RL78/G15
Serial Array Unit (UART Communication)

Introduction
This application note explains how to use UART communication through the serial array unit (SAU). ASCII characters transmitted from the device on the opposite side are analyzed to make responses.

Target Device
RL78/G15

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.
1. Specification

1.1 Overview of Specification

In this application note, UART communication is performed through the serial array unit (SAU). ASCII characters transmitted from the device on the opposite side are analyzed to make responses.

Table 1-1 shows the peripheral function to be used and its use. Figures 1-1 and 1-2 illustrate UART communication operation.

Table 1-1 Peripheral Function and Use

<table>
<thead>
<tr>
<th>Peripheral Function</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serial array unit UART0</td>
<td>Perform UART communication using the TxD0 pin (transmission) and the RxD0 pin (reception).</td>
</tr>
</tbody>
</table>

Figure 1-1 UART Reception Timing Chart

- Permit operation of UART
- Permit start of UART communication
- Reception & shift operation
- Reception completed
- ST: Start bit
- P: Parity bit
- SP: Stop bit
Figure 1-2 UART Transmission Timing Chart

- Permit operation of UART
- SS00
- SE00
- SDR00
- TxD0 pin
- Shift register 00
- INTST0
- TSF00

ST: Start bit
P: Parity bit
SP: Stop bit

Transfer completed
1.2 Outline of Operation

This sample code transmits, to the device on the opposite side, the data corresponding to that received from the device. If an error occurs, it transmits to the device the data corresponding to the error. Tables 1-2 and 1-3 show the correspondence between transmit data and receive data.

Table 1-2 Correspondence between Receive Data and Transmit Data

<table>
<thead>
<tr>
<th>Receive Data</th>
<th>Response (Transmit) Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>T (54H)</td>
<td>O (4FH), K (4BH), “CR” (0DH), “LF” (0AH)</td>
</tr>
<tr>
<td>t (74H)</td>
<td>o (6FH), k (6BH), “CR” (0DH), “LF” (0AH)</td>
</tr>
<tr>
<td>Other than above</td>
<td>U (55H), C (43H), “CR” (0DH), “LF” (0AH)</td>
</tr>
</tbody>
</table>

Table 1-3 Correspondence between Error and Transmit Data

<table>
<thead>
<tr>
<th>Error</th>
<th>Response (Transmit) Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parity error</td>
<td>P (50H), E (45H), “CR” (0DH), “LF” (0AH)</td>
</tr>
<tr>
<td>Framing error</td>
<td>F (46H), E (45H), “CR” (0DH), “LF” (0AH)</td>
</tr>
<tr>
<td>Overrun error</td>
<td>O (4FH), E (45H), “CR” (0DH), “LF” (0AH)</td>
</tr>
</tbody>
</table>

The following describes the major settings of the peripheral functions.

(1) Initial settings of the UART

- Use SAU0 channels 0 and 1 as UART.
- Use the P12/TxD0 pin and the P11/RxD0 pin for data output and data input, respectively.
- The data length is 8 bits.
- Set the data transfer direction to LSB first.
- Use even parity as the parity setting.
- Set the receive data level to standard.
- Set the transfer rate to 9600 bps.
- Use reception end interrupt (INTSR0), transmission end interrupt (INTST0), and error interrupt (INTSRE0).
- Set the interrupt priority orders of INTSR0, INTST0, and INTSRE0 to low priority.
2. Operation Confirmation Conditions

The operation of the sample code provided with this application note has been tested under the following conditions.

Table 2-1 Operation Confirmation Conditions

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCU used</td>
<td>RL78/G15 (R5F12068)</td>
</tr>
<tr>
<td>Board used</td>
<td>RL78/G15-20p Fast Prototyping Board (RTK5RLG150CLG000BJ)</td>
</tr>
<tr>
<td>Operating frequency</td>
<td>High-speed on-chip oscillator clock (fIH): 16 MHz</td>
</tr>
<tr>
<td>Operating voltage</td>
<td>5.0 V (can be operated at 2.4 V to 5.5 V))</td>
</tr>
<tr>
<td></td>
<td>SPOR operations (VSPOR)</td>
</tr>
<tr>
<td></td>
<td>At rising edge TYP. 2.57 V (2.44 V to 2.68 V)</td>
</tr>
<tr>
<td></td>
<td>At falling edge TYP. 2.52 V (2.40 V to 2.62 V)</td>
</tr>
<tr>
<td>Integrated development environment (CS+)</td>
<td>CS+ V8.09.00 from Renesas Electronics Corp.</td>
</tr>
<tr>
<td>C compiler (CS+)</td>
<td>CC-RL V1.12.00 from Renesas Electronics Corp.</td>
</tr>
<tr>
<td>Integrated development environment (e2studio)</td>
<td>e2 studio V2023-01 (23.1.0) from Renesas Electronics Corp.</td>
</tr>
<tr>
<td>C compiler (e2studio)</td>
<td>CC-RL V1.12.00 from Renesas Electronics Corp.</td>
</tr>
<tr>
<td>Integrated development environment (IAR)</td>
<td>IAR Embedded Workbench for Renesas RL78 V5.10.1 from IAR Systems Corp.</td>
</tr>
<tr>
<td>C compiler (IAR)</td>
<td>IAR C/C++ Compiler for Renesas RL78 V5.10.1.2667 from IAR Systems Corp.</td>
</tr>
<tr>
<td>Smart Configurator</td>
<td>V.1.5.0 from Renesas Electronics Corp.</td>
</tr>
<tr>
<td>Board support package (r_bsp)</td>
<td>V.1.40 from Renesas Electronics Corp.</td>
</tr>
</tbody>
</table>
3. Hardware Descriptions

3.1 Example of Hardware Configuration

Figure 3-1 shows an example of the hardware configuration used in the application note.

![Figure 3-1 Hardware Configuration](image)

Note 1. This schematic circuit diagram is simplified to show the outline of connections. When creating circuits, design them so that they meet electrical characteristics by properly performing pin processing. (Connect input-only ports to VDD or VSS individually through a resistor.)

3.2 List of Pins to be Used

Table 3-1 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>P04 / TxD0</td>
<td>Output</td>
<td>Data transmission pin</td>
</tr>
<tr>
<td>P05 / RxD0</td>
<td>Input</td>
<td>Data reception pin</td>
</tr>
</tbody>
</table>

Caution In this application note, only the used pins are processed. When actually designing your circuit, make sure the design includes sufficient pin processing and meets electrical characteristic requirements.
4. Software Explanation

4.1 Setting of Option Byte

Table 4-1 shows the option byte settings.

Table 4-1 Option Byte Settings

<table>
<thead>
<tr>
<th>Address</th>
<th>Setting Value</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>000C0H</td>
<td>11101111B</td>
<td>Disables the watchdog timer. (Counting stopped after reset)</td>
</tr>
<tr>
<td>000C1H</td>
<td>11111011B</td>
<td>SPOR detection voltage: Reset mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td>At rising edge TYP. 2.57 V (2.44 V ~ 2.68 V)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>At falling edge TYP. 2.52 V (2.40 V ~ 2.62 V)</td>
</tr>
<tr>
<td>000C2H</td>
<td>11111001.B</td>
<td>High-speed on-chip oscillator clock (fIH): 16 MHz</td>
</tr>
<tr>
<td>000C3H</td>
<td>10000101B</td>
<td>Enables on-chip debugging</td>
</tr>
</tbody>
</table>

4.2 List of Constants

Table 4-2 Constants lists the constants that are used in the sample code.

Table 4-2 Constants

<table>
<thead>
<tr>
<th>Constant Name</th>
<th>Setting Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>s_messageok0[4]</td>
<td>&quot;OK\n\n&quot;</td>
<td>Response message to reception of &quot;T&quot;.</td>
</tr>
<tr>
<td>s_messageok[4]</td>
<td>&quot;ok\n\n&quot;</td>
<td>Response message to reception of &quot;t&quot;.</td>
</tr>
<tr>
<td>s_messageuc[4]</td>
<td>&quot;UC\n\n&quot;</td>
<td>Response message to reception of characters other than &quot;T&quot; or &quot;t&quot;.</td>
</tr>
<tr>
<td>s_messagefe[4]</td>
<td>&quot;FE\n\n&quot;</td>
<td>Response message to a framing error.</td>
</tr>
<tr>
<td>s_messagepe[4]</td>
<td>&quot;PE\n\n&quot;</td>
<td>Response message to a parity error.</td>
</tr>
<tr>
<td>s_messageoe[4]</td>
<td>&quot;OE\n\n&quot;</td>
<td>Response message to an overrun error.</td>
</tr>
</tbody>
</table>
### 4.3 List of Variables

Table 4-3 lists global variables.

<table>
<thead>
<tr>
<th>Type</th>
<th>Variable Name</th>
<th>Contents</th>
<th>Function Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>uint8_t</td>
<td>g uart0rxbuf</td>
<td>Receive data buffer</td>
<td>main()</td>
</tr>
<tr>
<td>uint8_t</td>
<td>g uart0rxerr</td>
<td>Receive error factor</td>
<td>main(), r_Config_UART0_callback_receiveend(), r_Config_UART0_callback_error()</td>
</tr>
<tr>
<td>MD_STATUS</td>
<td>g uart0txend</td>
<td>transmission completion flag</td>
<td>main(), r_Config_UART0_callback_sendend()</td>
</tr>
<tr>
<td>uint8_t</td>
<td>gp uart0 tx address</td>
<td>Transmit data pointer</td>
<td>R_Config_UART0_Send(), r_Config_UART0_interrupt_send()</td>
</tr>
<tr>
<td>uint16_t</td>
<td>g uart0 tx count</td>
<td>Transmit data number counter</td>
<td>R_Config_UART0_Send(), r_Config_UART0_interrupt_send()</td>
</tr>
<tr>
<td>uint8_t</td>
<td>gp uart0 rx address</td>
<td>Receive data pointer</td>
<td>R_Config_UART0_Receive(), r_Config_UART0_interrupt_send()</td>
</tr>
<tr>
<td>uint16_t</td>
<td>g uart0 rx count</td>
<td>Receive data number counter</td>
<td>R_Config_UART0_Receive(), r_Config_UART0_interrupt_receive()</td>
</tr>
<tr>
<td>uint16_t</td>
<td>g uart0 rx length</td>
<td>Receive data number</td>
<td>R_Config_UART0_Receive(), r_Config_UART0_interrupt_receive()</td>
</tr>
</tbody>
</table>

### 4.4 List of Functions

Table 4-4 shows a list of functions.

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Outline</th>
</tr>
</thead>
<tbody>
<tr>
<td>main()</td>
<td>Main processing</td>
</tr>
<tr>
<td>r_Config_UART0_interrupt_send()</td>
<td>UART0 transmission end interrupt handling</td>
</tr>
<tr>
<td>r_Config_UART0_interrupt_receive()</td>
<td>UART0 reception end interrupt handling</td>
</tr>
<tr>
<td>r_Config_UART0_interrupt_error()</td>
<td>UART0 error interrupt handling</td>
</tr>
</tbody>
</table>
4.5 Specification of Functions

The function specifications of the sample code are shown below.

### main()

**Outline**  
Main processing

**Header**  
r_cg_macrodriver.h, r_cg_userdefine.h, Config_UART0.h

**Declaration**  
void main(void);

**Description**  
This function sends the data corresponding to the data received from the counterpart device to the counterpart device.

**Argument**  
None

**Return Value**  
None

### r_Config_UART0_interrupt_send()

**Outline**  
UART0 transmission end interrupt handling

**Header**  
r_cg_macrodriver.h, r_cg_userdefine.h, Config_UART0.h

**Declaration**  
#pragma interrupt r_Config_UART0_interrupt_send(vect=INTST0)

**Description**  
This function starts sending data. It then updates the send data pointer and the number of data to be sent counter.

**Argument**  
None

**Return Value**  
None

### r_Config_UART0_interrupt_receive()

**Outline**  
UART0 reception end interrupt handling

**Header**  
r_cg_macrodriver.h, r_cg_userdefine.h, Config_UART0.h

**Declaration**  
#pragma interrupt r_Config_UART0_interrupt_receive(vect=INTSR0)

**Description**  
This function stores the received data in RAM. It then updates the receive data pointer and the receive data count counter.

**Argument**  
None

**Return Value**  
None

### r_Config_UART0_interrupt_error()

**Outline**  
UART0 error interrupt handling

**Header**  
r_cg_macrodriver.h, r_cg_userdefine.h, Config_UART0.h

**Declaration**  
#pragma interrupt r_Config_UART0_interrupt_error(vect=INTSRE0)

**Description**  
This function stores the error factor in RAM and clears the receive error flag.

**Argument**  
None

**Return Value**  
None
4.6 Flowcharts

4.6.1 Main Processing

Figure 4-1 and 4-2 show flowcharts of the main processing.

Figure 4-1 Main Processing (1/2)
Figure 4-2 Main Processing (2/2)

UART0 reception status initialization function
R_Config_UART0_Receive()

Transmission completed?

YES

What is the receive data?

= “T”

= “t”

Other than “T”, “t”

= “T”

= “t”

UART0 data transmission function
R_Config_UART0_Send()

Argument : “OK”

Argument : “ok”

Argument : “UC”

UART0 data transmission function
R_Config_UART0_Send()

UART0 data transmission function
R_Config_UART0_Send()

Transmission completed?

NO

Enable reception interrupt

SRMK0 bit ← 0
SREM0 bit ← 0

argument : “OK

argument : “ok

argument : “UC”

A

B

C
4.6.2 UART0 Transmission End Interrupt Handling

Figure 4-3 shows the flowchart of the UART0 transmission end interrupt handling.

Figure 4-3 UART0 Transmission End Interrupt Handling

- **r_Config_UART0_interrupt_send()**
  - **Is there any data left?**
    - **NO**
    - **Update the pointer and the counter**
      - **UART0 transmission end processing function**
        - **r_Config_UART0_callback_sendend()**
          - **g_uart0txend = 1**
      - **Return**
    - **YES**
      - **Transmit data**
4.6.3 UART0 Reception End Interrupt Handling

Figure 4-4 shows the flowchart of the UART0 reception end interrupt handling.

Figure 4-4 UART0 Reception End Interrupt Handling

- **r_Config_UART0_interrupt_receive()**: Read receive data
  - Within the specified number of pieces of data?
    - NO: Store the receives data
      - Update the pointer and the counter
    - YES: Data setting completed?
      - NO: UART0 software overrun processing function
        - r_Config_UART0_callback_softwareoverrun()
      - YES: UART0 receive complete processing function
        - r_Config_UART0_callback_receiveend()
        - g_uart0rxerr = 0
- Return

"Within the specified number of pieces of data?" and "Data setting completed?" are decision points in the flowchart.
4.6.4 UART0 Error Interrupt Handling

Figure 4-5 shows the flowchart of the UART0 error interrupt handling.

**Figure 4-5 UART0 Error Interrupt Handling**

```
1. r_Config_UART0_interrupt_error()

2. Read receive data

3. Read the error flag

4. Clear the error flag
   SIR01 register ← (SSR01 & 07H)

5. UART0 receive error classification function
   r_Config_UART0_callback_error()

6. Return
```
5. **Sample Code**

Sample code can be downloaded from the Renesas Electronics website.

6. **Reference Documents**

RL78/G15 User’s Manual: Hardware (R01UH0959)
RL78 family user's manual software (R01US0015)
The latest versions can be downloaded from the Renesas Electronics website.

Technical update
The latest versions can be downloaded from the Renesas Electronics website.

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>Description</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>Jan.20.2023</td>
<td>-</td>
<td>First Edition</td>
<td></td>
</tr>
<tr>
<td>1.01</td>
<td>Apr.5.2023</td>
<td>Page.6</td>
<td>Update the Operation Confirmation Conditions; IAR added</td>
<td></td>
</tr>
</tbody>
</table>