RET" and "RETI" are used as the termination symbols for functions and interrupt processing, respectively.

5.7.1 Overall Flow

Figure 5.1 shows the overall flow of the sample code described in this application note.

![Flowchart](image-url)
### 5.7.2 I/O Port Settings

Figure 5.2 shows the flowchart for the I/O port settings.

![Flowchart for I/O Port Settings](image)

**SINIPORT**

- Sets analog input shared pins as digital I/Os

- Sets port register

- Treats unused ports

- RET

Each bit in PMC register ← 0

Sets P00 (TXD0), P02, and P04 to 1.

Sets P01 (RXD0) and P03 (INTP1) to 0.

Sets unused ports, where possible, to output.

---

**Note:** Refer to Section 2.3 “Connection of Unused Pins” in the “RL78/G10 User's Manual: Hardware (R01UG0384EJ)” for the setting of unused ports.

**Caution:** Provide proper treatment for unused pins so that their electrical specifications are met. Connect each of any unused input-only ports to V_DD or V_SS via a separate resistor.
5.7.3 Setting of Clock Generation Circuit

Figure 5.3 shows the flowchart for the clock generation circuit.

- **SINICLK**
- **Sets high-speed system clock** \(^{\text{Note}}\)
  - CMC register \(\leftarrow 00000000\text{B}\): Does not use high-speed system clock
  - MSTOP bit of CSC register \(\leftarrow 1\)

- **Sets operation speed mode control register** \(^{\text{Note}}\)
  - WUTMMCK0 bit of OSMC register \(\leftarrow 0\)
    - Stops clock supply from 12-bit interval timer

- **Selects CPU/peripheral hardware clock** \(f_{\text{CLK}}\) \(^{\text{Note}}\)
  - MCM0 bit of CKC register \(\leftarrow 1\)
    - Selects high-speed OCO clock \(f_{\text{IH}}\) as main system clock \(f_{\text{MAIN}}\).

- **Selects frequency of high-speed on-chip oscillator**
  - Bits 2 to 0 of HOCODIV register \(\leftarrow 010\)
    - Selects 10 MHz as HOCO frequency.

**RET**

---

**Note:** Set CMC, CKC, CSC, and OSMC registers only for the 16-pin version of the RL78/G10. (As for the 10-pin version, setting of these registers is unnecessary.)

**Caution:** Refer to Section 4.5.3 “Clock Generation Circuit Setup” in the “RL78/G10 Initialization CC-RLCPU Application Note (R01AN2668EJ)” for the setting of the CPU clock (SINICLK).
5.7.4 SAU Setting

Figure 5.4 and Figure 5.5 show the flowcharts for the SAU settings 1 of 2 and 2 of 2.

---

**Figure 5.4 SAU Setting (1 of 2)**

- **SINISAU**
  - Supplies clock to SAU
  - Selects serial clock
  - Stops operation of channels 0 and 1
  - Sets channel 0 operating mode
  - Sets channel 1 operating mode
  - Sets channel 0 communication operation
  - Sets channel 1 communication operation

---

**SAU Setting**

- **SAU0EN** bit ← 1: Starts clock supply.
  
- **SPS0 register** ← VAL_SPS0
  
- **ST0 register** ← 00000011B
  - Stops operation of channels 0 and 1.

---

**SMR00H register** ← 00H
- Operation clock: CK00 (10 MHz)
- Start trigger source: Software trigger
- **SMR00L register** ← 22H
- Operating mode: UART
- Interrupt: Transfer end interrupt

---

**SMR01H register** ← 01H
- Operation clock: CK00 (10 MHz)
- Start trigger source: Valid edge of RXD0 pin
- Edge to detect: Rising edge
- **SMR01L register** ← 22H
- Operating mode: UART
- Interrupt: Transfer end interrupt

---

**SCR00H register** ← 80H
- Communication operation: Enables transmission only
- Clock and data phase: Type 1
- Parity: No parity
- **SCR00L register** ← 97H
- Transfer sequence: LSB first
- Stop bit: 1 bit
- Data length: 8 bits

---

**SCR01H register** ← 44H
- Communication operation: Enables reception only
- Clock and data phase: Type 1
- Error interrupt: Enabled (INTSRE0)
- Parity: No parity check
- **SCR01L register** ← 97H
- Transfer sequence: LSB first
- Stop bit: 1 bit
- Data length: 8 bits
Figure 5.5 SAU Setting (2 of 2)

Sets transfer speed for channels 0 and 1
- Transfer baud rate setting (transfer clock setting by dividing operation clock (fMCK) by 130)
  - SDR00H register ← VAL_SDR0nH<sup>Note</sup>
  - SDR01H register ← VAL_SDR0nH<sup>Note</sup>

Sets output level to non-inverted
- SOL0 register ← 00H (Output is non-inverted.)

Sets initial value of channel 0 output (TXD0)
- SO0 register ← 01H (Initial value is high.)

Enables channel 0 output
- SOE0 register ← 01H (Enables channel 0 output.)

Sets noise filter
- NFEN0 register ← 01H
  (Turns on RXD0 signal noise filter.)

Clears error flags
- SIR01 register ← 07H

Sets interrupt mask
- STMK0 bit ← 1: Disables transmission end interrupt.
- SRMK0 bit ← 1: Disables reception end interrupt.
- SREMK0 bit ← 1: Disables reception error interrupt.

Sets interrupt priority
- Sets interrupt priority level 3 for INTST0.
- Sets interrupt priority level 2 for INTSR0.
- Sets interrupt priority level 1 for INTSRE0.
  - PR00L register ← 11101111B
  - PR10L register ← 11011111B

Sets ports
- PM0.0 bit ← 0: TxD0 output
- PM0.1 bit ← 1: RxD0 input

Note: Values of VAL_SPS0 in Figure 5.4 and VAL_SDR0nH in Figure 5.5 vary depending on the setting of the initial baud rate with SETTING.inc. For details, see Section エラー!参照元が見つかりません。“エラー!参照元が見つかりません。”.
5.7.5 TAU Setting

Figure 5.6 shows the flowchart for the TAU setting.

![Flowchart for TAU Setting]

- **SINITAU**: Supplies clock to timer array unit
- **Sets TAU operation clock**
- **Sets TAU channel 1 operating mode**
- **Disables TAU channel 1 output**
- **Sets TAU channel 1 output level**
- **Sets TAU channel 1 interrupt**
- **RET**

**Legend**

- TT0.0 bit ← 1
- TAU0EN bit ← 1
- TPS0 register ← 00H
  - CK00: fCLK
  - CK01: fCLK
- TMR01H register ← 01H
  - Sets valid edges for start trigger and capture trigger.
  - 16-bit timer mode
  - Operation clock: CK00
  - Operating mode: Capture mode
  - Valid edge: Falling edge
- TOE01 ← 0: Disables TO01 output
- TO01 ← 0: Sets TO01 output to LOW.
- TMMK00 ← 1: Disables INTTM01 interrupt.
  - TMIF00 ← 0: Clears INTTM01 interrupt request.
5.7.6 Main Processing

Figure 5.7, Figure 5.8, and Figure 5.9 show the flowcharts for the main processing 1 of 3 to 3 of 3.

![Flowchart of Main Processing](image)

- **Initialization SINIMAIN**
  - Initializes global variables and flags, starts operation of peripheral devices, and enables interrupts.
  - **COUNT ← 0**

- **No (MODE is different from CMODEINI.)**

- **Initialization mode?**
  - **Yes**
    - **MODE can be changed by interrupt processing.**
      - When baud rate correction is enabled:
        - IINTP0 interrupt: Changed to correction mode.
        - IINTP1 interrupt: Changed to correction mode.
      - When baud rate correction is disabled:
        - IINTP1 interrupt: Changed to transmission mode.
        - IINTSR0 interrupt: Changed to reception mode.

- **Waits for 1 ms SWAITMS**

- **Counts number of loops**

- **500 loops?**
  - **No (COUNT is not 500.)**
    - **Yes**
      - Inverts LED1 output
      - **COUNT ← 0**

- **Correction enabled?**
  - **No (CFLAGBRC is different from CBRCENABLE.)**

- **Yes**
  - **B**
  - **C**

---

Figure 5.7 Main Processing (1 of 3)
Figure 5.8 Main Processing (2 of 3)

- **B**
  - No (MODE is different from CMODEBRC.)
    - Correction mode?
      - Yes
        - Clears switch flag
      - No
        - Reception mode?
          - Yes
            - Stops UART0 operation
              - SSTOPUART0
            - No
              - Reconfigures baud rate
                - Reconfigures baud rate
                  - SPS0 register ← SPSDATA
                  - SDR00H ← DIVDATA
                  - SDR01H ← DIVDATA
              - Enables UART0 transmission operation
                - SSTARTUART0TX
              - Sends XON
                - SUART0SENDCODE
              - Set to transmission enable state
                - Enables UART0 reception operation
                  - SSTARTUART0RX
          - C
    - D

- **MODE can be changed by interrupt processing.**
  - IINTTM01 interrupt: Changed to reception mode.
  - IINTSR0 interrupt: Changed to transmission mode.
    - -----------------------------------------------
      - FLAGSWON ← 0
    - -----------------------------------------------

- **NO (MODE is different from CMODERX.)**
  - Reception mode?
    - Yes
      - Stops UART0 operation
        - SSTOPUART0
    - No
      - Reconfigures baud rate
        - Reconfigures baud rate
          - SPS0 register ← SPSDATA
          - SDR00H ← DIVDATA
          - SDR01H ← DIVDATA
        - Enables UART0 transmission operation
          - SSTARTUART0TX
        - Sends XON
          - SUART0SENDCODE
        - Set to transmission enable state
          - Enables UART0 reception operation
            - SSTARTUART0RX
    - C

- **After completion of pulse interval measurement, mode is changed to reception mode upon IINTTM01 interrupt.**
  - -----------------------------------------------
    - Stops operation in order to reconfigure baud rate.
  - -----------------------------------------------
    - Reconfigures baud rate.
      - SPS0 register ← SPSDATA
      - SDR00H ← DIVDATA
      - SDR01H ← DIVDATA
    - Resumes UART0 transmission operation.
  - -----------------------------------------------
    - Sends XON to indicate completion of baud rate correction.
  - -----------------------------------------------
    - Set to transmission enable state
      - CMCSTATE ← CSTATEXON
No (FLAGSWON is not 1.)

Switch is ON?
  Yes
  Clears switch flag

Transmission mode?
  No (MODE is the same as CMODETX.)

Transmits data
  SUART0SENDDATA

Reads buffer
  SGETDATASENDXON

Transmission enable state?
  Yes
  Turns off LED2
  Turns on LED2

Cancels HALT mode upon interrupt generation.

Figure 5.9 Main Processing (3 of 3)
5.7.7 Main Initialization Processing

Figure 5.10 shows the flowchart for the main initialization processing.

```
SINIMAIN

<table>
<thead>
<tr>
<th>Turns off LED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initializes ring buffer</td>
</tr>
<tr>
<td>Initializes variable used for transmission function</td>
</tr>
<tr>
<td>Sets number of captures</td>
</tr>
<tr>
<td>Initializes flags and status</td>
</tr>
<tr>
<td>Enables UART0 transmission operation SSTARTUART0TX</td>
</tr>
<tr>
<td>Enables INTP1 interrupt</td>
</tr>
</tbody>
</table>

Correction enabled?  Yes

| Changes RXD0 signal path |
| Enables INTP0 interrupt |

| Enables UART0 reception operation SSTARTUART0RX |

RET

---

Turns off LED1 and LED2.

- Initializes the pointer and counter that control the ring buffer.
  - SETPNT ← 00: Clears set pointer.
  - GETPNT ← 00: Clears read pointer.
  - DATACNT ← 00: Resets the number of data bytes.

- UASTT ← 0: Clears the number of bytes of remaining transmission data.
- FLAGSENDING ← 0: Clears flag to indicate that data is being transmitted.
- Sets the number of times of capture at the pulse width measurement to 5.
- LPCOUNT ← 5.
- MODE ← CMODEINI: Set to initialization mode.
- CMCSTATE ← XON: Set to transmission enable state
- FLAGSWON ← 0: Clears switch-on flag.

- Enables only the transmission operation of UART0.

- Sets to generate an interrupt on pressing of the switch.
  - PIF1 bit ← 0: Clears the interrupt request.
  - PMK1 bit ← 0: Enables the interrupt.

If the baud rate correction is enabled, makes the initial settings.

- ISC register ← 03H
  - ISC1 bit ← 1: Used as TI01 pin.
  - ISC0 bit ← 1: Used as INTP0 pin.

- Sets to generate an interrupt at the falling edge of the RXD0 pin.
  - PIF0 bit ← 0: Clears the interrupt request.
  - PMK0 bit ← 0: Enables the interrupt.

No (CFLAGBRC is different from CBRCENABLE.)
```
5.7.8 Wait Processing (millisecond units)

Figure 5.11 shows the flowchart for the wait processing (millisecond units).

---

**Figure 5.11 Wait Processing (millisecond units)**

```
SWAITMS

Saves HL register on stack

Stores argument on stack

Prepares variables for counting on stack

Is loop completed?

Yes

Clears AX

CLRW AX: AX ← 0

No (AX ≠ 1100)

AX < 1100

Yes

Counts number of loops

[SP+0] ← [SP+0] + 1

Frees the stack area that was being used.

SP ← SP + 4

Restores HL register from stack

POP HL

No (AX is 1100.)

Waits for approximately 1 ms by looping 1100 times.

INCW AX: AX ← AX + 1

No ([SP+0] is the same as [SP+2].)

PUSH HL

Stores the argument (number of loops) on the stack.

PUSH AX

CLRW AX: Clears the variable initial value to 0.

PUSH AX: Secures variables used for counting on the stack.
```
5.7.9 UART0 Transmission Enabling Processing

Figure 5.12 shows the flowchart for the UART0 transmission enabling processing.

Figure 5.12 UART0 Transmission Enabling Processing

SSTARTUART0TX

Enables channel 0 output

Enables UART0 transmission operation

Clears interrupt request flag

RET

SOE0 register ← 01H (enables channel 0 output)

Sets SS0 register to shift UART0 to wait state.
SS00 bit ← 1: Sets to transmission wait state.

Clears UART0 interrupt request flag register.
STIF bit ← 0: Clears transmission end interrupt.
5.7.10 UART0 Reception Enabling Processing

Figure 5.13 shows the flowchart for the UART0 reception enabling processing.

![Flowchart for UART0 Reception Enabling Processing](chart)

- **SSTARTUART0RX**: Enables UART0 reception operation.
  - Sets SS0 register to shift UART0 to wait state. SS01 bit ← 1: Sets to reception wait state.
  - Clears UART0 interrupt request flag register. SRIF bit ← 0: Clears reception end interrupt. SREIF bit ← 0: Clears reception error interrupt.

- **Clears interrupt request**: Enables reception-related interrupts to receive data. SRMK0 bit ← 0: Enables reception end interrupt. SREM0 bit ← 0: Enables reception error interrupt.

---

**Figure 5.13 UART0 Reception Enabling Processing**
5.7.11 UART0 Operation Stop Processing

Figure 5.14 shows the flowchart for the UART0 operation stop processing.

Figure 5.14 UART0 Operation Stop Processing

- SSTOPUART0
  - Disables interrupts
    - SRMK0 bit ← 1
    - SREMk0 bit ← 1
    - STMk0 bit ← 1
  - Clears interrupt request flag
    - SRIF0 bit ← 1
    - SREIF0 bit ← 1
    - STIF0 bit ← 1
  - Stops UART0 operation
    - Stops UART0 operation.
      - ST0 register ← 00000011B
  - Stops target channel output
    - Stops the target channel output.
      - SOE00 bit in SOE0 register ← 0

RET
5.7.12 UART0 Data Transmission Processing

Figure 5.15 shows the flowchart for the UART0 data transmission processing.

![Flowchart for UART0 Data Transmission Processing](image)

Figure 5.15 UART0 Data Transmission Processing
5.7.13 UART0 Control Code Transmission Processing

Figure 5.16 shows the flowchart for the UART0 control code transmission processing.

![Figure 5.16 UART0 Control Code Transmission Processing](image-url)
5.7.14 Ring Buffer Store Processing

Figure 5.17 shows the flowchart for the ring buffer store processing.

![Flowchart for Ring Buffer Store Processing]

- **SPUTDATA**
- **Free space in buffer?**
  - **Yes**
  - **Number of data + 1**
  - **Sets data set pointer**
  - **Stores data**
  - **Changes data set pointer**
  - **RET**
- **No (DATACNT is 16 or more.)**
  - Checks the number of data (DATACNT).
  - If there is no space in the buffer, sets CY flag to end the processing.
  - **Increments the number of data (DATACNT) by 1.**
  - **Sets the data storing pointer to register B.**
    - (Register B ← SETPNT)
  - **Stores data in the address indicated by register B.**
    - (DATABUF[B] ← register A)
  - **Updates the data storing pointer (+1).**
    - INC SETPNT: +1
    - CLR1 SETPNT.4: Rounds off to the lower 4 bits.
5.7.15 Ring Buffer Store and XOFF Transmission Processing

Figure 5.18 shows the flowchart for the ring buffer store and XOFF transmission processing.

```
SPUTDATASENDXOFF

Stores data in ring buffer SPUTDATA

Saves storing result on stack

Transmission enable state?

Yes

Free space in buffer 3 bytes or less?

Yes

Sets to transmission disable state

Sends XOFF SUART0SENDCODE

Saves PSW on the stack.

No (CMCSTATE is different from CSTATEXON.)

No (DATACNT is smaller than (CBUFSIZE – 3).)

Sets to transmission disable state.
CMCSTATE ← CSTATEXOFF
Sends XOFF to disable transmission.

Restores PSW

3 bytes or less?

Restores PSW from the stack.

RET

Figure 5.18 Ring Buffer Store and XOFF Transmission Processing
```
5.7.16 Ring Buffer Read Processing

Figure 5.19 shows the flowchart for the ring buffer read processing.

---

**Figure 5.19 Ring Buffer Read Processing**

1. **SGETDATA**
2. **Received data area in buffer?**
   - **Yes**
   - **Number of data – 1**
   - **Sets reading pointer**
   - **Reads data**
   - **Changes reading pointer**
   - **RET**
   - **No (DATACNT is 0 or less.)**
      - Checks the number of data (DATACNT). If there is no data stored in the ring buffer, sets CY flag to end the processing.
      - **Yes**
      - **Decrements the number of data (DATACNT) by 1.**
      - **Sets the data reading pointer to register C.** (Register C ← GETPNT)
      - **Reads data from the address indicated by register C.** (Register A ← DATABUF[C])
      - **Updates the data reading pointer (−1).**
        - INC GETPNT: +1
        - CLR1 GETPNT.4: Rounds off to the lower 4 bits.

---

Received data area in buffer?
5.7.17 Ring Buffer Read and XON Transmission Processing

Figure 5.20 shows the flowchart for the ring buffer read and XON transmission processing.

---

**Figure 5.20 Ring Buffer Read and XON Transmission Processing**

- **SGETDATASENDXON**
  - Reads ring buffer data **SGETDATA**
  - Saves reading result on stack
  - Transmission disable state?
    - Yes
      - Number of remaining data 3 or less?
        - Yes
          - Sets to transmission enable state.
          - Sends XON **SUART0SENDCODE**
          - Saves register A and PSW on the stack.
        - No
          - Sets to transmission enable state.
          - CMCSTATE ← CSTATEXON
          - Sends XON to enable transmission.
        - No (DATACNT is larger than 3.)
          - Restores register A and PSW from the stack.
    - No (CMCSTATE is different from CSTATEXOFF.)
  - Yes
    - Restores reading result
    - RET
5.7.18 UART0 Reception End Interrupt Processing

Figure 5.21 and Figure 5.22 show the flowcharts for the UART0 reception end interrupt processing 1 of 2 and 2 of 2.

![Flowchart](image-url)

**Figure 5.21 UART0 Reception End Interrupt Processing (1 of 2)**
Figure 5.22 UART0 Reception End Interrupt Processing (2 of 2)
5.7.19 UART0 Reception Error Interrupt Processing

Figure 5.23 shows the flowchart for the UART0 reception error interrupt processing.

![Flowchart]

**Figure 5.23 UART0 Reception Error Interrupt Processing**

- IINTSRE0
  - Saves AX register on stack
  - Clears reception end interrupt request
  - Reads reception data
  - Reads error status
  - Clears error status
  - Reception mode?
    - Yes
      - Free space in buffer?
        - Yes
          - Stores error status
            - SPUTDATASENDXOFF
          - Restores AX register from stack
          - RETI
        - No (DATACNT is not less than 16.)
          - Stores error status in the ring buffer. When the free space in the buffer is 3 bytes or less, sends XOFF.
      - No (MODE ≠ CMODERX)
        - Dummy-reads RXD0 register for the next processing.
        - Reads the UART0 error status (SSR01). (Register A ← SSR01 register & 00000111B)
        - Clears the error status. (SIR01 register ← register A)
    - No (MODE ≠ CMODERX)
      - Reads the UART0 error status (SSR01).
        - (Register A ← SSR01 register & 00000111B)
      - Clears the error status.
        - (SIR01 register ← register A)
  - Yes
    - Reads error status
      - SPUTDATASENDXOFF
      - Restores AX register from stack
      - RETI

PUSH AX

Clears unprocessed reception end interrupt.

Clears unprocessed reception end interrupt.

Dummy-reads RXD0 register for the next processing.

Reads the UART0 error status (SSR01). (Register A ← SSR01 register & 00000111B)

Clears the error status. (SIR01 register ← register A)
5.7.20 UART0 Transmission End Interrupt Processing

Figure 5.24 shows the flowchart for the UART0 transmission end interrupt processing.

![Flowchart of UART0 Transmission End Interrupt Processing](image)

- **IINTST0**: Saves AX and HL registers on stack
- **Enables multiple interrupt function**
- **Number of remaining transmission data – 1**
- **Data transmission all completed?**
  - **Yes**: Disables transmission interrupt, clears flag to indicate that data is being transmitted.
  - **No (UASTT is not 0.)**:
    - **PUSH AX, PUSH HL**
    - **Enables vector interrupts (IE ← 1) to receive the reception end interrupts and reception error interrupt.**
    - **Decrement the number of data to be transmitted (UASTT).**
    - **Reads the next transmission data from memory. (Register A ← ES:[HL])**
    - **Transmits data. (TXD0 register ← Register A)**
    - **Updates the pointer to the address for the next transmission data. (SENDPNT ← SENDPNT + 1)**
    - **READS Pointer**
    - **Restores AX and HL registers from stack**
    - **RETI**
5.7.21 INTTM01 Interrupt Processing

Figure 5.25 and Figure 5.26 show the flowcharts for the INTTM01 interrupt processing 1 of 2 and 2 of 2.

---

**Figure 5.25 INTTM01 Interrupt Processing (1 of 2)**

- **INTTM01**
- **Saves AX register on stack**
- **Valid data?**
  - Yes
  - **Reads captured value.**
  - **Adds captured value**
  - **Counts number of captures**
- **Capture operation completed?**
  - Yes
  - **Disables capture interrupt**
  - **Changes RXD0 signal path**
- **PUSH AX**
  - No (LPCOUNT is 5.)
  - **Overflown?**
    - Yes
    - **Increments the upper digits**
    - **Reads the captured value.**
      - Register X ← register A ← TDR01L register
      - Register A ← TDR01H register
      - **Adds the captured value to the accumulated value.**
      - **CAPTUREL ← CAPTUREL + AX register**
      - **CAPTUREH ← CAPTUREH + carry**
      - **Counts the number of captures.**
        - (LPCOUNT ← LPCOUNT − 1)
  - No (OVF bit is 0.)
    - **Valid data?**
      - Yes
      - **Reads the captured value.**
        - **For the first capture (LPCOUNT=5) does not add.**
        - **If an overflow has occurred by capture, increments the upper 8 bits by 1 (INC CAPTUREH)**
      - No
        - **Capture operation completed?**
          - Yes
          - **Disables INTTM01 interrupt.**
            - TMMK01 bit ← 1: Masks the interrupt.
          - **Separates INTP0 and TI01 from RxD0.**
            - ISC register ← 00H
          - **G**

Calculates bit length (count value)

Initializes SPS set value.

---

**AX > 256?**

---

Yes

Increments SPS set value.

Divides AX register value by 2

AX register – 1

Clears LSB

Sets to reception mode

Restores AX register

---

No (AX is 256 or less.)

---

Divides the accumulated value by 8 and calculates the count value per bit.
AX register ← AX register >> 3
AX register ← AX register + (CAPTUREH << 5)

Initializes the value set in the SPS0 register.
SPSDATA ← 00H

---

Repeats the following until the value (AX) set in the SDR register is no more than 256.
SPSDATA ← SPSDATA + 1
AX register ← AX register >> 1

AX register ← AX register – 1

Register A ← register X & 11111110B
DIVDATA ← register A

After capture completion, sets the mode to reception mode.
MODE ← CMODERX

POP AX

---

Figure 5.26 INTTM01 Interrupt Processing (2 of 2)
5.7.22 INTP0 Interrupt Processing

Figure 5.27 shows the flowchart for the INTP0 interrupt processing.

Figure 5.27 INTP0 Interrupt Processing

- **INTP0**: Sets to correction mode.
  - **MODE**: Sets to correction mode.
  - **CMODEBRC**: Sets to correction mode.
- **Sets to transmission disable state**: Sets to transmission disable state.
  - **CMCSTATE**: Sets to transmission disable state.
  - **CSTATEXOFF**: Sets to transmission disable state.
- **Initializes accumulated capture data**: Initializes accumulated capture data.
  - **CAPTUREL**: Sets the capture correction value and data for rounding off.
  - **CAPTUREH**: Clears the upper digits.
- **Enables TM01 operation**: Enables TM01 operation.
  - **TS01 bit**: Enables TM01 operation.
- **Enables INTTM01 interrupt**: Enables INTTM01 interrupt.
  - **TMIF01 bit**: Clears INTTM01 interrupt request.
  - **TMMK01 bit**: Enables INTTM01 interrupt.
- **Disables INTP0 interrupt**: Disables INTP0 interrupt.
  - **PMK0 bit**: Disables INTP0 interrupt.
- **Turns off LED1**: Turns off LED1.
- **RETI**: Returns from interrupt.
5.7.23 INTP1 Interrupt Processing

Figure 5.28 shows the flowchart for the INTP1 interrupt processing.

```
INTP1

Saves AX register on stack

PUSH AX

Waits for 10 ms
SWAITMS

If the switch is off, ends the processing without performing the following steps.

No (P03 is 1.)

Switch ON?

Yes

Sets switch flag

No (MODE is different from CMODEIN.

Initialization mode?

Yes

Turns on LED1

No (CFLAGBRC is different from CBRCENABLE.)

Correction enabled?

Yes

Sets to transmission mode

Sets to transmission mode.
MODE ← CMODETX

PMK0 bit ← 1: Disables INTP0 interrupt.

Disables INTP0 interrupt

Changes RXD0 signal path

Sets to transmission disable state

Sets to correction mode

Enables UART0 reception operation
SSTARTUART0RX

Transmits correction code
SUAR0SENDCODE

Restores AX register

POP AX

RET1
```
6. Operation Verification

6.1 Transmission Waveform Verification

Figure 6.1 shows the waveform when “ABC” is transmitted, and Figure 6.2 when XON and XOFF are transmitted.

![Waveform when “ABC” is Transmitted](image1)

![Waveforms when XON (Left) and XOFF (Right) are Transmitted](image2)
6.2 Baud Rate Correction Operation Verification

We used an oscilloscope to measure “55H” transmitted from the data transmission side and “11H (XON)” transmitted from the data reception side for each set baud rate value. The waveforms at those times are shown in Figure 6.3 through Figure 6.12. Also, Table 6.1 shows the actually measured baud rates calculated from the waveforms and the errors.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1200</td>
<td>1200</td>
<td>0</td>
<td>1200</td>
<td>0</td>
</tr>
<tr>
<td>9600</td>
<td>9574</td>
<td>-0.27</td>
<td>9574</td>
<td>0</td>
</tr>
<tr>
<td>38400</td>
<td>38460</td>
<td>0.16</td>
<td>38460</td>
<td>0</td>
</tr>
<tr>
<td>115200</td>
<td>116900</td>
<td>1.5</td>
<td>115400</td>
<td>-1.3</td>
</tr>
<tr>
<td>312500</td>
<td>312500</td>
<td>0</td>
<td>312500</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 6.1 Actually Measured Baud Rates and Errors

Figure 6.3 Waveforms on Data Transmission Side (Left) and Data Reception Side (Right) (1200bps)
Figure 6.4 Waveforms on Data Transmission Side (Left) and Data Reception Side (Right) (9600bps)

Figure 6.5 Waveforms on Data Transmission Side (Left) and Data Reception Side (Right) (38400bps)
Figure 6.6 Waveforms on Data Transmission Side (Left) and Data Reception Side (Right) (115200bps)

Figure 6.7 Waveforms on Data Transmission Side (Left) and Data Reception Side (Right) (312500bps)
6.3 Example of Baud Rate Correction Failure

If the baud rate of the data transmission side is too fast then baud rate correction will not be performed correctly. For example, Figure 6.8 shows the waveform when 55H is transmitted at 921,600 bps, which is outside the operation range of this sample program, and Figure 6.9 shows the contents of the variables at that time. Since the 11H (XON) that shows the completion of correction is not transmitted, and the value of MODE is ‘B’, we know that the operation of baud rate correction has not completed.

![Waveform when Baud Rate Correction Fails](image)

**Figure 6.8 Waveform when Baud Rate Correction Fails**

<table>
<thead>
<tr>
<th>Item</th>
<th>Source</th>
<th>Slope</th>
<th>Mode</th>
<th>Combination</th>
</tr>
</thead>
<tbody>
<tr>
<td>[MODE]</td>
<td>0x0000 General-purpose register (2)</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>[LPCOUNT]</td>
<td>0x02 ?(1)</td>
<td>0xfe3a</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>[UARTSTT]</td>
<td>0x0000 ?(1)</td>
<td>0xfe36</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>[SENDPTT]</td>
<td>0x0000 ?(2)</td>
<td>0xfe32</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>[DATAOUT]</td>
<td>n (n..00) ?(1)</td>
<td>0..f..00</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 6.9 Contents of Variables when Baud Rate Correction Fails**
6.4 Transitions between Operating Modes

Figure 6.10 shows the contents of the variables (on the transmission side) after the completion of baud rate correction. Since the “MODE” is ‘T’ we know that it has transitioned to transmission mode.

Figure 6.10 shows the contents of the variables (on the reception side) after the completion of baud rate correction. Since the “MODE” is ‘R’ we know that it has transitioned to reception mode.
6.5 Software Handshake Communication

Figure 6.12 shows the contents of the variables on the data reception side when the three bytes of data (“ABC”) have been transmitted five times from the data transmission side. Since the value of CMCSTATE (which shows whether transmission is currently enabled or disabled) is 0, and the value of DATACNT (which shows the number of bytes in the received data area in the buffer) is 14, we know that it transitioned to transmission enabled state along the way, and the 15th byte of data was not transmitted.

Figure 6.13 shows the contents of the variables on the data reception side when, thereafter, the switch on the data reception side was pressed 11 times. Since the value of CMCSTATE has become 1, we know that the value of DATACNT was once 3, so that it transitioned to transmission enabled state. After that, the transmission of the 15th byte of data that had been interrupted was restarted, so the value of DATACNT is 4.
7. **Sample Code**

The user can get the sample code from the Renesas Electronics website.

8. **Reference Documents**

RL78/G10 User’s Manual: Hardware (R01UH0384JJ)

RL78 Family User’s Manual: Software (R01US0015JJ)

:Get the latest version from the Renesas Electronics website.

Technical Updates/Technical News

:Get the latest information from the Renesas Electronics website.

**Website and Support**

Renesas Electronics Website


Inquiries

[http://www.renesas.com/contact/](http://www.renesas.com/contact/)

All trademarks and registered trademarks are the property of their respective owners.
# Revision History

<table>
<thead>
<tr>
<th>Rev.</th>
<th>Date</th>
<th>Page</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>2016/10/05</td>
<td>-</td>
<td>First edition issued</td>
</tr>
</tbody>
</table>
**General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products**

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. **Handling of Unused Pins**
   - Handle unused pins in accordance with the directions given under Handling of Unused Pins in the manual.
   - The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.

2. **Processing at Power-on**
   - The state of the product is undefined at the moment when power is supplied.
   - The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.
   - In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.
   - In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.

3. **Prohibition of Access to Reserved Addresses**
   - Access to reserved addresses is prohibited.
   - The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.

4. **Clock Signals**
   - After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.
   - When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.

5. **Differences between Products**
   - Before changing from one product to another, i.e. to a product with a different part number, confirm that the change will not lead to problems.
   - The characteristics of Microprocessing unit or Microcontroller unit products in the same group but having a different part number may differ in terms of the internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.
Notice

1. 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.

2. 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.

3. 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.

4. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.

5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality.” The recommended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.

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

   “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-theft systems; anti-crime systems; and safety equipment etc.

   Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantation etc.), or may cause serious property damages (nuclear reactor control systems, military equipment etc.). 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 for which it is not intended. 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 which the product is not intended by Renesas Electronics.

6. 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.

7. 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 malfunction under certain 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 systems manufactured by you.

8. 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.

9. 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. You should not use Renesas Electronics products or 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. When exporting the Renesas Electronics 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.

10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.

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

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

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

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

Renesas Electronics Americas Inc.
2831 Scott Boulevard Santa Clara, CA 95050-2549, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
925 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3
Tel: +1-905-237-2004

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH
Arcadiaallee 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
Rm0909, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100191, P.R.China
Tel: +86-10-6235-1155, Fax: +86-10-6235-1979

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, P. R. China 200333
Tel: +86-21-2226-3888, Fax: +86-21-2226-9999

Renesas Electronics Hong Kong Limited
Unit 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2286-6688, Fax: +852-2386-0022

Renesas Electronics Taiwan Co., Ltd.
13F., No.363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886-2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
8 Bendemeer Road, Unit 105-02 Hysan Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn. Bhda.
Unit 1307, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-6930, Fax: +60-3-7955-9510

Renesas Electronics India Pvt. Ltd.
No 77/3, 109 Fast Road, HAL II Stage, Indianagari, Bangalore, India
Tel: +91-80-67287000, Fax: +91-80-6728777

Renesas Electronics Korea Co., Ltd.
12F., 340 Teheran-ro, Gangnam-Gu, Seoul, 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-6141

© 2016 Renesas Electronics Corporation. All rights reserved.