Introduction

This application note describes slave transmission and reception implemented via the serial interface IICA. Using IICA, the single master system described here performs slave operation (address reception, and data transmission and reception).

Target Device

RL78/G13

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 IIC Communication Timing Chart................................................................................................. 5
   1.2 Control of Serial RAM ................................................................................................................ 19
      1.2.1 Command Settings ............................................................................................................. 19
      1.2.2 Continuous Data Writing..................................................................................................... 21
      1.2.3 Continuous Data Reading................................................................................................... 22

2. Operation Check Conditions ............................................................................................................ 23

3. Related Application Note .................................................................................................................. 23

4. Description of the Hardware ............................................................................................................ 24
   4.1 Hardware Configuration Example.............................................................................................. 24
   4.2 List of Pins to be Used ............................................................................................................... 24

5. Description of the Software .............................................................................................................. 25
   5.1 Operation Outline ....................................................................................................................... 25
   5.2 List of Option Byte Settings ........................................................................................................ 26
   5.3 List of Constants ........................................................................................................................ 26
   5.4 List of Variables ........................................................................................................................ 27
   5.5 Variable Specifications ............................................................................................................... 27
   5.6 List of Functions ......................................................................................................................... 28
   5.7 Function Specifications ............................................................................................................... 29
   5.8 Flowcharts.................................................................................................................................. 31
      5.8.1 Initialization Function ........................................................................................................ 31
      5.8.2 System Function ................................................................................................................ 32
      5.8.3 CPU Clock Setup ............................................................................................................... 33
      5.8.4 I/O Port Setup .................................................................................................................... 34
      5.8.5 Serial Interface IICA Setup ................................................................................................. 35
      5.8.6 Main Processing .................................................................................................................. 41
      5.8.7 Main Initial Setting .............................................................................................................. 44
      5.8.8 Internal Address Update Processing .................................................................................. 44
      5.8.9 Command Execution Processing ....................................................................................... 45
      5.8.10 Slave Reception Setting .................................................................................................. 46
      5.8.11 IICA0 Interrupt Processing ............................................................................................. 47
      5.8.12 IICA0 Slave Handler ........................................................................................................ 48

6. Sample Code .................................................................................................................................... 51

7. Documents for Reference ................................................................................................................. 51
1. Specifications

This application note describes a method for using a serial interface IICA for master transmission/reception (address transmission data transmission and reception) in a single-master configuration. It is assumed that each slave has a register to specify an address in the slave.

When the slave address 0b1010000 is specified, the slave becomes capable of transmitting and receiving data.

The following is a summary of the slave specifications assumed in this application note.

- **Slave address:** 0b1010000
- At the specified address, an arbitrary number of data bytes can be read out or written.
- The slave serial RAM area is register addresses 0x80 to 0xFF (128 bytes). Slave operation is specified using the command register at register address 0x00.
- When 0x01 to 0x7F is specified as the register address, NACK is returned and there is disengagement from communication.
- When the register address exceeds 0xFF, the serial RAM area is selected, and only the lower 7 bits of the specified register address are handled as valid.

Caution: This sample code corresponds to the Application Note for the RL78/G13 serial interface IICA (master transmission/reception) (R01AN2759E).

Peripheral functions used and applications are shown in Table 1.1, and the IIC communication is summarized in

Address reception: Each device connected to the IIC bus has a unique address. Each device receives the address of a transfer destination (slave) from the master. That is, it receives one byte of data consisting of 7 bits (indicating the address) and 1 bit (indicating the transfer direction). The slave generates an acknowledgement after receiving one byte of data.

Data transmission/reception: The slave sends/receives data to/from the master after receiving the address.

**Figure 1.1.** IIC communication timing charts appear in the document.
Address reception: Each device connected to the IIC bus has a unique address. Each device receives the address of a transfer destination (slave) from the master. That is, it receives one byte of data consisting of 7 bits (indicating the address) and 1 bit (indicating the transfer direction). The slave generates an acknowledgement after receiving one byte of data.

Data transmission/reception: The slave sends/receives data to/from the master after receiving the address.
1.1 IIC Communication Timing Chart

(1) Master-to-slave communication 1 (start condition – address – data)

Note: The time between the falling edges of SDAA0 and SCLA0 is 4.0 μs or more for the standard mode and 0.6 μs or more for the fast mode.

Figure 1.2  IIC Communication Timing Chart (Master-to-Slave Communication Example) (1/4)
(1) The start condition trigger is set (STT0 = 1) on the master side. Then, the SDAA0 line falls, thereby generating a start condition. Later, when the start condition is detected (STD0 = 1), the master enters a master device communication state (MSTS0 = 1). The SCLA0 line falls at the end of the hold period. This completes preparations for communication.

(2) The values of the address and data direction bit W (transmission) are written to the IICA0 register on the master side. Then, the slave address is transmitted.

(3) If the received address and slave address match \(^\text{Note}\), the slave hardware sends ACK0 to the master. When the ninth clock signal rises, the master detects ACK (ACKD0 = 1).

(4) When the ninth clock signal falls, an address transmission end interrupt (INTIICA0) occurs on the master side. If the addresses match, an address match interrupt (INTIICA0) occurs on the slave side. Both the master and the slave which has the matching address enter a wait state (SCLA0 line: Low) \(^\text{Note}\).

(5) The master writes transmit data to the IICA0 register and cancels the wait.

(6) The slave selects an 8-clock wait (WTIM0 = 0) because it receives data. When the slave cancels the wait (WREL0 = 1), the master starts transferring data to the slave.

Note: When there is a mismatch between a received address and the local address, the slave side does not return ACK to the master side (NACK). Moreover, a slave-side INTIICA0 interrupt (address match interrupt) does not occur, and a wait state is not entered on the slave side. However, on the master side an INTIICA0 interrupt (address transmission complete interrupt) occurs, regardless of whether the result is ACK or NACK.

In the RL78 family, the local address (7 bits) is represented by the upper 7 bits in the SVA0 register. The lowermost bit in the SVA0 register is fixed at 0.
(2) Master-to-slave communication 2 (address – data – data)

Figure 1.3  IIC Communication Timing Chart (Master-to-Slave Communication Example) (2/4)
(3) If the received address and slave address match, the slave hardware sends ACK to the master. When the ninth clock signal rises, the master detects ACK (ACKD0 = 1).

(4) When the ninth clock signal falls, an address transmission end interrupt (INTIICA0) occurs on the master side. If the addresses match, an address match interrupt (INTIICA0) occurs on the slave side. Both the master and the slave which has the matching address enter a wait state (SCLA0 line: Low).

(5) The master writes transmit data to the IICA0 register and cancels the wait.

(6) The slave selects an 8-clock wait (WTIM0 = 0) because it receives data. When the slave cancels the wait (WREL0 = 1), the master starts transferring data to the slave.

(7) When the eighth clock signal falls after the data transfer, the slave hardware generates a wait (SCLA0 line: Low) and a transfer end interrupt (INTIICA0) occurs on the slave side.

(8) When the slave reads the receive data and cancels the wait (WREL0 = 1), the slave sends ACK to the master. When the ninth clock signal rises, the master detects ACK (ACKD0 = 1).

(9) When the ninth clock signal falls, the master generates a wait (SCLA0 line: Low) and a transfer end interrupt (INTIICA0) occurs on the master side.

(10) The master writes transmit data to the IICA0 register and cancels the wait. Then, the master starts transferring data to the slave.
(3) Master-to-slave communication 3 (data – data – stop condition)

<table>
<thead>
<tr>
<th>Master side</th>
<th>Bus line</th>
<th>Slave side</th>
</tr>
</thead>
<tbody>
<tr>
<td>IICA0 register</td>
<td>SCLA0 (bus) (clock line)</td>
<td>IICA0 register</td>
</tr>
<tr>
<td>ACKD0 (ACK detection)</td>
<td>SDAA0 (bus) (data line)</td>
<td>ACKD0 (ACK detection)</td>
</tr>
<tr>
<td>WTIM0 (wait control)</td>
<td></td>
<td>SPD0 (SP detection)</td>
</tr>
<tr>
<td>ACKE0 (ACK control)</td>
<td></td>
<td>WTIM0 (wait control)</td>
</tr>
<tr>
<td>MSTSO (communication state)</td>
<td></td>
<td>ACKE0 (ACK control)</td>
</tr>
<tr>
<td>STT0 (ST trigger)</td>
<td></td>
<td>MSTSO (communication state)</td>
</tr>
<tr>
<td>SPT0 (SP trigger)</td>
<td></td>
<td>WREL0 (wait cancel)</td>
</tr>
<tr>
<td>WREL0 (wait cancel)</td>
<td></td>
<td>INTIICA0 (interrupt)</td>
</tr>
<tr>
<td>INTIICA0 (interrupt)</td>
<td></td>
<td>TRC0 (transmission/reception)</td>
</tr>
<tr>
<td>TRC0 (transmission/reception)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: The time from when SCLA0 rises after a stop condition is issued till when a stop condition is generated is 4.0 μs or more for the standard mode and 0.6 μs or more for the fast mode.

Figure 1.4   IIC Communication Timing Chart (Master-to-Slave Communication Example) (3/4)
(7) When the eighth clock signal falls after the data transfer, the slave hardware generates a wait (SCLA0 line: Low) and a transfer end interrupt (INTIICA0) occurs on the slave side.

(8) When the slave reads the receive data and cancels the wait (WREL0 = 1), the slave sends ACK to the master. When the ninth clock signal rises, the master detects ACK (ACKD0 = 1).

(9) When the ninth clock signal falls, the master generates a wait (SCLA0 line: Low) and an address transmission end interrupt (INTIICA0) occurs on the master side.

(10) The master writes transmit data to the IICA0 register and cancels the wait. Then, the master starts transferring the data to the slave.

(11) When the eighth clock signal falls after the data transfer, the slave hardware generates a wait (SCLA0 line: Low) and a transfer end interrupt (INTIICA0) occurs on the slave side.

(12) When the slave reads the receive data and cancels the wait (WREL0 = 1), the slave sends ACK to the master. When the ninth clock signal rises, the master detects ACK (ACKD0 = 1).

(13) When the ninth clock signal falls, the master generates a wait (SCLA0 line: Low) and a transfer end interrupt (INTIICA0) occurs on the master side.

(14) When the stop condition trigger is set (SPT0 = 1), the SDAA0 line falls and the SCLA0 line rises. Upon the elapse of the stop condition setup time, the SDAA0 line rises, thereby generating a stop condition.

(15) When the stop condition is generated, the slave detects it (SPD0 = 1) and a IICA0 interrupt (stop condition interrupt) occurs on the slave side.
(4) Master-to-slave communication 4 (data – restart condition – address)

Note: The time from when SCLA0 rises after a restart condition is issued till when a start condition is generated is 4.7 μs or more for the standard mode and 0.6 μs or more for the fast mode.

Figure 1.5  IIC Communication Timing Chart (Master-to-Slave Communication Example) (4/4)
(7) When the eighth clock signal falls after the data transfer, the slave hardware generates a wait (SCLA0 line: Low) and a transfer end interrupt (INTIICA0) occurs on the slave side.

(8) The slave reads the receive data and cancels the wait (WREL0 = 1). Then, the slave sends ACK to the master. When the ninth clock signal rises, the master detects ACK (ACKD0 = 1).

(9) When the ninth clock signal falls, the master generates a wait (SCLA0 line: Low) and a transfer end interrupt (INTIICA0) occurs on the master side.

(10) The start condition trigger is set (STT0 = 1) on the master side again. Then, the SCLA0 line rises. Upon the elapse of the restart condition setup time, the SDAA0 line falls, thereby generating a start condition. Later, at the end of the hold period after the start condition is detected (STD0 = 1), the bus clock line falls, thereby completing preparations for communication.

(11) The master writes the slave address to the IICA0 register and starts transferring the address to the slave.
(5) Slave-to-master communication 1 (start condition – address – data)

Note: The time between the falling edges of SDAA0 and SCLA0 is 4.0 μs or more for the standard mode and 0.6 μs or more for the fast mode.

Figure 1.6  IIC Communication Timing Chart (Slave-to-Master Communication Example) (1/3)
(1) The start condition trigger is set (STT0 = 1) on the master side. Then, the SDAA0 line falls, thereby generating a start condition. Later, when the start condition is detected (STD0 = 1), the master enters a master device communication state (MSTS0 = 1). The SCLA0 line falls at the end of the hold period. This completes preparations for communication.

(2) The values of the address and data direction bit R (reception) are written to the IICA0 register on the master side. Then, the slave address is transmitted.

(3) If the received address and slave address match Note, the slave hardware sends ACK to the master. When the ninth clock signal rises, the master detects ACK (ACKD0 = 1).

(4) When the ninth clock signal falls, an address transmission end interrupt (INTIICA0) occurs on the master side. If the addresses match, an address match interrupt (INTIICA0) occurs on the slave side. Both the master and the slave which has the matching address enter a wait state (SCLA0 line: Low).

(5) The master selects an 8-clock wait (WTIM0 = 0) because it receives data.

(6) The slave writes transmit data to the IICA0 register and cancels the wait.

(7) When the master cancels the wait (WREL0 = 1), the slave starts transferring data to the master.

Note: When there is a mismatch between a received address and the local address, the slave side does not return ACK to the master side (NACK). Moreover, a slave-side INTIICA0 interrupt (address match interrupt) does not occur, and a wait state is not entered on the slave side. However, on the master side an INTIICA0 interrupt (address transmission complete interrupt) occurs, regardless of whether the result is ACK or NACK.

In the RL78 family, the local address (7 bits) is represented by the upper 7 bits in the SVA0 register. The lowermost bit in the SVA0 register is fixed at 0.
(6) Slave-to-master communication 2 (address – data – data)

Figure 1.7  IIC Communication Timing Chart (Slave-to-Master Communication Example) (2/3)
(3) If the received address and slave address match, the slave hardware sends ACK to the master. When the ninth clock signal rises, the master detects ACK (ACKD0 = 1).

(4) When the ninth clock signal falls, an address transmission end interrupt (INTIICA0) occurs on the master side. If the addresses match, an address match interrupt (INTIICA0) occurs on the slave side. Both the master and the slave which has the matching address enter a wait state (SCLA0 line: Low).

(5) The master selects an 8-clock wait (WTIM0 = 0) because it receives data.

(6) The slave writes transmit data to the IICA0 register and cancels the wait.

(7) When the master cancels the wait (WREL0 = 1), the slave starts transferring data to the master.

(8) When the eighth clock signal falls, the master generates a wait (SCLA0 line: Low) and a transfer end interrupt (INTIICA0) occurs on the master side. The master hardware sends ACK to the slave.

(9) The master reads the receive data and cancels the wait (WREL0 = 1).

(10) When the ninth clock signal rises, the slave detects ACK (ACKD0 = 1).

(11) When the ninth clock signal falls, the slave generates a wait (SCLA0 line: Low) and a transfer end interrupt (INTIICA0) occurs on the slave side.

(12) The slave writes transmit data to the IICA0 register and cancels the wait. Then, the slave starts transferring data to the master.
(7) Slave-to-master communication 3 (data – data – stop condition)

Note: The time from when SCLA0 rises after a stop condition is issued till when a stop condition is generated is 4.0 μs or more for the standard mode and 0.6 μs or more for the fast mode.

Figure 1.8  IIC Communication Timing Chart (Slave-to-Master Communication Example) (3/3)
(8) When the eighth clock signal falls, the master generates a wait (SCLA0 line: Low) and a transfer end interrupt (INTIICA0) occurs on the master side. The master hardware sends ACK to the slave.

(9) The master reads the receive data and cancels the wait (WREL0 = 1).

(10) When the ninth clock signal rises, the slave detects ACK (ACKD0 = 1).

(11) When the ninth clock signal falls, the slave generates a wait (SCLA0 line: Low) and a transfer end interrupt (INTIICA0) occurs on the slave side.

(12) The slave writes transmit data to the IICA0 register and cancels the wait. Then, the slave starts transferring data to the master.

(13) When the eighth clock signal falls, a transfer end interrupt (INTIICA0) occurs on the master side and the master generates a wait (SCLA0 line: Low). The master hardware sends ACK to the slave.

(14) The master sets a NACK response (ACKE0 = 0) to inform the slave that the master has sent the last data (at the end of communication). Then, the master changes the wait time to 9 clock periods (WTIM0 = 1).

(15) After the master cancels the wait (WREL0 = 1), the slave detects NACK (ACKD0 = 0) at the rising edge of the ninth clock signal.

(16) When the ninth clock signal falls, the master and slave generate a wait (SCLA0 line: Low) and a transfer end interrupt (INTIICA0) occurs on the master and slave sides.

(17) When the master issues a stop condition (SPT0 = 1), the SDAA0 line falls, thereby canceling the wait on the master side. Later, the master waits until the SCLA0 line rises.

(18) The slave cancels the wait (WREL0 = 1) to terminate communication. Then, the SCLA0 line rises.

(19) The master confirms that the SCLA0 line has risen. Upon the elapse of the stop condition setup time after this confirmation, the master makes the SDAA0 line rise and issues a stop condition. When the stop condition is generated, the slave detects the stop condition (SPD0 = 1) and a stop condition interrupt (INTIICA0) occurs on the master and slave sides.
1.2 Control of Serial RAM

1.2.1 Command Settings

In this application note, commands are used to specify slave operations. The command setting sequence is shown in Figure. 1.9, and the command setting timing chart appears in Figure. 1.10.

<table>
<thead>
<tr>
<th>Command Setting Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>ST</td>
</tr>
<tr>
<td>▲</td>
</tr>
</tbody>
</table>

Figure. 1.9  Command Setting Sequence

Upon receiving the slave address 0b1010000 following a start condition (ST), the slave returns ACK\textsuperscript{note} after receiving a transfer direction specification bit. When the transfer direction W\textsubscript{__} is specified, it is determined that the second-next received data will be written to the address specified by the next received data. Upon receiving the register address 0x00, the slave returns ACK, and determines that the next received data is a command. Upon receiving a stop condition (SP), communication is ended.

Upon receiving an address and data, the slave returns ACK. If the address or data cannot be confirmed, the SDA maintains a high level state (NACK response).

Note: In the RL78 family, if the local address and the slave address match, an ACK response is sent automatically.
A list of command functions appears in Table 1.2.

Bit 7 (the MSB) of the data written to register address 0x00 (command register) indicates whether the command is valid or invalid. When bit 7 is 1, the slave judges that the command is valid, and when bit 7 is 0, the slave ignores the command as invalid. Bit 6 indicates whether memory functions are used or not. When memory functions are used, bit 6 is set to 1. Bits 5 to 3 are unused, and so are all set to 0. Bit 2 indicates whether writing is forbidden or not. While bit 2 is set to 1 (from when writing is prohibited until writing is again permitted), upon write data reception the slave sends a NACK response and disengages from communication. Bit 1 indicates whether memory is initialized or not. When bit 1 is set to 1, the slave memory is initialized to initialization data specified by bit 0. After the completion of initialization, the slave sets bit 1 of the command register to 0.

### Table 1.2 Command functions

<table>
<thead>
<tr>
<th>Bit</th>
<th>Meaning</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Command setting</td>
<td>1: command valid, 0: command invalid</td>
</tr>
<tr>
<td>6</td>
<td>Memory function selection</td>
<td>1: use memory functions, 0: do not use memory functions</td>
</tr>
<tr>
<td>5 to 3</td>
<td>Unused</td>
<td>Fixed at 0</td>
</tr>
<tr>
<td>2</td>
<td>Write selection</td>
<td>1: writing forbidden, 0: writing permitted</td>
</tr>
<tr>
<td>1</td>
<td>Initialization selection</td>
<td>1: initialize memory (serial RAM area), 0: do nothing</td>
</tr>
<tr>
<td>0</td>
<td>Initialization data selection</td>
<td>1: value of lowest 7 bits of register address&lt;sup&gt;note&lt;/sup&gt;, 0: 0x00</td>
</tr>
</tbody>
</table>

<sup>Note: At each address in the serial RAM area, the value of the lowest 7 bits of the address is written. For example, at register addresses 0x80, 0x81, 0x82, the respective values 0x00, 0x01, 0x02 are written.</sup>
1.2.2 Continuous Data Writing

Regarding cases in which a register address is specified and data is written continuously to the serial RAM of a slave, the sequence is shown in Figure 1.11, and the timing chart appears in Figure 1.12.

Upon receiving the slave address 0b1010000 following a start condition (ST), the slave returns ACK\textsuperscript{note} after receiving a transfer direction specification bit. When the transfer direction $W$ is specified, it is determined that the second-next received data will be written to the address specified by the next received data. Upon receiving the register address, the slave returns ACK and stores the register address value.

Then, received data is written in sequence from the specified register address (sequential write). Upon receiving a stop condition (SP), communication is ended.

Upon receiving the address and data, the slave returns ACK. If the address or data cannot be confirmed, the SDA maintains a high level state (NACK response).

Note: In the RL78 family, if the local address and the slave address match, an ACK response is sent automatically.

![Figure 1.11 Sequence for Continuous Data Writing by Register Address Specification](image1)
![Figure 1.12 Timing Chart for Continuous Data Writing by Register Address Specification](image2)

The timing chart for data writing when writing is forbidden is shown in Figure 1.13. The slave returns ACK in response to the slave address and a register address, but returns NACK in response to write data, and disengages from communication. The master confirms the NACK response and generates a stop condition, ending communication.

![Figure 1.13 Timing Chart for Data Writing When Writing is Forbidden](image3)
1.2.3 Continuous Data Reading

Regarding cases in which a register address is specified and data is read continuously from a slave, the sequence is shown in Figure 1.14, and the timing chart appears in Figure 1.15.

Upon receiving the slave address 0b1010000 following a start condition (ST), the slave returns ACK\textsuperscript{note} after receiving a transfer direction specification bit. When the transfer direction $W$ is specified, it is determined that the second-next received data will be written to the address specified by the next received data. Upon receiving the register address, the slave returns ACK and stores the register address value. However, before receiving data scheduled for writing, if after a restart condition (ST) the slave address 0b1010000 and the transfer direction $R$ are received, the slave reads data in sequence from the previously specified register address and transmits the data to the master (sequential read).

Upon detecting a NACK from the master, the slave halts data transmission. With a stop condition (SP) from the master, communication is completed.

Upon receiving the address and data, the slave returns ACK. If the address or data cannot be confirmed, the SDA maintains a high level state (NACK response).

---

**Figure 1.14** Sequence for Continuous Data Reading by Register Address Specification

**Figure 1.15** Timing Chart for Continuous Data Reading by Register Address Specification
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/G13 (R5F100LEA)</td>
</tr>
<tr>
<td>Operating frequency</td>
<td>• High-speed on-chip oscillator (HOCO) clock: 32 MHz</td>
</tr>
<tr>
<td></td>
<td>• CPU/peripheral hardware clock: 32 MHz</td>
</tr>
<tr>
<td>Operating voltage</td>
<td>5.0 V (Operation is possible over a voltage range of 2.9 V to 5.5 V.)</td>
</tr>
<tr>
<td></td>
<td>LVD operation (V_{LVD}): Reset mode which uses 2.81 V (2.76 V to 2.87 V)</td>
</tr>
<tr>
<td>Integrated development environment</td>
<td>CS+ V8.01.00 from Renesas Electronics Corp.</td>
</tr>
<tr>
<td>(CS+)</td>
<td></td>
</tr>
<tr>
<td>C compiler (CS+)</td>
<td>CC-RL V1.08.00 from Renesas Electronics Corp.</td>
</tr>
<tr>
<td>Integrated development environment</td>
<td>e² studio V7.3.0 from Renesas Electronics Corp.</td>
</tr>
<tr>
<td>(e² studio)</td>
<td></td>
</tr>
<tr>
<td>C compiler (e² studio)</td>
<td>CC-RL V1.08.00 from Renesas Electronics Corp.</td>
</tr>
</tbody>
</table>

3. Related Application Note

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

- RL78/G13 Initialization (R01AN2575E) Application Note
- RL78/G13 Serial Interface IICA (for Master Transmission/Reception) (R01AN2759E) Application Note
4. Description of the Hardware

4.1 Hardware Configuration Example

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

Figure 4.1  Hardware Configuration

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 proper pin treatment and make sure that the hardware's electrical specifications are met (connect the input-only ports separately to VDD or VSS via a resistor).
2. Connect any pins whose name begins with EVSS to VSS and any pins whose name begins with EVDD to VDD, respectively.
3. VDD must be held at not lower than the reset release voltage (VLVD) that is specified as LVD.

4.2 List of Pins to be Used

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

Table 4.1  Pins to be Used and their Functions

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>I/O</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>P60/SCLA0</td>
<td>Input/Output</td>
<td>IICA0 serial clock I/O pin</td>
</tr>
<tr>
<td>P61/SDAA0</td>
<td>Input/Output</td>
<td>IICA0 serial data transmission/reception pin</td>
</tr>
</tbody>
</table>
5. Description of the Software

5.1 Operation Outline

The sample program covered in this application note provides IICA slave transmission and reception (address reception, and data transmission and reception) through the serial interface IICA.

(1) Initialize serial interface IICA.

<Conditions for setting>
- Select the fast mode as the operation mode.
- Set the transfer clock frequency to 400 kHz.
- To set the slave address to 0b1010000, set 0xA0 to slave address register 0 (SVA0).
- Turn the digital filter on
- Generate an interrupt in response to the ninth clock signal
- Disable stop condition interrupts.
- Use the P60/SCLA0 pin for transfer clock input and the P61/SDAA0 pin for data transmission/reception.

(2) Get the communication buffer (16 bytes) ready for use.

(3) The IICA wakeup function is used to reduce power consumption. A STOP instruction is executed, and an interrupt (INTIICA0) occurring when the local address or an extension code is received is awaited.

(4) When the local address or an extension code is received, wakeup occurs, and data communication is begun. When an extension code has been received, the slave exits from communication and returns to (3).

(5) Register address reception

1. When the received register address is 0x10 to 0x7F, the slave exits from communication. (NACK response)
2. When the register address is 0x00, ACK is returned. The next data (command) is analyzed, and ACK/NACK is returned. Moreover, processing specified by a command is performed.
3. When the register address is 0x80 to 0xFF, the register address is stored. Thereafter, an instruction from the master is awaited.

(6) Data transmission/reception

1. When the transfer direction is R, data is read in sequence from the specified register address, and data that has been read is transmitted from the slave to the master (sequential read) until a NACK response from the master is detected.
2. When the transfer direction is W, received data is written in sequence (sequential write) from the specified register address (serial RAM region).

(7) When a stop condition from the master is detected, the slave exits from communication.

(8) When a start condition from the master is detected, processing is repeated from (4).

(9) The above processing (3) to (7) is repeated.

Caution This sample code is related to the RL78/G13 IICA Master Transmission/Reception (R01AN2759J) Application Note only. This sample code performs actions conforming to instructions from the master rather than actions determined in advance. The conditions for completion of communication are detection of a stop condition, and NACK detection during data transmission. When communication is completed, the next start of communication from the master is awaited. During serial RAM initialization and other communication processing, wait states (SCLA0 pin at low level) are used to synchronize with the master.

Remark Higher-level main processing is performed while checking the value of the variable g_iica0_slave_status_flag indicating the processing state of the IICA0 interrupt.
5.2 List of Option Byte Settings
Table 5.1 summarizes the settings of the option bytes.

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>000C0H/010C0H</td>
<td>01101110B</td>
<td>Disables the watchdog timer. (Stops counting after the release from the reset state.)</td>
</tr>
<tr>
<td>000C1H/010C1H</td>
<td>01111111B</td>
<td>LVD reset mode, 2.81 V (2.76 V to 2.87 V)</td>
</tr>
<tr>
<td>000C2H/010C2H</td>
<td>11101000B</td>
<td>HS mode, HOCO: 32 MHz</td>
</tr>
<tr>
<td>000C3H/010C3H</td>
<td>10000101B</td>
<td>Enables the on-chip debugger.</td>
</tr>
</tbody>
</table>

5.3 List of Constants
Table 5.2 lists the constants that are used in this sample program.

<table>
<thead>
<tr>
<th>Constant</th>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA_LENGTH</td>
<td>0x10</td>
<td>IIC transmit/receive data length</td>
</tr>
<tr>
<td>REGADDR</td>
<td>1</td>
<td>Number of bytes of register address</td>
</tr>
<tr>
<td>ON_COMM</td>
<td>0x00</td>
<td>In communication</td>
</tr>
<tr>
<td>DET_STOP</td>
<td>0x01</td>
<td>Stop condition detected</td>
</tr>
<tr>
<td>TX_END</td>
<td>0x02</td>
<td>NACK detected (data transmission ends)</td>
</tr>
<tr>
<td>TD_REQ</td>
<td>0x04</td>
<td>Data transmission request</td>
</tr>
<tr>
<td>RX_END</td>
<td>0x08</td>
<td>Reception of specified number of data items completed</td>
</tr>
<tr>
<td>DR_REQ</td>
<td>0x10</td>
<td>Received data read request</td>
</tr>
<tr>
<td>AR_END</td>
<td>0x20</td>
<td>Register address (reception) completed</td>
</tr>
<tr>
<td>NOT_SEL</td>
<td>0x80</td>
<td>After restart, address mismatch/extension code received</td>
</tr>
</tbody>
</table>
5.4 List of Variables

Table 5.3 lists the global variables that are used in this sample program.

<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_iica0_slave_status_flag</td>
<td>IICA0 slave flag</td>
<td>R_IICA0_Slave_Receive(), iica0_slave_handler(),main()</td>
</tr>
<tr>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>int16_t</td>
<td>g_iica0_rx_address</td>
<td>IICA0 receive buffer address</td>
<td>R_IICA0_Slave_Receive(), iica0_slave_handler()</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>int16_t</td>
<td>g_iica0_rx_len</td>
<td>IICA0 receive data length</td>
<td>R_IICA0_Slave_Receive(), iica0_slave_handler()</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>int16_t</td>
<td>g_iica0_rx_cnt</td>
<td>IICA0 receive data count</td>
<td>R_IICA0_Slave_Receive(), iica0_slave_handler()</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>int16_t</td>
<td>g_iica0_tx_address</td>
<td>IICA0 transmit buffer address</td>
<td>R_IICA0_Slave_Send(), iica0_slave_handler()</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>g_iica0_tx_cnt</td>
<td>IICA0 transmit data count</td>
<td>R_IICA0_Slave_Send(), iica0_slave_handler()</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>static int8_t</td>
<td>g_rx_data[DATA_LENGTH]</td>
<td>Data receive buffer</td>
<td>R_IICA0_Slave_Receive()</td>
</tr>
<tr>
<td>static int8_t</td>
<td>g_serial_RAM[128]</td>
<td>Serial RAM area</td>
<td>main(), R_Command()</td>
</tr>
<tr>
<td>static int8_t</td>
<td>g_com_stat</td>
<td>Command status</td>
<td>main(), R_Command()</td>
</tr>
<tr>
<td>int16_t</td>
<td>g_reg_addr</td>
<td>Internal address</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Address of serial ram area)</td>
<td></td>
</tr>
</tbody>
</table>

5.5 Variable Specifications

[Variable name] g_iica0_slave_status_flag

<table>
<thead>
<tr>
<th>Summary</th>
<th>IICA0 slave flag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explanation</td>
<td>Indicates the slave operation state</td>
</tr>
</tbody>
</table>

Table 5.4 Variable Specifications

<table>
<thead>
<tr>
<th>Bit</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>1: Received other than slave address 0b1010000 (address mismatch after restart) 0: Received slave address 0b1010000</td>
</tr>
<tr>
<td>6</td>
<td>Fixed to 0</td>
</tr>
<tr>
<td>5</td>
<td>1: Register address set (includes update of same value) 0: Register address not set</td>
</tr>
<tr>
<td>4</td>
<td>1: Received data exceeding specified number of times 0: Did not receive data exceeding specified number of times</td>
</tr>
<tr>
<td>3</td>
<td>1: Received data the specified number of times (also used for register address setting) 0: Did not receive data the specified number of times</td>
</tr>
<tr>
<td>2</td>
<td>1: Data transmission request from master 0: No data transmission request from master</td>
</tr>
<tr>
<td>1</td>
<td>1: NACK detected (master has completed reception of last data) 0: NACK not detected</td>
</tr>
<tr>
<td>0</td>
<td>1: Stop condition detected 0: Stop condition not detected</td>
</tr>
</tbody>
</table>
5.6 List of Functions

Table 5.5 summarizes the functions that are used in this sample program.

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Outline</th>
</tr>
</thead>
<tbody>
<tr>
<td>R_Increment_Address</td>
<td>Internal address update processing</td>
</tr>
<tr>
<td>R_Command</td>
<td>Command execution processing</td>
</tr>
<tr>
<td>R_IICA0_Slave_Receive</td>
<td>Slave reception setting</td>
</tr>
<tr>
<td>r_iica0_interrupt</td>
<td>IICA0 interrupt processing</td>
</tr>
<tr>
<td>iica0_slave_handler</td>
<td>Slave processing within IICA 0 interrupt</td>
</tr>
</tbody>
</table>
5.7 Function Specifications

This section describes the specifications for the functions that are used in this sample program.

---

**[Function Name] R_Increment_Address**

<table>
<thead>
<tr>
<th>Synopsis</th>
<th>Internal address update processing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>-</td>
</tr>
<tr>
<td>Declaration</td>
<td>void R_Increment_Address(void)</td>
</tr>
<tr>
<td>Explanation</td>
<td>Updates the serial RAM address. After 0xFF returns to 0x80.</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Return value</td>
<td>None</td>
</tr>
<tr>
<td>Remarks</td>
<td>None</td>
</tr>
</tbody>
</table>

---

**[Function Name] R_Command**

<table>
<thead>
<tr>
<th>Synopsis</th>
<th>Command execution processing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>-</td>
</tr>
<tr>
<td>Declaration</td>
<td>void R_Command(void)</td>
</tr>
<tr>
<td>Explanation</td>
<td>Performs processing corresponding to the command written to the command register.</td>
</tr>
<tr>
<td>Arguments</td>
<td>None</td>
</tr>
<tr>
<td>Return value</td>
<td>None</td>
</tr>
<tr>
<td>Remarks</td>
<td>None</td>
</tr>
</tbody>
</table>

---

**[Function Name] R_IICA0_Slave_Receive**

<table>
<thead>
<tr>
<th>Synopsis</th>
<th>Slave reception setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>r_cg_serial.h</td>
</tr>
<tr>
<td>Declaration</td>
<td>void R_IICA0_Slave_Receive(uint8_t * const rx_buf and uint16_t rx_num)</td>
</tr>
<tr>
<td>Explanation</td>
<td>Specifies the slave reception mode.</td>
</tr>
<tr>
<td>Arguments</td>
<td>rx_buf</td>
</tr>
<tr>
<td></td>
<td>rx_num</td>
</tr>
<tr>
<td>Return value</td>
<td>None</td>
</tr>
<tr>
<td>Remarks</td>
<td>None</td>
</tr>
</tbody>
</table>

---
### Function: r_iica0_interrupt

**Synopsis**
IICA0 interrupt processing

**Header**
-

**Declaration**
static void __near r_iica0_interrupt(void)

**Explanation**
Interrupt handler for IICA0 interrupt.

**Arguments**
None

**Return value**
None

**Remarks**
In this sample code, in order to meet the specifications, processing to stop the wakeup function operation is added to the beginning of this function, which is generated by the code generator. Even when code is generated, a setting for code generation is used so as to output only an initialization function so as not to exert an influence; interrupt processing is provided in the "r_serial_user.c" file.

### Function: iica0_slave_handler

**Synopsis**
Slave processing within IICA0 interrupt

**Header**
r_cg_serial.h

**Declaration**
static void iica0_slave_handler(void)

**Explanation**
Performs slave transmission and reception during IICA0 interrupt handling.

**Arguments**
None

**Return value**
None

**Remarks**
In this sample code, code generated by the code generator is not used.
5.8 Flowcharts

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

![Overall Flow Diagram](image)

**Figure 5.1** Overall Flow

5.8.1 Initialization Function

Figure 5.2 shows the flowchart for the initialization function.

![Initialization Function Diagram](image)

**Figure 5.2** Initialization Function
5.8.2 System Function

Figure 5.3 shows the flowchart for the system function.

![Flowchart for System Function]

1. **R_Systeminit()**
2. Disuse peripheral I/O redirection function
   - PIOR register ← 00H
3. Set up CPU clock
   - R_CGC_Create()
4. Set up I/O
   - R_PORT_Create()
5. Set up serial interface IICA
   - R_IICA0_Create()
6. Disabled the invalid memory
   - IAWCTL register ← 00H

Figure 5.3 System Function
5.8.3 CPU Clock Setup

Figure 5.4 shows the flowchart for the CPU clock setup.

Caution: For details on the procedure for setting up the CPU clock (R_CGC_Create()), refer to the section entitled “Flowcharts” in RL78/G13 Initialization Application Note (R01AN2575E).
5.8.4 I/O Port Setup

Figure 5.5 shows the flowchart for the I/O port setup.

Note: Refer to the section entitled "Flowcharts" in RL78/G13 Initialization Application Note (R01AN2575E) for the configuration of the unused ports.

Caution: Provide proper treatment for unused pins so that their electrical specifications are observed. Connect each of any unused input-only ports to VDD or VSS via separate resistors.

Figure 5.5 I/O Port Setup
5.8.5 Serial Interface IICA Setup

Figure 5.6 shows the flowchart for the serial interface IICA setup.

```
R_IICA0_Create

Supply clock signals to IICA0

Deactivate IICA0

Disable IICA0 interrupts

Clear IICA0 interrupt request flag

Set IICA0 interrupt priority level to 3

Set up IICA0
  • Fast mode
  • Set IICA low-level width
  • Set IICA high-level width
  • Turn digital filter on
  • Select fCLK/2 as operation clock
  • Set local address to A0H
  • Allow generation of a start condition without detecting stop condition after enabling operation (IICE0 = 1)
  • Disable communication reservation function
  • Prevent interrupt request from being generated by stop condition detection
  • Generate interrupt request upon falling edge of 9th clock signal
  • Enable acknowledgement
  Pull data line low during duration of 9th clock signal

Enable IICA0 interrupts

Enable IICA0

Release data line and clock line

Set up IICA0 pin

return

IICA0EN bit ← 1
IICE0 bit ← 0
IICAMK0 bit ← 1
IICAIF0 bit ← 0
IICAPR10 bit ← 1
IICAPR00 bit ← 1
SMC0 bit ← 1
IICWL0 register ← 15H
IICWH0 register ← 14H
DFC0 bit ← 1
PRS0 bit ← 1
SVA0 register ← A0H
STCEN0 register ← 1
IICRSV0 bit ← 1
SPIE0 bit ← 0
WTIM0 bit ← 1
ACKE0 bit ← 1
IICAMK0 bit ← 0
IICE0 bit ← 1
LREL0 bit ← 1
P6 register ← 00H
PM6 register ← F0H
```
Starting clock signal supply to serial interface IICA0

- Peripheral enable register 0 (PER0)
  Start supplying clock signals to IICA0 by using IIENAEN.

Symbol: PER0

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTCEN</td>
<td>IICA1EN</td>
<td>ADCEN</td>
<td>IICA0EN</td>
<td>SAU1EN</td>
<td>SAU0EN</td>
<td>TAU1EN</td>
<td>TAU0EN</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>1</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

**Bit 4**

<table>
<thead>
<tr>
<th>IICA0EN</th>
<th>Serial interface IIICA0 input clock control</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Stops supply of input clock.</td>
</tr>
<tr>
<td>1</td>
<td>Enables supply of input clock.</td>
</tr>
</tbody>
</table>

Setting up the IICA0 operation mode

- **IICA control register 01 (IICCTL01)**
  - Select an operation clock frequency.
  - Turn the digital filter on.
  - Select the fast mode.
  - Disable the wakeup function.

Symbol: IICCTL01

<table>
<thead>
<tr>
<th>Bit</th>
<th>WUP0</th>
<th>CLD0</th>
<th>DAD0</th>
<th>SMC0</th>
<th>DFC0</th>
<th>0</th>
<th>PRS0</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>0</td>
<td>x</td>
<td>x</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

**Bit 7**

<table>
<thead>
<tr>
<th>WUP0</th>
<th>Address match wakeup control</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Disable the address match wakeup function in STOP mode.</td>
</tr>
<tr>
<td>1</td>
<td>Enable the address match wakeup function in STOP mode.</td>
</tr>
</tbody>
</table>

**Bit 3**

<table>
<thead>
<tr>
<th>SMC0</th>
<th>Operation mode selection</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Standard mode</td>
</tr>
<tr>
<td>1</td>
<td>Fast mode</td>
</tr>
</tbody>
</table>

**Bit 2**

<table>
<thead>
<tr>
<th>DFC0</th>
<th>Digital filter operation control</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Turns the digital filter off.</td>
</tr>
<tr>
<td>1</td>
<td>Turns the digital filter on.</td>
</tr>
</tbody>
</table>

**Bit 0**

<table>
<thead>
<tr>
<th>PRS0</th>
<th>Operation clock frequency selection</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Selects fCLK as the operation clock frequency.</td>
</tr>
<tr>
<td>1</td>
<td>Selects fCLK/2 as the operation clock frequency.</td>
</tr>
</tbody>
</table>

Configuring the transfer clock

- IICA low-level width setting register 0 (IICWL0)
- IICA high-level width setting register 0 (IICWH0)
  Set the low-level width and high-level width of the SCLA0 pin signal.

Symbol: IICWL0

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Symbol: IICWH0

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>


Setting the local address

- Slave address register 0 (SVA0)
  Set the local address.

Symbol: SVA0

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Setting up the IICA operation

- IICA control register 00 (IICCTL00)
  
  Enable I²C operation.
  
  Disable stop condition interrupts.
  
  Set the wait and interrupt request generation timing.
  
  Enable acknowledgement output.

Symbol: IICCTL00

<table>
<thead>
<tr>
<th>Bit</th>
<th>Function Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>IICE0</td>
</tr>
<tr>
<td>6</td>
<td>LREL0</td>
</tr>
<tr>
<td>5</td>
<td>WREL0</td>
</tr>
<tr>
<td>4</td>
<td>SPIE0</td>
</tr>
<tr>
<td>3</td>
<td>WTIM0</td>
</tr>
<tr>
<td>2</td>
<td>ACKE0</td>
</tr>
<tr>
<td>1</td>
<td>STT0</td>
</tr>
<tr>
<td>0</td>
<td>SPT0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>Enables operation.</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>Stops operation. Resets the IICA status register 0 (IICS0). Also stops internal operation.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 6</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>Makes a transition from the current communication state to the standby state. Automatically cleared to 0 after the transition.</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>Normal operation.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 4</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>Enabled</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>Disabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 3</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>Interrupt request is generated at the falling edge of the ninth clock signal. Waits with the clock output remaining at low level, after nine clock signals are output.</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>Interrupt request is generated at the falling edge of the eighth clock signal. Waits with the clock output remaining at low level, after eight clock signals are output.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 2</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>Enables acknowledgements.</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>Disables acknowledgements.</td>
</tr>
</tbody>
</table>

Setting up the IICA pins

- Port register 6 (P6)
- Port mode register 6 (PM6)

Use P60 for SCLA0 and P61 for SDAA0 in output mode.

Symbol: P6

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>P61</td>
<td>P60</td>
</tr>
</tbody>
</table>

Bit 1

<table>
<thead>
<tr>
<th>Bit 1</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Output 0</td>
</tr>
<tr>
<td>1</td>
<td>Output 1</td>
</tr>
</tbody>
</table>

Bit 0

<table>
<thead>
<tr>
<th>Bit 0</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Output 0</td>
</tr>
<tr>
<td>1</td>
<td>Output 1</td>
</tr>
</tbody>
</table>

Symbol: PM6

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>P61</td>
<td>P60</td>
</tr>
</tbody>
</table>

Bit 1

<table>
<thead>
<tr>
<th>Bit 1</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Output mode (output buffer on)</td>
</tr>
<tr>
<td>1</td>
<td>Input mode (output buffer off)</td>
</tr>
</tbody>
</table>

Bit 0

<table>
<thead>
<tr>
<th>Bit 0</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Output mode (output buffer on)</td>
</tr>
<tr>
<td>1</td>
<td>Input mode (output buffer off)</td>
</tr>
</tbody>
</table>

5.8.6 Main Processing

Figure 5.7 through Figure 5.8 shows the flowchart for the main processing.

![Flowchart](image_url)

**Figure 5.7 Main Processing (1/2)**
Figure 5.8   M ain Processing (2/2)

Data transmission processing

If the internal address specifies the serial RAM area, branch to Yes

IICA0 register ← g_serial_RAM[( g_reg_addr & 0x7F )]

Internal address update processing

R_Increment_Address()

Send RAM data

Send command register data

Data reception processing

If the register address is 0x00 (command), branch to Yes

Variable: g_serial_RAM[( g_reg_addr & 0x7F )] ← IICA0 register

WREL0 bit ← 1

Specified data reception completion processing

WREL0 bit ← 1

Variable: g_iica0_slave_status_flag ← 0

Communication completion processing

LREL0 bit ← 1

Clear communication status

Cancel wait for iica0

Cancel wait for IICA0

Exit from the current communications

Address >= 0x80?

Address = 0x00?

Internal address update processing

R_Increment_Address()
Setting up the wakeup function

- IICA control register 01 (IICCTL01)
  Enable the wakeup function.

Symbol: IICCTL01

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>WUP0</th>
<th>CLD0</th>
<th>DAD0</th>
<th>SMC0</th>
<th>DFC0</th>
<th>0</th>
<th>PRS0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>x</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Bit 7

<table>
<thead>
<tr>
<th>WUP0</th>
<th>Address match wakeup control</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Disables the address match wakeup function in STOP mode</td>
</tr>
<tr>
<td>1</td>
<td>Enables the address match wakeup function in STOP mode</td>
</tr>
</tbody>
</table>


Communication termination setting

- IICA control register 00 (IICCTL00)
  Terminate the current communication, and make the system enter a communication standby

Symbol: IICCTL00

<table>
<thead>
<tr>
<th>Bit 6</th>
<th>IICE0</th>
<th>LREL0</th>
<th>WREL0</th>
<th>SPIE0</th>
<th>WTIM0</th>
<th>ACK0E</th>
<th>STT0</th>
<th>SPT0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>x</td>
<td>0/1</td>
<td>0/1</td>
<td>0/1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Bit 6

<table>
<thead>
<tr>
<th>LREL0</th>
<th>Transition from the communication state</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal operation</td>
</tr>
<tr>
<td>1</td>
<td>Makes a transition from the current communication state to the standby state</td>
</tr>
</tbody>
</table>


IIC bus wait cancellation setting

- IICA Control Register00 (IICCTL00)
  End the current wait state and set the communication state

Symbol: IICCTL00

<table>
<thead>
<tr>
<th>Bit 5</th>
<th>IICE0</th>
<th>LREL0</th>
<th>WREL0</th>
<th>SPIE0</th>
<th>WTIM0</th>
<th>ACK0E</th>
<th>STT0</th>
<th>SPT0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0/1</td>
<td>0/1</td>
<td>0/1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Bit 5

<table>
<thead>
<tr>
<th>WREL0</th>
<th>Wait cancellation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Do not cancel wait</td>
</tr>
<tr>
<td>1</td>
<td>Cancel wait. This setting is automatically cleared after wait is canceled.</td>
</tr>
</tbody>
</table>

5.8.7 Main Initial Setting
Figure 5.9 shows the flowchart for the main initial setting.

![Flowchart of Main Initial Setting](image)

5.8.8 Internal Address Update Processing
Figure 5.10 shows the flowchart for the internal address update processing.

![Flowchart of Internal Address Update Processing](image)

- If the internal address is the serial RAM area, the process branches to YES
- Variable: g_reg_addr ← Variable: g_reg_addr + 1
- Variable: g_reg_addr ← Variable: g_reg_addr | 0x80
- Variable: g_reg_addr ← Variable: g_reg_addr & 0xFF
5.8.9 Command Execution Processing

Figure 5.11 shows the flowchart for the command execution processing.

```
R_Command()

Clear initialization data

Store received command

Command valid?

No

Set command status

Operation permission?

No

Operation permission?

Yes

Set command status

Initialization operation?

No

Select initialization data

Serial RAM initialization

addrpt=0, 0x80, +1

Write initialization data

Initialization data update

Serial RAM initialization

Clear command status

return

Variable: data ← 0x00

Variable: work ← IICA0 register
(execute to use bit 0 of the command)

If bit 7 of the command is 1, activate the command

If bit 6 of the command is 1, operation is permitted

Variable: g_com_stat ← Variable: work & 0xC4

If bit 1 of the command is 1, it initializes

variable: work ← Variable: work & 0x01
0x00: Initialize at 0x00
0x01: Initialize with the value of the lower 7 bits of the address of the serial RAM area

Variable: g_serial_RAM[addr_pt] ← Variable: data

Variable: data ← Variable: data + Variable: work

Variable: g_com_stat ← 0x00
```

Figure 5.11 Command Execution Processing
5.8.10 Slave Reception Setting

Figure 5.12 shows the flowchart for the slave reception setting.

```
R_IICA0_Slave_Receive ()

Set the number of bytes to receive
Variable: g_iica0_rx_len ← Second argument

Clear received data byte count
Variable: g_iica0_rx_cnt ← 0

Set receive data buffer
Variable: gp_iica0_rx_address ← First argument

Clear slave status flag
Variable: g_iica0_slave_status_flag ← 0

return
```

Figure 5.12 Slave Reception Setting
5.8.11 IICA0 Interrupt Processing

Figure 5.13 shows the flowchart for IICA0 interrupt processing.

![Flowchart of IICA0 Interrupt Processing](image)

- **r_iica0_interrupt()**
- **Address match in stop mode**
  - **No**
    - **IIC bus is slave device status?**
    - **Yes**
      - **Slave processing within IICA 0 interrupt**
      - **iica0_slave_handler()**
      - **return**
  - **If MSTS0 = 0 (slave state), branch to Yes**
  - **WUP0 bit ← 0**
  - **Enable reading of IICA status register 0 (IICS0)**

Figure 5.13 IICA0 Interrupt Processing
5.8.12 IICA0 Slave Handler

Figure 5.14 through Figure 5.16 show the flowcharts for the IICA0 slave handler.

---

**Stop condition detection processing**

In case of stop condition detection (SPD0 = 1), it is evacuated from communication (LREL0 bit ← 1)

Variable: g_iica0_slave_status_flag ← 0x01

---

In case of start / condition detection (STD0 = 1), branch to Yes (slave address reception processing)

---

If it is an address match (COI 0 = 1), it branches to Yes

---

**Data transmission start processing**

If the transfer direction R (TRC0 = 1), branch to Yes (slave transmission)

If the transfer direction W (TRC0 = 0), branch to No (Register address reception start processing)

WTIM0 bit ← 1

---

If the variable g_iica0_tx_cnt> 0, the process branches to Yes

IICA0 register ← *gp_iica0_tx_address

Pointer: gp_iica0_tx_address ← gp_iica0_tx_address + 1

Counter: g_iica0_tx_cnt ← g_iica0_tx_cnt – 1

---

**Data transmission request processing**

Variable: g_iica0_slave_status_flag ← 0x04 (Data transmission request)
A

Prepare to receive register address

Accept ACK response

Select 8-clock wait

Receive activation

Register address reception start processing

Variable: `g_reg_flag ← REGADDR` (Register address length)
Variable: `g_reg_end_flag ← 0` (Clear setting completion flag)
Variable: `g_reg_addr ← 0x00` (Clear register address)

ACKE0 bit ← 1 (Accept ACK response)

WTIM0 bit ← 0 (Wait for 8-clock)

WREL0 bit ← 1 (Cancel waiting for IIC bus)

B

Exits from the current communications

Set communication status

Communication evacuation processing

LREL0 bit ← 1 (Evacuate from IIC communication)

Variable: `g_iica0_slave_status_flag ← 0x80` (Address mismatch)

C

Slave reception?

No

Yes

Data communication processing

If the transfer direction R (TRC0 = 1), branch to Yes (slave transmission)
If the transfer direction W (TRC0 = 0), branch to No (slave reception)

D

Transmission end?

No

Yes

Exits from the current communications

Set communication status

Transmission completion processing

When NACK detection (ACKD 0 = 0),
transmission is completed

LREL0 bit ← 1 (Evacuate from IIC communication)

Variable: `g_iica0_slave_status_flag ← 0x02` (Detect NACK)

Data transmission processing

IICA0 register ← *gp_iica0_tx_address

Pointer: `gp_iica0_tx_address ← +1` (Update the address)
Variable: `g_iica0_tx_cnt ← -1` (Update the number of remaining data)

E

No

Yes

With remaining transmission data?

No

Yes

Send the next data

Update transmission parameters

Figure 5.15 IICA0 Slave Handler (2/3)
Figure 5.16  IICA0 Slave Handler (3/3)
6.  **Sample Code**

The sample code is available on the Renesas Electronics Website.

7.  **Documents for Reference**

User’s Manual:
- RL78/G13 User's Manual: Hardware (R01UH0146EJ)

The latest version can be downloaded from the Renesas Electronics website.

Technical Updates/Technical News

The latest information can be downloaded from the Renesas Electronics website.

**Website and Support**

Renesas Electronics Website

http://www.renesas.com/index.jsp

Inquiries

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>Apr. 16, 2015</td>
<td>—</td>
<td>First edition issued</td>
</tr>
<tr>
<td>2.00</td>
<td>Apr. 1, 2019</td>
<td></td>
<td>Completely revised</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. Precaution against Electrostatic Discharge (ESD)
   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.

2. Processing at power-on
   The state of the product is undefined at the time 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 time 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 time 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 time when power is supplied until the power reaches the level at which resetting is specified.

3. Input of signal during power-off state
   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.

4. 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 the 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.

5. Clock signals
   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is 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. Additionally, 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.

6. Voltage application waveform at input pin
   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between $V_{IL}$ (Max.) and $V_{IH}$ (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between $V_{IL}$ (Max.) and $V_{IH}$ (Min.).

7. Prohibition of access to reserved addresses
   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.

8. Differences between products
   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of 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 or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.

2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving 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, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.

3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellecutal property rights of Renesas Electronics or others.

4. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.

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

   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

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

6. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.

7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet, Renesas Electronics products are not intended or 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 implantations; etc.), or may cause serious property damage (space system; underwater repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.

8. Renesas Electronics shall not be responsible for the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet, Renesas Electronics products are not intended or 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 implantations; etc.), or may cause serious property damage (space system; underwater repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.

9. Renesas Electronics products and technologies shall 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 shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.

10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.

11. This document shall not be reprinted, 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.

(Note1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

(Rev.4.0-1 November 2017)

Corporate Headquarters
TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks
All trademarks and registered trademarks are the property of their respective owners.