M16C/63, 64A, 64C, 65, 65C, 6C, 5LD, 56D, 5L, 56, 5M, and 57 Groups

Troubleshooting for I^{2}C Mode of Serial Interface in Development

R01AN1895EJ0110
Rev. 1.10
Dec. 1, 2014

Abstract

This application note describes troubles in development and their solutions for the M16C/63, 64A, 64C, 65, 65C, 6C, 5LD, 56D, 5L, 56, 5M, and 57 Groups. The application note is designed to be a guide for solving issues during development.

Products

M16C/63, 64A, 64C, 65, 65C, 6C, 5LD, 56D, 5L, 56, 5M, and 57 Groups

This document uses i, for example UARTi or UiMR register, to indicate serial interface channels which can use special mode 1. UARTi which can use special mode 1 differs depending on the MCU used. Refer to the User's Manual: Hardware for details.
I^{2}C-bus interface with UARTi special mode 1 can achieve the simplified I^{2}C interface by using software to control the function for I^{2}C-bus communication which is added to the UARTi clock synchronous circuit.
This I^{2}C-bus communication is controlled by software and has some limitations on software processing time or timing compared with the I^{2}C-bus interface module controlled by hardware.
When using this application note, careful evaluation is recommended on your system including inter-influences among programs with/without I^{2}C-bus communication.

The table below lists examples of trouble introduced in the application note.

Examples of Trouble Introduced in the Application Note

Trouble Example	Chapter
Common Troubles in Master/Slave Operation of I^{2}C Mode	Chapter 1.
Troubles in Master Operation of I^{2}C Mode	Chapter 2.
Troubles in Slave Operation of I^{2}C Mode	Chapter 3.

How to use this application note

- Trouble examples and their reference sections are provided with hyperlinks in the beginning of each chapter. Click the hyperlink of the appropriate trouble example to see details.
- To go back to the previous page from the linked page, press the ALT + ← keys.
- Reference application notes are introduced for M16C/65C as a representative of products. Visit the Renesas Electronics website to see if the appropriate application note is available for the product used.

Abbreviations and acronyms are used to indicate some circuits, modes, signals, and so on in the application note. The table below lists those abbreviations and acronyms.

Abbreviations and Acronyms Used in the Application Note

Proper Name/Meaning	Abbreviation/Acronym
Clock asynchronous serial I/O mode	UART mode
High-impedance	Hi-Z
High-performance Embedded Workshop	HEW
Device which the M16C MCU communicates with	Target device

Contents

1. Common Troubles in Master/Slave Operation of I^{2}C Mode

Table 1.1 lists Examples of Trouble and Possible Causes. Refer to the sections in the "Refer to" column for detailed explanations and troubleshooting.

Table 1.1 Examples of Trouble and Possible Causes

Section	Trouble	Possible Cause	Refer to
1.1	Unexpected Interrupt Occurred	IR Bit Is Not Set to 0 After the Mode Is Changed	1.1.1
		Interrupt Source is Switched	1.1.2
1.2	Fast-Mode Communication is not Available (1)	UiBRG Count Source is Less Than 10 MHz	1.2.1
		SDAi Digital Delay Value is Inappropriate	1.2.2
1.3	Conditions Cannot be Detected	Input Level Does Not Satisfy the Recommended Operating Conditions	1.3.1
		UiBRG Count Source is Less Than 10 MHz	1.3.2
		SDAi Digital Delay Value is Inappropriate	1.3.3
		UiBRG Register Value is Less Than 03h	1.3.4
1.4	Bit Slippage Occurred	UiBRG Register Value is Less Than 03h	1.4.1
1.5	Expected Data Cannot be Received	UiRB Register Is Read with the Receive Interrupt	1.5.1
		Source of the Transmit Interrupt is Inappropriate	1.5.2
1.6	Overrun Error Occurred	Timing to Read the UiRB Register is not Right	1.6.1
1.7	Communication Becomes Unavailable with the SDAi Pin Being Held Low	Bit Slippage Occurred	1.7.1
1.8	Unexpected Transfer Rate	Synchronous Sampling Clock Delay is not Taken into Account	1.8.1
—	When the Cause of the Issue Cannot be Found/Determined	—	5.

Note:
1. Fast-mode is the I^{2}C-bus standard and the maximum communication rate is 400 kbps.

1.1 Unexpected Interrupt Occurred

Examples:

- When changing the clock phase setting to “with clock delay” by setting the CKPH bit in the UiSMR3 register to 1, a receive interrupt occurred.
- When setting the transmit interrupt source to “transmission completed” by setting the UiIRS bit in the UiC1 register to 1, a transmit interrupt occurred.

1.1.1 IR Bit Is Not Set to 0 After the Mode Is Changed

Check whether the IR bit for each UART interrupt control register is set to 0 (interrupt not requested) after any of the following bits is changed.

- Bits SMD2 to SMD0 in the UiMR register
- IICM bit in the UiSMR register
- IICM2 bit in the UiSMR2 register
- CKPH bit in the UiSMR3 register

When the mode or clock phase is changed, the IR bit in each UART interrupt control register may become 1 (interrupt requested).

Solution:

When any of the following bits is changed, set the IR bit in the UART interrupt control register to 0 (interrupt not requested).

- Bits SMD2 to SMD0 in the UiMR register
- IICM bit in the UiSMR register
- IICM2 bit in the UiSMR2 register
- CKPH bit in the UiSMR3 register

Application Notes:

- I^{2}C-bus Interface Using UARTi Special Mode 1 (Master Transmit/Receive) (REJ05B1422)
- I^{2}C-bus Interface Using UARTi Special Mode 1 (Slave Transmit/Receive) (REJ05B1423)

❖ When you are not sure if this is the cause of the issue:

Run the program and halt it at the code where registers UiMR, UiSMR, UiSMR2, and UiSMR3 are set using a program or debugger such as the E8a emulator. From that point, step through the program to see if an interrupt occurs. If an interrupt occurs where the UiMR, UiSMR, UiSMR2, or UiSMR3 register is set, this item is the cause of the issue.

Also refer to the following section in 4. Analysis Methods:

- 4.3 Examining Code where an Interrupt Occurs

1.1.2 Interrupt Source is Switched

If the interrupt source is switched in the program, check whether the appropriate interrupt is disabled before switching the interrupt source.
When switching the interrupt source, follow the notes below the table “I^{2}C Mode Functions” in the section “Special Mode 1 (I^{2}C Mode)” in the User’s Manual: Hardware. If the procedure is not followed, an unexpected interrupt may occur.

Solution:
When switching the interrupt source, follow the procedure below:
(1) Disable the appropriate interrupt for the source to be switched.
(2) Switch the interrupt source.
(3) Set 0 to the IR bit (interrupt not requested) in the interrupt control register for that interrupt.
(4) Set bits ILVL2 to ILVL0 in the interrupt control register for that interrupt.

Application Notes:
- I^{2}C-bus Interface Using UARTi Special Mode 1 (Master Transmit/Receive) (REJ05B1422)
- I^{2}C-bus Interface Using UARTi Special Mode 1 (Slave Transmit/Receive) (REJ05B1423)

❖ When you are not sure if this is the cause of the issue:
Run the program and halt it at the code where the interrupt source is switched by a program or debugger such as the E8a emulator. From that point, step through the program to see if an interrupt occurs. If an interrupt occurs where the interrupt source is switched, this item is the cause of the issue.
Also refer to the following section in 4. Analysis Methods:
- 4.3 Examining Code where an Interrupt Occurs

1.2 Fast-Mode Communication is not Available

Example:

- When data is received at 400 kbps, the data cannot be received correctly.

1.2.1 UiBRG Count Source is Less Than 10 MHz

When using I2C-bus standard Fast-mode, check whether the UiBRG count source is set to 10 MHz or higher. This is required to have appropriate setup time and hold time for the start and stop conditions. For I2C-bus standard Fast-mode, the minimum values of the setup time and hold time for start and stop conditions are both 600 ns whereas the MCU requires six or more cycles of the UiBRG count source for the setup time and the hold time to detect the start and stop conditions. When the UiBRG count source is 10 MHz, the setup time and hold time become 600 ns. Therefore, if the UiBRG count source is less than 10 MHz, the I2C-bus standard cannot be satisfied.

Solution:

When using I2C-bus standard Fast-mode, set the UiBRG count source to 10 MHz or higher.

Application Notes:

- I2C-bus Interface Using UARTi Special Mode 1 (REJ05B1349)
- I2C-bus Interface Using UARTi Special Mode 1 (Master Transmit/Receive) (REJ05B1422)
- I2C-bus Interface Using UARTi Special Mode 1 (Slave Transmit/Receive) (REJ05B1423)

❖ When you are not sure if this is the cause of the issue:

Using a program or debugger such as the E8a emulator, examine the system clock select bit (CM07 bit in the CM0 register, CM11 bit in the CM1 register, and CM21 bit in the CM2 register) and UiBRG count source select bit (bits CLK1 and CLK0 in the UiC0 register) to see the CPU clock and count source settings. If the UiBRG count source is less than 10 MHz, this item is the cause of the issue.
Also refer to the following sections in 4. Analysis Methods:

- 4.1 Examining a Register Setting
- 4.2 Checking the Frequency of Operating Peripheral Function Clock f1

1.2.2 SDAi Digital Delay Value is Inappropriate

Check whether the SDAi digital delay setup bit (bits DL2 to DL0 in the UiSMR3 register) is set considering the setup time and hold time for the start and stop conditions. For I^{2}C-bus standard Fast-mode, the minimum values of the setup time and hold time for start and stop conditions are both 600 ns. The SDAi digital delay setup bit needs to be set with a value that satisfies the I^{2}C-bus standard.

Solution:
Set the SDAi digital delay setup bit to the value that satisfies the standard of the setup time and hold time for the start and stop conditions.

Application Notes:

- I^{2}C-bus Interface Using UARTi Special Mode 1 (REJ05B1349)
- I^{2}C-bus Interface Using UARTi Special Mode 1 (Master Transmit/Receive) (REJ05B1422)
- I^{2}C-bus Interface Using UARTi Special Mode 1 (Slave Transmit/Receive) (REJ05B1423)

❖ When you are not sure if this is the cause of the issue:
Modify the program to have less cycles of the SDAi output delay set by the SDAi digital delay setup bit, and check if Fast-mode communication is available. If the communication is available, this item may be the cause of the issue.
Examine pins SCLi and SDAi during communication by an oscilloscope. If the SDAi pin changes from high to low after the SCLi pin changes from high to low when outputting a start condition, or the SDAi pin level is changed while the SCLi pin is driven high when outputting data, this item is the cause of the issue.

Also refer to the following sections in 4. Analysis Methods:

- 4.1 Examining a Register Setting
- 4.5 Examining Signals when Communicating in I^{2}C Mode

1.3 Conditions Cannot be Detected

Examples:

- A start condition cannot be detected.
- A restart condition cannot be detected.
- A stop condition cannot be detected.

1.3.1 Input Level Does Not Satisfy the Recommended Operating Conditions

Check whether low/high level of an input signal satisfies the recommended operating conditions.
The standards of low input voltage, high input voltage, and low output voltage are different from the I2C-bus standards. If the recommended operating conditions, which are described in the User's Manual: Hardware, are not satisfied even though the I2C-bus standards are satisfied, each condition may not be recognized correctly.

Solution:
Make sure that the recommended operating conditions for I/O ports which share the pins with SCLi and SDAi are satisfied.

Application Notes:

- I2C-bus Interface Using UARTi Special Mode 1 (REJ05B1349)
- I2C-bus Interface Using UARTi Special Mode 1 (Master Transmit/Receive) (REJ05B1422)
- I2C-bus Interface Using UARTi Special Mode 1 (Slave Transmit/Receive) (REJ05B1423)

❖ When you are not sure if this is the cause of the issue:
Examine the pins SCLi and SDAi during communication by an oscilloscope. If the input voltage does not satisfy the recommended operating conditions of the electrical characteristics in the User's Manual: Hardware, this item may be the cause of the issue.
Also refer to the following section in 4. Analysis Methods:

- 4.5 Examining Signals when Communicating in I2C Mode

1.3.2 UiBRG Count Source is Less Than 10 MHz

When using I^{2}C-bus standard Fast-mode, check whether the UiBRG count source is set to 10 MHz or higher. This is required to have appropriate setup time and hold time for the start and stop conditions. For I^{2}C-bus standard Fast-mode, the minimum values of the setup time and hold time for the start and stop conditions are both 600 ns whereas the MCU requires six or more cycles of the UiBRG count source for the setup time and the hold time to detect the start and stop conditions. When the UiBRG count source is 10 MHz, the setup time and hold time become 600 ns. Therefore if the UiBRG count source is less than 10 MHz, the I^{2}C-bus standard cannot be satisfied.

Solution:
When using I^{2}C-bus standard Fast-mode, set the UiBRG count source to 10 MHz or higher.

Application Notes:
- I^{2}C-bus Interface Using UARTi Special Mode 1 (REJ05B1349)
- I^{2}C-bus Interface Using UARTi Special Mode 1 (Master Transmit/Receive) (REJ05B1422)
- I^{2}C-bus Interface Using UARTi Special Mode 1 (Slave Transmit/Receive) (REJ05B1423)

❖ When you are not sure if this is the cause of the issue:
Using a program or debugger such as the E8a emulator, examine the system clock select bit (CM07 bit in the CM0 register, CM11 bit in the CM1 register, and CM21 bit in the CM2 register) and UiBRG count source select bit (bits CLK1 and CLK0 in the UiC0 register) to see the CPU clock and count source settings. If the UiBRG count source is less than 10 MHz, this item is the cause of the issue.
Also refer to the following sections in 4. Analysis Methods:
- 4.1 Examining a Register Setting
- 4.2 Checking the Frequency of Operating Peripheral Function Clock f1

1.3.3 SDAi Digital Delay Value is Inappropriate

Check whether the SDAi digital delay setup bit (bits DL2 to DL0 in the UiSMR3 register) is set considering the setup time and hold time for the start and stop conditions. For I^{2}C-bus standard Fast-mode, the minimum values of the setup time and hold time are both 600 ns. The SDAi digital delay setup bit needs to be set with a value that satisfies the I^{2}C-bus standard.

Solution:
Set the SDAi digital delay setup bit to the value that satisfies the standard of the setup time and hold time for the start and stop conditions.

Application Notes:
- I^{2}C-bus Interface Using UARTi Special Mode 1 (REJ05B1349)
- I^{2}C-bus Interface Using UARTi Special Mode 1 (Master Transmit/Receive) (REJ05B1422)
- I^{2}C-bus Interface Using UARTi Special Mode 1 (Slave Transmit/Receive) (REJ05B1423)

❖ When you are not sure if this is the cause of the issue:
Modify the program to have less cycles of the SDAi output delay set by the SDAi digital delay setup bit, and check if Fast-mode communication is available. If the communication is available, this item is the cause of the issue.
Examine pins SCLi and SDAi during communication by an oscilloscope. If the SDAi pin changes from high to low after the SCLi pin changes from high to low when outputting a start condition, this item is the cause of the issue.
Also refer to the following sections in 4. Analysis Methods:
- 4.1 Examining a Register Setting
- 4.5 Examining Signals when Communicating in I^{2}C Mode

1.3.4 UiBRG Register Value is Less Than 03h

Check whether the value in the UiBRG register is less than 03h. It takes up to three cycles of the UiBRG count source until the internal circuit recognizes the SCL clock level. If a value less than 03h is set to the UiBRG register, the MCU may not be able to recognize the condition waveform generated by itself.

Solution:

When using I2C mode, set a value equal to or greater than 03h to the UiBRG register.

Application Notes:

- I2C-bus Interface Using UARTi Special Mode 1 (REJ05B1349)
- I2C-bus Interface Using UARTi Special Mode 1 (Master Transmit/Receive) (REJ05B1422)
- I2C-bus Interface Using UARTi Special Mode 1 (Slave Transmit/Receive) (REJ05B1423)

❖ When you are not sure if this is the cause of the issue:

Check the setting value in the UiBRG register during communication by a program or debugger such as the E8a emulator. If the value in the UiBRG register is less than 03h, this item is the cause of the issue.

Also refer to the following section in 4. Analysis Methods:

- 4.1 Examining a Register Setting

1.4 Bit Slippage Occurred

Example:

- A receive interrupt does not occur even though the data is received.

1.4.1 UiBRG Register Value is Less Than 03h

Check whether the value in the UiBRG register is less than 03h. It takes up to three cycles of the UiBRG count source until the internal circuit recognizes the SCL clock level. Therefore a value equal to or greater than 03h needs to be set to the UiBRG register. Then the connectable I2C-bus bit rate becomes less than or equal to one-third of the UiBRG count source speed. If a value between 00h to 02h is set to the UiBRG register, bit slippage may occur.

Solution:

When using I2C mode, set a value equal to or greater than 03h to the UiBRG register.

Application Notes:

- I2C-bus Interface Using UARTi Special Mode 1 (REJ05B1349)
- I2C-bus Interface Using UARTi Special Mode 1 (Master Transmit/Receive) (REJ05B1422)
- I2C-bus Interface Using UARTi Special Mode 1 (Slave Transmit/Receive) (REJ05B1423)

❖ When you are not sure if this is the cause of the issue:

Check the setting value in the UiBRG register during communication by a program or debugger such as the E8a emulator. If the value in the UiBRG register is less than 03h, this item is the cause of the issue.

Also refer to the following section in 4. Analysis Methods:

- 4.1 Examining a Register Setting

1.5 Expected Data Cannot be Received

Example:

- The received data read with the receive interrupt is not an expected data.

1.5.1 UiRB Register Is Read with the Receive Interrupt

Check whether the UiRB register is read with the receive interrupt by setting the IICM2 bit in the UiSMR2 register to 1 (use transmit/receive interrupt).

With I2C mode, when selecting UART transmit/receive interrupt as the interrupt source, both the transmit and receive interrupts occur on the data reception. Interrupt occurrence timing and the data order to store in the UiRB register are different between the transmit and receive interrupts.

The receive interrupt occurs before transmitting the ACK/NACK bit after 8-bit data is received.

Data order at the receive interrupt:

b15	...	b9	b8	b7							b0
			D0	—	D7	D6	D5	D4	D3	D2	D1

The transmit interrupt occurs during transmitting 8-bit data and the ACK/NACK bit.

Data order at the transmit interrupt:

b15	...	b9	b8	b7							b0
			D8	D7	D6	D5	D4	D3	D2	D1	D0

Solution:

When reading the UiRB register with the UART receive interrupt by setting the IICM2 bit to 1, process the data as required. Otherwise read the register with the transmit interrupt.

Application Notes:

- I2C-bus Interface Using UARTi Special Mode 1 (REJ05B1349)
- I2C-bus Interface Using UARTi Special Mode 1 (Master Transmit/Receive) (REJ05B1422)
- I2C-bus Interface Using UARTi Special Mode 1 (Slave Transmit/Receive) (REJ05B1423)

❖ When you are not sure if this is the cause of the issue:

Check the timing to read the received data. If the received data is read with the receive interrupt, this item may be the cause of the issue.

Also refer to the following section in 4. Analysis Methods:

- 4.5 Examining Signals when Communicating in I2C Mode

1.5.2 Source of the Transmit Interrupt is Inappropriate

Check whether the UiIRS bit in the UiC1 register is set to 1 (transmission completed). When using I^{2}C mode, the UiIRS bit must be set to 1.

Solution:
Set the UiIRS bit in the UiC1 register to 1.

Application Notes:

- I^{2}C-bus Interface Using UARTi Special Mode 1 (REJ05B1349)
- I^{2}C-bus Interface Using UARTi Special Mode 1 (Master Transmit/Receive) (REJ05B1422)
- I^{2}C-bus Interface Using UARTi Special Mode 1 (Slave Transmit/Receive) (REJ05B1423)

❖ When you are not sure if this is the cause of the issue:
Check whether the UiIRS bit is set to 1 by a program or debugger such as the E8a emulator. If not, this item is the cause of the issue.
Also refer to the following sections in 4. Analysis Methods:

- 4.1 Examining a Register Setting
- 4.5 Examining Signals when Communicating in I^{2}C Mode

1.6 Overrun Error Occurred

Example:
- An overrun error occurs when receiving data continuously.

1.6.1 Timing to Read the UiRB Register is not Right

Check whether the next data is read before completing read processing for the previous data in the UiRB register due to slow operation of the CPU clock or an interrupt generated for another function. If so, next data reception may be started and the eighth bit of the next data may be received before reading the UiRB register.

Solution:
For the operation at data reception, design your system to write the UiTB register after reading the data in the UiRB register.
Alternatively, enable the SCL wait auto insert function (set the SWC bit in the UiSMR2 to 1 or set the SWC9 bit in the UiSMR4 register to 1). Enabling this function holds the SCLi pin low after the eighth bit is received when setting the SWC bit, and after the ninth bit is received when setting the SWC9 bit.

Application Notes:
- I2C-bus Interface Using UARTi Special Mode 1 (REJ05B1349)
- I2C-bus Interface Using UARTi Special Mode 1 (Master Transmit/Receive) (REJ05B1422)
- I2C-bus Interface Using UARTi Special Mode 1 (Slave Transmit/Receive) (REJ05B1423)

❖ When you are not sure if this is the cause of the issue:
Add a program to invert a test port immediately before reading the received data. Then examine the test port and the RXDi pin using an oscilloscope. If the timing that the received data is input to the RXDi pin comes faster than the timing that the test port is inverted, this item may be the cause of the issue.

1.7 Communication Becomes Unavailable with the SDAi Pin Being Held Low

Example:

- Communication is stopped during transmission with the SDAi pin being held low.

1.7.1 Bit Slippage Occurred

Check whether noise occurs on the SCLi pin or a signal which does not satisfy the standard is input to the SCLi pin. If so, clocks may not match between the master and slave devices (bit slippage).
Noise or a substandard signal on the SCLi pin may or may not be recognized as a clock due to differences of the characteristics such as VIH/VIL or noise cancelling ability between the master and slave devices, and this may cause bit slippage.
If bit slippage occurs, the SDAi pin may be held low.

Solution:

If the SDAi pin is held low by the target device, disable the serial interface once and switch pins SCLi and SDAi to programmable I/O ports. Then output a pseudo-clock (Hi-Z and low output) from the port corresponding to the SCLi pin to examine whether the SDAi pin is released.
If the SDAi pin is not released by a pseudo-clock output, output a pseudo-clock repeatedly until the SDAi pin is released.

After the SDAi pin is confirmed to be released, set the mode to I^{2}C mode again, issue a start condition and stop condition, and terminate the communication once.

As most of the slave devices initialize (reset) the I^{2}C-bus by a start condition and stop condition, perform processing for restarting communication.
If the manual for your slave device describes the initialization (reset) method, initialize (reset) the slave device following the manual.

Application Notes:

- I^{2}C-bus Interface Using UARTi Special Mode 1 (REJ05B1349)
- I^{2}C-bus Interface Using UARTi Special Mode 1 (Master Transmit/Receive) (REJ05B1422)
- I^{2}C-bus Interface Using UARTi Special Mode 1 (Slave Transmit/Receive) (REJ05B1423)

❖ When you are not sure if this is the cause of the issue:
When bit slippage occurs, data from the master or slave device is not output completely and the data output may be stopped during the operation (0 is output).
Reset the M16C MCU to check whether the SDAi pin is released from being held low. If the SDAi pin is released, the M16C MCU may cause the SDAi pin being held low. If the SDAi pin is not released, the slave device may cause the SDAi pin being held low.
When the M16C MCU is possibly the cause, halt the program immediately before the transmit condition is satisfied (e.g. before setting transmit data) by modifying the program or using the E8a emulator, and examine the state of the CLKi pin with an oscilloscope. If the external clock is low while the CKPOL bit in the UiC0 register is 0, or if the external clock is high while the CKPOL bit is 1, this item may be the cause of the issue.
Also refer to the following sections in 4. Analysis Methods:

- 4.4 Determining the Device that Holds Pins SDAi and SCLi Low
- 4.5 Examining Signals when Communicating in I^{2}C Mode

1.8 Unexpected Transfer Rate

Example:

• The bit rate is slower than the expected rate even though it is set taking into account the count source.

1.8.1 Synchronous Sampling Clock Delay is not Taken into Account

Check whether the synchronous sampling clock delay is taken into account when the CSC bit in the UiSMR2 register is set to 1 (clock synchronization enabled).
When the CSC bit is set to 1, the sampling clock delay of “noise filter width + 1 to 1.5 cycles of the UiBRG count source” is introduced. Then there is also a delay for recognition of a high level of the SCL clock. As the result, the high width of the SCL clock is extended. Therefore, the actual SCL clock becomes slower than SCL clock bit rate setting.

Solution:

Specify the bit rate taking into account the sampling clock delay of “approx. 100 ns (1) of the noise filter width + 1 to 1.5 cycles of the UiBRG count source”.

The bit rate can be calculated as follows:

Bit rate = 1 ÷ (((2 × (n + 1)) ÷ fj) + 100 ns + (1.5 cycles × 1 ÷ fj) + tF + tR)

fj: UiBRG count source
n: Setting value in the UiBRG register
tF: SCL clock fall time
tR: SCL clock rise time

Note:

1. Maximum 200 ns

Application Notes:

- I2C-bus Interface Using UARTi Special Mode 1 (REJ05B1349)
- I2C-bus Interface Using UARTi Special Mode 1 (Master Transmit/Receive) (REJ05B1422)
- I2C-bus Interface Using UARTi Special Mode 1 (Slave Transmit/Receive) (REJ05B1423)

❖ When you are not sure if this is the cause of the issue:

Check the setting value of the CSC bit in the UiSMR2 register by a program or debugger such as the E8a emulator. If the CSC bit is 1 and the bit rate is set not taking into account the sampling clock delay of 1 to 1.5 cycles of the UiBRG count source, this item may be the cause of the issue.
Also refer to the following sections in 4. Analysis Methods:

- 4.1 Examining a Register Setting
- 4.2 Checking the Frequency of Operating Peripheral Function Clock f1

2. Troubles in Master Operation of I2C Mode

Table 2.1 lists Examples of Trouble and Possible Causes. Refer to the sections in the "Refer to" column for detailed explanations and troubleshooting.

Table 2.1 Examples of Trouble and Possible Causes

Section	Trouble	Possible Cause	Refer to
2.1	Conditions Cannot be Generated Correctly	Condition Generate Bits are not Set in I2C Mode	2.1.1
		Timing to Select the Condition Generate Circuit is not Right	2.1.2
		Multiple Conditions are Generated Simultaneously	2.1.3
		Timing to Set the Condition Generate Bit is not Right	2.1.4
		SDAi Digital Delay Length is Not an Appropriate Value	2.1.5
		Pins SCLi and SDAi are not Pulled Up	2.1.6
		Pull-up Resistance Value is not Appropriate Value	2.1.7
2.2	SDAi Pin Becomes Low/Low Pulse is Output	Initial Value of the SDAi Output is not Set to High	2.2.1
		Ports Corresponding to Pins SCLi and SDAi are Not Set to High	2.2.2
		ACK is Output	2.2.3
		ACK/NACK Setting During Idle State is Not Appropriate with the Serial I/O Circuit Selected	2.2.4
		Serial I/O Circuit is Not Selected After Generating the Condition	2.2.5
2.3	SCLi Pin Becomes Low/Low Pulse is Output	Ports Corresponding to Pins SCLi/SDAi are Not Set to High	2.3.1
		WAIT-State is not Cleared when Using the SCL WAIT Function	2.3.2
		SWC2 Bit in the UiSMR2 Register is Set to 1	2.3.3
		Clock Phase is Changed Before Generating the Start Condition	2.3.4
		Serial I/O Circuit is Not Selected After Generating the Condition	2.3.5
2.4	Data is Not Output as Expected	Clock Synchronization is Not Enabled	2.4.1
		ACK is Output	2.4.2
		Serial I/O Circuit is Not Selected After Generating the Condition	2.4.3
		Next Data is Set While the Transmit Buffer is Full	2.4.4
		Transmit Interrupt Source is Not "Transmission Completed"	2.4.5
		Arbitration Lost Has Been Detected	2.4.6
2.5	Fast-Mode Communication is not Available	UiBRG Count Source is Less Than 10 MHz	2.5.1
		Communication Bit Rate is Set to 400 kbps	2.5.2
2.6	ACK is Always Returned	Ninth Bit of the Transmit Data is Not Set to 1	2.6.1
2.7	Unexpected Arbitration Lost Occurs	Arbitration Lost is Detected when Receiving the ACK/NACK Bit	2.7.1
		Arbitration Lost is Detected when Receiving the Data	2.7.2
—	When the Cause of the Issue Cannot be Found/Determined	—	5.

2.1 Conditions Cannot be Generated Correctly

Example:

- The start condition is not output even though processing for generating a start condition is performed.

2.1.1 Condition Generate Bits are not Set in I2C Mode

Check whether condition generate bits are set in I2C mode (IICM bit in the UiSMR register is 1).
Condition generate bits (bits STAREQ, RSTAREQ, and STPREQ in the UiSMR4 register) can be set to 1 when in I2C mode. Do not write 1 to these bits in any other modes.

Solution:

When generating conditions, set the mode to I2C mode, and then set each condition generate bit (bits STAREQ, RSTAREQ, and STPREQ in the UiSMR4 register) to 1.

Application Notes:

- I2C-bus Interface Using UARTi Special Mode 1 (REJ05B1349)
- I2C-bus Interface Using UARTi Special Mode 1 (Master Transmit/Receive) (REJ05B1422)

❖ When you are not sure if this is the cause of the issue:

Check the program to see if I2C mode is selected before each condition generate bit is set. If not, this item may be the cause of the issue.
Also refer to the following section in 4. Analysis Methods:

- 4.1 Examining a Register Setting

2.1.2 Timing to Select the Condition Generate Circuit is not Right

Check whether the condition generate circuit is selected (STSPSEL bit in the UiSMR4 register is set to 1) after setting the condition generate bits (bits STAREQ, RSTAREQ, and STPREQ in the UiSMR4 register) to 1.
When generating conditions automatically by the hardware, one of the condition generate bits needs to be set to 1 before setting the STSPSEL bit in the UiSMR4 register to 1.
When not generating conditions automatically by the hardware, conditions need to be generated by a program using ports while the STSPSEL bit is 0.

Solution:

When the STSPSEL bit is set to 0, generate the start condition and stop condition by a program using ports.

Application Notes:

- I2C-bus Interface Using UARTi Special Mode 1 (REJ05B1349)
- I2C-bus Interface Using UARTi Special Mode 1 (Master Transmit/Receive) (REJ05B1422)

❖ When you are not sure if this is the cause of the issue:

Check the program to see if each condition generate bit is set before selecting the condition generate circuit. If not, this item may be the cause of the issue.
Also refer to the following section in 4. Analysis Methods:

- 4.1 Examining a Register Setting

2.1.3 Multiple Conditions are Generated Simultaneously

Check whether multiple condition generate bits (bits STAREQ, RSTAREQ, and STPREQ in the UiSMR4 register) are set to 1 simultaneously.
Multiple condition generate bits cannot be set to 1 simultaneously.

Solution:
Do not select multiple condition generate bits simultaneously. When multiple conditions need to be generated, generate each condition separately.

Application Notes:
- I2C-bus Interface Using UARTi Special Mode 1 (REJ05B1349)
- I2C-bus Interface Using UARTi Special Mode 1 (Master Transmit/Receive) (REJ05B1422)

❖ When you are not sure if this is the cause of the issue:
When generating conditions, check the program to see if the condition generate circuit is selected while multiple condition generate bits are set to 1. In that case this item may be the cause of the issue.
Also refer to the following section in 4. Analysis Methods:
- 4.1 Examining a Register Setting

2.1.4 Timing to Set the Condition Generate Bit is not Right

When setting the condition generate bit after setting the STSPSEL bit in the UiSMR4 register to 0, check whether wait processing for 1/2 cycle of the SCL clock or longer is performed after the STSPSEL bit is set to 0.
For example, when generating the stop condition and then generating the next start condition, set the STSPSEL bit back to 0 (select serial I/O circuit) once and wait 1/2 cycle of the SCL clock or longer before setting the start condition generate bit.
If each condition generate bit (bits STAREQ, RSTAREQ, and STPREQ in the UiSMR4 register) is set from 0 to 1 within 1/2 cycle of the SCL clock after setting the STSPSEL bit to 0, the condition may not be generated correctly.

Solution:
Wait 1/2 cycle of the transmit/receive clock or longer after setting the STSPSEL bit to 0, and then set each condition generate bit to 1.

Application Notes:
- I2C-bus Interface Using UARTi Special Mode 1 (REJ05B1349)
- I2C-bus Interface Using UARTi Special Mode 1 (Master Transmit/Receive) (REJ05B1422)

Technical Update:
- TN-16C-130A/EA

❖ When you are not sure if this is the cause of the issue:
When setting the condition generate bit after setting the STSPSEL bit to 0, check whether the program waits 1/2 cycle of the SCL clock or longer after setting the STSPSEL bit to 0. If not, this item is the cause of the issue.

2.1.5 SDAi Digital Delay Length is Not an Appropriate Value

Check whether the delay length of the SDA digital delay function (setting value of bits DL2 to DL0 in the UiSMR3 register) is appropriate.
When the transfer rate is fast or the count source is divided, the SCLi pin level may be changed before the SDAi pin level is changed depending on the SDAi digital delay length. For example, when generating the start condition, the SDAi pin is changed to low after the SCLi pin is changed to low.

Solution:
Specify the SDAi digital delay length with a value which satisfies the following three conditions.

- When generating the condition, the SDAi pin level is changed while the SCLi pin is high.
- When outputting data, the SDAi pin level is changed while the SCLi pin is low.
- The electrical characteristics such as the setup time or hold time, and I2C-bus standard must be satisfied.

Check the actual output waveform and specify the delay length with enough margin.

Application Notes:

- I2C-bus Interface Using UARTi Special Mode 1 (REJ05B1349)
- I2C-bus Interface Using UARTi Special Mode 1 (Master Transmit/Receive) (REJ05B1422)

❖ When you are not sure if this is the cause of the issue:
Modify the program to change the SDAi digital delay length set by the SDAi digital delay setup bit (bits DL2 to DL0 in the UiSMR3 register), and see if the condition can be generated correctly. In that case this item may be the cause of the issue.
Examine pins SCLi and SDAi during communication using an oscilloscope. When outputting the start condition, if the SDAi pin is changed from high to low after the SCLi pin is changed from high to low, this item is the cause of the issue.
Also refer to the following sections in 4. Analysis Methods:

- 4.1 Examining a Register Setting
- 4.5 Examining Signals when Communicating in I2C Mode

2.1.6 Pins SCLi and SDAi are not Pulled Up

Check whether pins SCLi and SDAi are set to the N-channel open drain output pins (NCH bit in the UiC0 register is 1) and they are pulled up.
Wired-AND is used to connect with the other device in I2C-bus communication. Thus pins SCLi and SDAi need to be set to the N-channel open drain output pins and pulled up.

Solution:
When using I2C-bus communication, select N-channel open drain output (set the NCH bit in the UiC0 register to 1), and pull up pins SCLi and SDAi.

Application Notes:

- I2C-bus Interface Using UARTi Special Mode 1 (REJ05B1349)
- I2C-bus Interface Using UARTi Special Mode 1 (Master Transmit/Receive) (REJ05B1422)

❖ When you are not sure if this is the cause of the issue:
Examine the setting value of the NCH bit in the UiC0 register by a program or debugger such as the E8a emulator. If the NCH bit is 0 (pins SDAi and SCLi are CMOS output), this item is the cause of the issue.
Also refer to the following sections in 4. Analysis Methods:

- 4.1 Examining a Register Setting
- 4.5 Examining Signals when Communicating in I2C Mode
- 4.6 Checking Pull-Up for the N-Channel Open Drain Output Pin

2.1.7 Pull-up Resistance Value is not Appropriate Value

Check whether the pull-up resistance value for pins SCLi and SDAi are appropriate.
High level for N-channel open drain output is achieved by pulling up the appropriate pins. Thus a time required to become high level depends on the resistance value. The higher the resistance value is, the longer it takes to become high level. Therefore if the resistance value is too high, a high width may become short or the next low level may be output before the level becomes high. Then it may cause the issue such that high level is not transmitted to the target device.

Solution:
Specify the resistance value which satisfies the following two conditions.

- The input voltage rises up to the level that the target device can recognize the high level properly.
- Period to hold high level satisfies the electrical characteristics and the I^{2}C-bus standard.

Check the actual output waveform and specify the resistance value with enough margin.

Application Notes:

- I^{2}C-bus Interface Using UARTi Special Mode 1 (REJ05B1349)
- I^{2}C-bus Interface Using UARTi Special Mode 1 (Master Transmit/Receive) (REJ05B1422)

❖ When you are not sure if this is the cause of the issue:
Examine pins SCLi and SDAi with an oscilloscope. If the waveform output from pins SCLi and SDAi are rounded a lot and become out of the standard range of the target device specification, this item is the cause of the issue.
Also refer to the following section in 4. Analysis Methods:

- 4.5 Examining Signals when Communicating in I^{2}C Mode

2.2 SDAi Pin Becomes Low/Low Pulse is Output

Example:

- A short low pulse is output before generating the start condition or after generating the stop condition.

2.2.1 Initial Value of the SDAi Output is not Set to High

For the initial setting of the SDAi output value, check whether the corresponding port register is set to 1 (high level) before I2C mode is selected (bits SMD2 to SMD0 in the UiMR register).
The initial value for SDAi output can be set by the port register.
With I2C mode selected, when the STSPSEL bit in the UiSMR4 is 0 and the ACKC bit is 0, the level of the serial I/O circuit (initial value of output on the SDAi pin) is output. Thus when I2C mode is selected while the port register is set to 0 or the port register is not set (0 as the value after a reset is 0), the SDA pin becomes low.

Solution:

Set the port register corresponding to the SDAi pin to 1 while the serial interface is disabled (bits SMD2 to SMD0 in the UiMR register are 000b). Then select I2C mode (set bits SMD2 to SMD0 to 010b).

Application Notes:

- I2C-bus Interface Using UARTi Special Mode 1 (REJ05B1349)
- I2C-bus Interface Using UARTi Special Mode 1 (Master Transmit/Receive) (REJ05B1422)

❖ When you are not sure if this is the cause of the issue:

Halt the program at the point where the issue occurs using a debugger such as the E8a emulator. For example, if a low pulse is output when generating the stop condition, halt the program before generating the stop condition. Then examine the value on the port register corresponding to the SDAi pin. If the value of the port register is 0, this item may be the cause of the issue.
Also refer to the following section in 4. Analysis Methods:

- 4.5 Examining Signals when Communicating in I2C Mode

2.2.2 Ports Corresponding to Pins SCLi and SDAi are Not Set to High

Check whether the I^{2}C mode is selected after setting the ports corresponding to pins SCLi and SDAi to high (e.g. set the port register to 1 or set the output level of the peripheral function which shares the pins to high).
Generally when switching the pin function, or when switching an operating mode of the peripheral function, a short pulse for several nanoseconds, which is a period until a function or operating mode is switched, may appear due to the difference between delays of switching signals on the internal switching circuits.

Solution:
Perform one or both of the following operations while the serial interface is disabled (bits SMD2 to SMD0 in the UiMR register are 000b). Then select I^{2}C mode (set bits SMD2 to SMD0 to 010b).

- Set the port registers corresponding to pins SCLi and SDAi to 1.
- Set the output level of the peripheral function which shares the pins to high.

Application Notes:

- I^{2}C-bus Interface Using UARTi Special Mode 1 (REJ05B1349)
- I^{2}C-bus Interface Using UARTi Special Mode 1 (Master Transmit/Receive) (REJ05B1422)

❖ When you are not sure if this is the cause of the issue:
Modify the program to set 1 to the port registers corresponding to pins SCLi and SDAi and set the output level of the peripheral function which shares the pins to high. Then examine pins SCLi and SDAi using an oscilloscope. If low pulses are not output from pins SCLi and SDAi, this item may be the cause of the issue.
Also refer to the following section in 4. Analysis Methods:

- 4.5 Examining Signals when Communicating in I^{2}C Mode

2.2.3 ACK is Output

Check whether ACK is output by setting the ACKC bit to 1 (ACK data output) and the ACKD bit to 0 (ACK) in the UiSMR4 register. If so, the SDAi pin is driven low due to ACK output.

Solution:
When ACK output is not required, set the ACKD bit to 1 (NACK) so the SDAi pin is not driven low.

Application Notes:
- I^{2}C-bus Interface Using UARTi Special Mode 1 (REJ05B1349)
- I^{2}C-bus Interface Using UARTi Special Mode 1 (Master Transmit/Receive) (REJ05B1422)

❖ When you are not sure if this is the cause of the issue:
Check the setting values of bits ACKC and ACKD in the UiSMR4 register by a program or debugger such as the E8a emulator. If the ACKC bit is set to 1 and the ACKD bit is set to 0, this item is the cause of the issue.
Also refer to the following sections in 4. Analysis Methods:
- 4.1 Examining a Register Setting
- 4.5 Examining Signals when Communicating in I^{2}C Mode

2.2.4 ACK/NACK Setting During Idle State is Not Appropriate with the Serial I/O Circuit Selected

Before generating the start condition (as one of the cases), check whether the ACK/NACK setting is NACK output by setting bits ACKC and ACKD to 1 while the STSPSEL bit in the UiSMR4 register is 0 (select serial I/O circuit).
When the STSPSEL bit is set to 0, the serial I/O circuit is connected. Then a low level may be output depending on the state of the serial I/O circuit (e.g. the last bit of the previous transmit data is low).
Setting NACK output by setting bits ACKC and ACKD to 1 can suppress an unexpected low pulse output.

Solution:
Before generating the start condition (as one of the cases), set bits ACKC and ACKD to 1 while the STSPSEL bit is 0.

Application Notes:
- I^{2}C-bus Interface Using UARTi Special Mode 1 (REJ05B1349)
- I^{2}C-bus Interface Using UARTi Special Mode 1 (Master Transmit/Receive) (REJ05B1422)

❖ When you are not sure if this is the cause of the issue:
Examine bits ACKC and ACKD when not generating any conditions by a program or debugger such as the E8a emulator. If bits ACKC and ACKD are not set to 1, this item may be the cause of the issue.
Also refer to the following sections in 4. Analysis Methods:
- 4.1 Examining a Register Setting
- 4.5 Examining Signals when Communicating in I^{2}C Mode

2.2.5 Serial I/O Circuit is Not Selected After Generating the Condition

Check whether the STSPSEL bit in the UiSMR4 register is set to 0 (select serial I/O circuit) after generating each condition.
After setting the STSPSEL bit to 1 and generating the start condition, the STSPSEL bit has to be set back to 0 when transmitting data.
When the STSPSEL bit is set to 1, pins SCLi and SDAi are connected to the condition generate circuit. Then pins SCLi and SDAi continue outputting low after generating the start condition.

Solution:
When transmitting/receiving data, set the STSPSEL bit to 0 (select serial I/O circuit).

Application Notes:

- I2C-bus Interface Using UARTi Special Mode 1 (REJ05B1349)
- I2C-bus Interface Using UARTi Special Mode 1 (Master Transmit/Receive) (REJ05B1422)

❖ When you are not sure if this is the cause of the issue:
Examine the STSPSEL bit when transmitting/receiving data by a program or debugger such as the E8a emulator. If the STSPSEL bit is not set to 0, this item may be the cause of the issue.
Also refer to the following sections in 4. Analysis Methods:

- 4.1 Examining a Register Setting
- 4.5 Examining Signals when Communicating in I2C Mode

2.3 SCLi Pin Becomes Low/Low Pulse is Output

Example:

- After 8-bit data is received, the SCLi pin becomes low and the communication is stopped.

2.3.1 Ports Corresponding to Pins SCLi/SDAi are Not Set to High

Check whether the I^{2}C mode is selected (set bits SMD2 to SMD0 in the UiMR register to 010b) after setting the ports corresponding to pins SCLi and SDAi to high (set the port register to 1 or set the output level of the peripheral function which shares the pins to high).
Generally when switching the pin function, or when switching an operating mode of the peripheral function, a short pulse for several nanoseconds, which is a period until a function or operating mode is switched, may appear due to the difference between delays of switching signals on the internal switching circuits.

Solution:

Perform one or both of the operations below while the serial interface is disabled (bits SMD2 to SMD0 in the UiMR register are 000b). Then select I^{2}C mode.

- Set the port registers corresponding to pins SCLi and SDAi to 1.
- Set the output level of the peripheral function which shares the pins to high.

Application Notes:

- I^{2}C-bus Interface Using UARTi Special Mode 1 (REJ05B1349)
- I^{2}C-bus Interface Using UARTi Special Mode 1 (Master Transmit/Receive) (REJ05B1422)

❖ When you are not sure if this is the cause of the issue:

Modify the program to set 1 to the port registers corresponding to pins SCLi and SDAi and set the output level of the peripheral function which shares the pins to high. Then examine pins SCLi and SDAi using an oscilloscope. If low pulses are not output from pins SCLi and SDAi, this item may be the cause of the issue.
Also refer to the following sections in 4. Analysis Methods:

- 4.5 Examining Signals when Communicating in I^{2}C Mode

2.3.2 WAIT-State is not Cleared when Using the SCL WAIT Function

When using the SCL wait function by setting the SWC bit in the UiSMR2 register to 1, check whether wait-state is cleared after reading the received data and generating an acknowledge.
When using the SCL wait function by setting the SWC9 bit in the UiSMR4 register to 1, check whether wait-state is cleared after an acknowledge is determined.
When the SWC bit is set to 1, the SCLi pin is held low after receiving 8 bits. When the SWC9 bit is set to 1, the SCLi pin is held low after receiving 9 bits.
The SCLi pin can be released from being held low by setting both the SWC and SWC9 bits to 0.

Solution:
When using the SCL wait function, set bits SWC and SWC9 to 0 to release the SCLi pin from being held low after necessary processing such that the receive data is read or an acknowledge is generated/determined.

Application Notes:
- I2C-bus Interface Using UARTi Special Mode 1 (REJ05B1349)
- I2C-bus Interface Using UARTi Special Mode 1 (Master Transmit/Receive) (REJ05B1422)

❖ When you are not sure if this is the cause of the issue:
Check bits SWC and SWC9 when the SCLi pin is held low by a program or debugger such as the E8a emulator. If bits SWC and SWC9 are set to 1 (hold the SCLi pin low), this item is the cause of the issue.
Also refer to the following sections in 4. Analysis Methods:
- 4.1 Examining a Register Setting
- 4.5 Examining Signals when Communicating in I2C Mode

2.3.3 SWC2 Bit in the UiSMR2 Register is Set to 1

Check whether the SCLi pin is held low by setting the SWC2 bit in the UiSMR2 register to 1.
The SWC2 bit is not necessary for normal operation. The SCLi pin is held low during transmission or reception by setting the SWC2 bit to 1 (hold the SCLi pin low).
The SCLi pin can be released from being held low by setting the SWC2 bit to 0.

Solution:
When the SCLi pin is held low by setting the SWC2 bit to 1, set the SWC2 bit to 0 to release the SCLi pin from being held low after the necessary operation is performed.

Application Notes:
- I2C-bus Interface Using UARTi Special Mode 1 (REJ05B1349)
- I2C-bus Interface Using UARTi Special Mode 1 (Master Transmit/Receive) (REJ05B1422)

❖ When you are not sure if this is the cause of the issue:
Check the SWC2 bit when the SCLi pin is held low by a program or debugger such as the E8a emulator. If the SWC2 bit is set to 1, this item is the cause of the issue.
Also refer to the following sections in 4. Analysis Methods:
- 4.1 Examining a Register Setting
- 4.5 Examining Signals when Communicating in I2C Mode

2.3.4 Clock Phase is Changed Before Generating the Start Condition

Check whether the clock phase is changed by setting the clock phase set bit (CKPH bit in the UiSMR3 register) to 1 (with clock delay) before generating the start condition.
When the CKPH bit is set to 1, the clock phase is changed and the initial value of the SCLi pin becomes low. Thus if the CKPH bit is set to 1 before generating the start condition (while the STSPSEL bit in the UiSMR4 register is 0 and the serial I/O circuit is selected), the SCLi pin is driven low and the start condition cannot be output properly.

Solution:
When changing the CKPH bit before generating the start condition, follow the procedure below:

- Set the CKPH bit to 0 before generating the start condition (while the STSPSEL bit is 0).
- Set the CKPH bit to 1 after generating the start condition (after setting the STSPSEL bit to 1 and before setting it to 0).
- Set the CKPH bit to 0 after generating the stop condition (after setting the STSPSEL bit to 1 and before setting it to 0).

Application Notes:

- I2C-bus Interface Using UARTi Special Mode 1 (REJ05B1349)
- I2C-bus Interface Using UARTi Special Mode 1 (Master Transmit/Receive) (REJ05B1422)

❖ When you are not sure if this is the cause of the issue:
Check the procedure for setting the CKPH bit by a program or debugger such as the E8a emulator. If the CKPH bit is set to 1 before generating the start condition, or the CKPH bit is set to 0 before generating the stop condition, this item may be the cause of the issue.
Also refer to the following sections in 4. Analysis Methods:

- 4.1 Examining a Register Setting
- 4.5 Examining Signals when Communicating in I2C Mode

2.3.5 Serial I/O Circuit is Not Selected After Generating the Condition

Check whether the STSPSEL bit in the UiSMR4 register is set to 0 (select serial I/O circuit) after generating each condition.
After setting the STSPSEL bit to 1 and generating the start condition, the STSPSEL bit has to be set back to 0 when transmitting data.
When the STSPSEL bit is set to 1, pins SCLi and SDAi are connected to the condition generate circuit. Then pins SCLi and SDAi continue outputting low after generating the start condition.

Solution:
When transmitting/receiving data, set the STSPSEL bit to 0 (select serial I/O circuit).

Application Notes:

- I2C-bus Interface Using UARTi Special Mode 1 (REJ05B1349)
- I2C-bus Interface Using UARTi Special Mode 1 (Master Transmit/Receive) (REJ05B1422)

❖ When you are not sure if this is the cause of the issue:
Examine the STSPSEL bit when transmitting/receiving data by a program or debugger such as the E8a emulator. If the STSPSEL bit is not set to 0, this item may be the cause of the issue.
Also refer to the following sections in 4. Analysis Methods:

- 4.1 Examining a Register Setting
- 4.5 Examining Signals when Communicating in I2C Mode

2.4 Data is Not Output as Expected

Example:

- When transmitting data continuously, 1 byte of transmit data is sometimes dropped.

2.4.1 Clock Synchronization is Not Enabled

Check whether clock synchronization is enabled (CSC bit in the UiSMR2 register is 1).
When clock synchronization is enabled, if the clock on the M16C MCU and the clock input on the SCLi become different due to, for example, a wait inserted by the target device, the clock internally generated is synchronized with the clock input from the SCLi pin.
When clock synchronization is disabled (CSC bit is 0), if the target device inserts a wait, clocks are not synchronized. Then data cannot be output correctly.
For example, when clock synchronization is disabled, data transmission is still continued even if the target device holds the clock low to insert a wait. Then data output while the clock is held low cannot be received in the target device. If the clock is released from being held low during transmission, the remaining clock and data at that point are output, and this may cause bit slippage.
By enabling clock synchronization, if the target device holds the clock low, communication is temporarily stopped, and the communication can be restarted after the clock is released from being held low.

Solution:

When using the M16C MCU as the master device, enable clock synchronization (set the CSC bit to 1).

Application Notes:

- I2C-bus Interface Using UARTi Special Mode 1 (REJ05B1349)
- I2C-bus Interface Using UARTi Special Mode 1 (Master Transmit/Receive) (REJ05B1422)

❖ When you are not sure if this is the cause of the issue:

Check the setting value of the CSC bit by a program or debugger such as the E8a emulator. If the CSC bit is set to 0, this item may be the cause of the issue.
Check whether pins SCLi and SDAi are held low by the target device using an oscilloscope. In that case this item is the cause of the issue.
Also refer to the following sections in 4. Analysis Methods:

- 4.1 Examining a Register Setting
- 4.5 Examining Signals when Communicating in I2C Mode

2.4.2 ACK is Output

Check whether ACK is output by setting the ACKC bit to 1 (ACK data output) and the ACKD bit to 0 (ACK) in the UiSMR4 register. If so, the SDAi pin is driven low due to ACK output.

Solution:
When ACK output is not required, set the ACKD bit to 1 (NACK) so the SDAi pin is not driven low.

Application Notes:

- I2C-bus Interface Using UARTi Special Mode 1 (REJ05B1349)
- I2C-bus Interface Using UARTi Special Mode 1 (Master Transmit/Receive) (REJ05B1422)

❖ When you are not sure if this is the cause of the issue:
Check the setting values of bits ACKC and ACKD by a program or debugger such as the E8a emulator. If the ACKC bit is set to 1 and the ACKD bit is set to 0, this item is the cause of the issue.
Also refer to the following sections in 4. Analysis Methods:

- 4.1 Examining a Register Setting
- 4.5 Examining Signals when Communicating in I2C Mode

2.4.3 Serial I/O Circuit is Not Selected After Generating the Condition

Check whether the STSPSEL bit in the UiSMR4 register is set to 0 (select serial I/O circuit) after generating each condition.
After setting the STSPSEL bit to 1 and generating the start condition, the STSPSEL bit has to be set back to 0 when transmitting data.
When the STSPSEL bit is set to 1, pins SCLi and SDAi are connected to the condition generate circuit. Thus pins SCLi and SDAi continue outputting low after generating the start condition.

Solution:
When transmitting/receiving data, set the STSPSEL bit to 0 (select serial I/O circuit).

Application Notes:

- I2C-bus Interface Using UARTi Special Mode 1 (REJ05B1349)
- I2C-bus Interface Using UARTi Special Mode 1 (Master Transmit/Receive) (REJ05B1422)

❖ When you are not sure if this is the cause of the issue:
Examine the STSPSEL bit when transmitting/receiving data by a program or debugger such as the E8a emulator. If the STSPSEL bit is not set to 0, this item may be the cause of the issue.
Also refer to the following sections in 4. Analysis Methods:

- 4.1 Examining a Register Setting
- 4.5 Examining Signals when Communicating in I2C Mode

2.4.4 Next Data is Set While the Transmit Buffer is Full

Check whether the next data is set while the transmit buffer is full.
If a value is set to the UARTi transmit buffer (UiTB) register while data is present in the UiTB register (TI bit in the UiC1 register is 0), the value in the register is overwritten and the overwritten data may become missing data.

Solution:
When setting transmit data after verifying the transmit buffer register becomes empty, verify the IR bit for transmit interrupt, and if the register is confirmed to be empty, set the next transmit data.
The interrupt source for the transmit interrupt can be selected by the UARTi transmit interrupt source select bit (UiIRS bit in the UiC1 register).

Application Notes:
- I^{2}C-bus Interface Using UARTi Special Mode 1 (REJ05B1349)
- I^{2}C-bus Interface Using UARTi Special Mode 1 (Master Transmit/Receive) (REJ05B1422)

❖ When you are not sure if this is the cause of the issue:
Check the program whether the IR bit for the transmit interrupt is verified after setting a value in the transmit buffer register, and then the next transmit data is set. If the verification is not performed with the IR bit for the transmit interrupt, this item may be the cause of the issue.
Also refer to the following section in 4. Analysis Methods:
- 4.5 Examining Signals when Communicating in I^{2}C Mode

2.4.5 Transmit Interrupt Source is Not “Transmission Completed”

Check whether the UiIRS bit in the UiC1 register is set to 1 (transmission completed).
When using I^{2}C mode, the UiIRS bit must be set to 1.

Solution:
Set the UiIRS bit in the UiC1 register to 1 (transmission completed).

Application Notes:
- I^{2}C-bus Interface Using UARTi Special Mode 1 (REJ05B1349)
- I^{2}C-bus Interface Using UARTi Special Mode 1 (Master Transmit/Receive) (REJ05B1422)
- I^{2}C-bus Interface Using UARTi Special Mode 1 (Slave Transmit/Receive) (REJ05B1423)

❖ When you are not sure if this is the cause of the issue:
Check whether the UiIRS bit is set to 1 by a program or debugger such as the E8a emulator. If the UiIRS bit is not set to 1, this item may be the cause of the issue.
Also refer to the following sections in 4. Analysis Methods:
- 4.1 Examining a Register Setting
- 4.5 Examining Signals when Communicating in I^{2}C Mode

2.4.6 Arbitration Lost Has Been Detected

Check whether data is transmitted when the ALS bit in the UiSMR2 register is 1 (stop the SDAi output) while the ABT bit in the UiRB register is 1 (arbitration lost to be detected).
When the ALS bit is set to 1, if an arbitration lost is detected, the SDAi pin output is stopped. If data is transmitted while the ABT bit is 1, the data is not output from the SDAi pin.

Solution:
To restart communication after the ABT bit becomes 1 while the ALS bit is set to 1, set the ABT bit to 0 first, and then restart communication.

Application Notes:

- I2C-bus Interface Using UARTi Special Mode 1 (REJ05B1349)
- I2C-bus Interface Using UARTi Special Mode 1 (Master Transmit/Receive) (REJ05B1422)

❖ When you are not sure if this is the cause of the issue:
Check the ABT bit when data output is stopped by a program or debugger such as the E8a emulator. If the ABT bit is 1, this item may be the cause of the issue.

2.5 Fast-Mode Communication is not Available

Example:

- When data is transmitted at 400 kbps, the data cannot be received correctly in the target device.

2.5.1 UiBRG Count Source is Less Than 10 MHz

Check whether the UiBRG count source is set to 10 MHz or higher, and the setup time and the hold time for the start and stop conditions are 600 ns or longer.

For I2C-bus standard Fast-mode, the minimum values of the setup time and hold time are both 600 ns whereas the MCU requires six or more cycles of the UiBRG count source for the setup time and the hold time to detect start and stop conditions. When the UiBRG count source is 10 MHz, the setup time and hold time become 600 ns. Therefore, if the UiBRG count source is less than 10 MHz, the I2C-bus standard cannot be satisfied.

Solution:

When using I2C-bus standard Fast-mode, set the UiBRG count source to 10 MHz or higher.

Application Notes:

- I2C-bus Interface Using UARTi Special Mode 1 (REJ05B1349)
- I2C-bus Interface Using UARTi Special Mode 1 (Master Transmit/Receive) (REJ05B1422)

❖ When you are not sure if this is the cause of the issue:

By using a program or debugger such as the E8a emulator, examine the system clock select bit (CM07 bit in the CM0 register, CM11 bit in the CM1 register, and CM21 bit in the CM2 register) and UiBRG count source select bit (bits CLK1 and CLK0 in the UiC0 register) to see the CPU clock and count source settings. If the UiBRG count source is less than 10 MHz, this item is the cause of the issue.

Also refer to the following sections in 4. Analysis Methods:

- 4.1 Examining a Register Setting
- 4.2 Checking the Frequency of Operating Peripheral Function Clock f1

2.5.2 Communication Bit Rate is Set to 400 kbps

Check whether the communication bit rate is set to 400 kbps and the low width of the SCL clock is less than 1.3 μs.

Duty of the SCL clock generated in I2C mode is 50%. When the maximum value (400 kbps) of the SCL clock in I2C-bus Fast-mode is set, low width of the SCL clock is 1.25 μs. This value does not meet the I2C-bus standard for Fast-mode (fLOW = Min. 1.3 μs).

Solution:

Set the SCL clock bit rate to 384.6 kbps or less, and the low width of the SCL clock to 1.3 μs or more.

Refer to “SCL Clock Frequency” in the section “Special Mode 1 (I2C Mode)” in the User’s Manual: Hardware.

Application Notes:

- I2C-bus Interface Using UARTi Special Mode 1 (REJ05B1349)
- I2C-bus Interface Using UARTi Special Mode 1 (Master Transmit/Receive) (REJ05B1422)

❖ When you are not sure if this is the cause of the issue:

By using a program or debugger such as the E8a emulator, examine the system clock select bit (CM07 bit in the CM0 register, CM11 bit in the CM1 register, and CM21 bit in the CM2 register) and UiBRG count source select bit (bits CLK1 and CLK0 in the UiC0 register) to see the communication bit rate setting. If the communication bit rate is more than 384.6 kbps, the I2C-bus standard for Fast-mode (fLOW = Min. 1.3 μs) is not satisfied. Thus this item is the cause of the issue.

Also refer to the following sections in 4. Analysis Methods:

- 4.1 Examining a Register Setting
- 4.2 Checking the Frequency of Operating Peripheral Function Clock f1
- 4.5 Examining Signals when Communicating in I2C Mode

2.6 ACK is Always Returned

Example:

- An ACK is returned even when a nonexistent device is specified.

2.6.1 Ninth Bit of the Transmit Data is Not Set to 1

Check whether the ninth bit is set to 0 when writing the transmit data to the transmit buffer register, or check whether the transmit data is written in bytes.

In I2C mode, the ninth bit of the transmit data is transmitted as the ACK/NACK bit. Therefore if the ninth bit is set to 0, a low is output at the timing for the ACK/NACK bit. Then it may be recognized as an ACK returned by the target device. If the transmit data is written in bytes, a low pulse may also be output for the ACK/NACK bit depending on the state of the ninth bit in the transmit buffer register.

Solution:

When transmitting data, set 1 for data on the ninth bit and write the transmit buffer register in words.

Application Notes:

- I2C-bus Interface Using UARTi Special Mode 1 (REJ05B1349)
- I2C-bus Interface Using UARTi Special Mode 1 (Master Transmit/Receive) (REJ05B1422)

❖ When you are not sure if this is the cause of the issue:

Check the contents of the transmit buffer register by a program or debugger such as the E8a emulator. If 0 is written to the ninth bit when writing to the transmit buffer register, or the write operation is performed in bytes, this item may be the cause of the issue.

2.7 Unexpected Arbitration Lost Occurs

Example:

- An arbitration lost occurs even when the multi-master communication is not performed.

2.7.1 Arbitration Lost is Detected when Receiving the ACK/NACK Bit

Check whether an arbitration lost is detected when receiving the ACK/NACK bit during data transmission in master transmission or during slave address transmission in master transmission or reception.
When receiving the ACK/NACK bit, by setting the ninth bit in the UiTB register to 1, the SDAi output is released on the ninth bit of data transmission and an acknowledge is received from the target device. If an ACK is received at this time, the data set for releasing the SDAi pin and the received data (ACK) do not match. Then an arbitration lost occurs.

Solution:
Ignore an arbitration lost if it occurs when receiving the ACK/NACK bit.
When the ALS bit in the UiSMR2 register is set to 1 (stop the SDAi output), if an arbitration lost is detected, the SDAi output is stopped. Set the ABT bit in the UiRB register to 0 (arbitration lost not to be detected) and then start next communication.

Application Notes:

- I2C-bus Interface Using UARTi Special Mode 1 (REJ05B1349)
- I2C-bus Interface Using UARTi Special Mode 1 (Master Transmit/Receive) (REJ05B1422)

❖ When you are not sure if this is the cause of the issue:
Check the timing of the arbitration lost occurrence by a program or debugger such as the E8a emulator.
If an arbitration lost occurs when receiving the ACK/NACK bit, this item is the cause of the issue.

2.7.2 Arbitration Lost is Detected when Receiving the Data

Check whether an arbitration lost is detected when receiving data during data reception in master reception.
When receiving the ACK/NACK bit, 9-bit data (00FFh or 01FFh) is set to the UiTB register (b8 is 0 when generating an ACK, and 1 when generating a NACK) to release the SDAi pin and data is received from the target device. Then the data set for releasing the SDAi pin and the received data do not match. Thus an arbitration lost occurs.

Solution:
Ignore an arbitration lost if it occurs when receiving data.
When the ALS bit in the UiSMR2 register is set to 1 (stop the SDAi output), if an arbitration lost is detected, the SDAi output is stopped. Set the ABT bit in the UiRB register to 0 (arbitration lost not to be detected) and then start next communication.

Application Notes:

- I2C-bus Interface Using UARTi Special Mode 1 (REJ05B1349)
- I2C-bus Interface Using UARTi Special Mode 1 (Master Transmit/Receive) (REJ05B1422)

❖ When you are not sure if this is the cause of the issue:
Check the timing of the arbitration lost occurrence by a program or debugger such as the E8a emulator.
If an arbitration lost occurs when receiving data, this item is the cause of the issue.

3. Troubles in Slave Operation of I^{2}C Mode

Table 3.1 lists Examples of Trouble and Possible Causes. Refer to the sections in the “Refer to” column for detailed explanations and troubleshooting.

Table 3.1 Examples of Trouble and Possible Causes

Section	Trouble	Possible Cause	Refer to
3.1	Data is Not Output as Expected	ACK bit and Transmit Data are Not Set in Appropriate Timing	3.1.1
		Transmit Interrupt Source is not “Transmission Completed”	3.1.2
		Restart Condition is Detected	3.1.3
		Bit Slippage Occurred	3.1.4
3.2	Bit Slippage Occurred	UART Module is Not Initialized After the Stop Condition is Detected	3.2.1
3.3	Conditions Cannot be Detected	Setup Time and Hold Time for SDAi are Inappropriate	3.3.1
		Restart Condition is Detected	3.3.2
—	When the Cause of the Issue Cannot be Found/Determined	—	5.

3.1 Data is Not Output as Expected

Example:

- Data cannot be received after the restart condition is received.

3.1.1 ACK bit and Transmit Data are Not Set in Appropriate Timing

Check whether the clock is input from the target device before setting the ACK bit and transmit data. If so, an unexpected data may be output.

Solution:

When the SWC bit is 1, the SCLi pin is held low after receiving 8 bits. Holding the SCLi pin low can place the other device in the wait state.
When the ACK bit and transmit data cannot be set within a given time, set the SWC bit to 1.
After the ACK bit and transmit data are set, set the SWC bit to 0 to release the SCLi pin from being held low.

Application Notes:

- I^{2}C-bus Interface Using UARTi Special Mode 1 (REJ05B1349)
- I^{2}C-bus Interface Using UARTi Special Mode 1 (Slave Transmit/Receive) (REJ05B1423)

❖ When you are not sure if this is the cause of the issue:

Add the program to invert a test port immediately before setting the transmit data. Examine the test port and the SDAi pin using an oscilloscope.
If the timing that the transmit data is output from the SDAi pin comes faster than the timing that the test port is inverted, this item may be the cause of the issue.
Also refer to the following section in 4. Analysis Methods:

- 4.5 Examining Signals when Communicating in I^{2}C Mode

3.1.2 Transmit Interrupt Source is not “Transmission Completed”

Check whether the UiIRS bit in the UiC1 register is set to 1 (transmission completed).
When using I^{2}C mode, the UiIRS bit must be set to 1.

Solution:

Set the UiIRS bit in the UiC1 register to 1 (transmission completed).

Application Notes:

- I^{2}C-bus Interface Using UARTi Special Mode 1 (REJ05B1349)
- I^{2}C-bus Interface Using UARTi Special Mode 1 (Master Transmit/Receive) (REJ05B1422)
- I^{2}C-bus Interface Using UARTi Special Mode 1 (Slave Transmit/Receive) (REJ05B1423)

❖ When you are not sure if this is the cause of the issue:

Check whether the UiIRS bit is set to 1 by a program or debugger such as the E8a emulator. If the UiIRS bit is not set to 1, this item is the cause of the issue.
Also refer to the following sections in 4. Analysis Methods:

- 4.1 Examining a Register Setting
- 4.5 Examining Signals when Communicating in I^{2}C Mode

3.1.3 Restart Condition is Detected

Check whether the restart condition is received in slave mode. If the restart condition is detected in slave mode, the subsequent processing may not be executed correctly.

Solution:
Do not use the restart condition in slave mode.

Application Notes:

- I2C-bus Interface Using UARTi Special Mode 1 (REJ05B1349)
- I2C-bus Interface Using UARTi Special Mode 1 (Slave Transmit/Receive) (REJ05B1423)

❖ When you are not sure if this is the cause of the issue:
Examine pins SCLi and SDAi using an oscilloscope.
If the restart condition is issued in slave mode, this item is the cause of the issue.
Also refer to the following section in 4. Analysis Methods:

- 4.5 Examining Signals when Communicating in I2C Mode

3.1.4 Bit Slippage Occurred

Check whether noise occurs on the SCLi pin or a signal which does not satisfy the standard is input to the SCLi pin. If so, clocks may not match between the master and slave devices (bit slippage). This can be caused by differences of the characteristics such as Vih/Vil or noise canceling ability between the master and slave devices. Noise or a signal below standard on the SCLi pin may or may not be recognized as a clock due to the differences.
If bit slippage occurs, the SDAi pin may be held low.

Solution:
If the SDAi pin is held low by the target device, disable the serial interface once and switch the pins SCLi and SDAi to the programmable I/O ports. Then output a pseudo-clock (Hi-Z and low output) from the port corresponding to the SCLi pin to see whether the SDAi pin is released.
If the SDAi pin is not released by a pseudo-clock output, output a pseudo-clock repeatedly until the SDAi pin is released.

After the SDAi pin is confirmed to be released, set the mode to I2C mode again, issue a start condition and stop condition, and terminate the communication once.

Most of the slave devices initialize (reset) the I2C-bus by a start condition and stop condition. Then perform processing for restarting communication.
If the manual for your slave device describes the initialization (reset) method, initialize (reset) the slave device following the manual.

Application Notes:
- I2C-bus Interface Using UARTi Special Mode 1 (REJ05B1349)
- I2C-bus Interface Using UARTi Special Mode 1 (Slave Transmit/Receive) (REJ05B1423)

❖ When you are not sure if this is the cause of the issue:
When bit slippage occurs, data from the master or slave device is not output completely and the data output may be stopped during the operation (0 is output).
Reset the M16C MCU to check whether the SDAi pin is released from being held low. If the SDAi pin is released, the M16C MCU may cause the SDAi pin being held low. If the SDAi pin is not released, the slave device may cause the SDAi pin being held low.
When the M16C MCU is possibly the cause, halt the program immediately before the transmit conditions are satisfied (e.g. before setting transmit data) by modifying the program or using the E8a emulator, and examine the state of the CLKi pin with an oscilloscope. If the external clock is low while the CKPOL bit in the UiC0 register is 0, or if the external clock is high while the CKPOL bit is 1, this item may be the cause of the issue.
Also refer to the following section in 4. Analysis Methods:
- 4.5 Examining Signals when Communicating in I2C Mode

3.2 Bit Slippage Occurred

Example:

- When receiving data continuously, the second frame of data cannot be received as expected.

3.2.1 UART Module is Not Initialized After the Stop Condition is Detected

Check whether the UART module is initialized after the stop condition is detected. If not, the rising edge on the SCL of the stop condition is received as the clock and this causes bit slippage.

Solution:

When operating in slave mode, initialize the UART module each time the stop condition is detected.

Application Notes:

- I2C-bus Interface Using UARTi Special Mode 1 (REJ05B1349)
- I2C-bus Interface Using UARTi Special Mode 1 (Slave Transmit/Receive) (REJ05B1423)

❖ When you are not sure if this is the cause of the issue:

Check the program whether the UART module is initialized in processing (e.g. interrupt handling) when the stop condition is detected. If the UART module is not initialized, this item may be the cause of the issue.

3.3 Conditions Cannot be Detected

Examples:

- A start condition is not detected.
- A restart condition is not detected.
- A stop condition is not detected.

3.3.1 Setup Time and Hold Time for SDAi are Inappropriate

When generating the start condition, check whether the hold time is six or more cycles of the UiBRG count source until SCLi falls after SDAi falls.
When generating the stop condition, check whether the setup time is six or more cycles of the UiBRG count source before SDAi rises after SCLi rises.
If the setup time and hold time are not set properly, conditions may not be detected.

Solution:
Set the setup time and hold time to six or more cycles of the UiBRG count source.

Application Notes:

- I^{2}C-bus Interface Using UARTi Special Mode 1 (REJ05B1349)
- I^{2}C-bus Interface Using UARTi Special Mode 1 (Slave Transmit/Receive) (REJ05B1423)

❖ When you are not sure if this is the cause of the issue:
Examine pins SCLi and SDAi using an oscilloscope.
Check the setup time and hold time for SDAi whether they are six or more cycles of the UiBRG count source. If not, this item is the cause of the issue.
Also refer to the following section in 4. Analysis Methods:

- 4.5 Examining Signals when Communicating in I^{2}C Mode

3.3.2 Restart Condition is Detected

Check whether the restart condition is received in slave mode. If the restart condition is detected in slave mode, the subsequent processing may not be executed correctly.

Solution:
Do not use the restart condition in slave mode.

Application Notes:

- I^{2}C-bus Interface Using UARTi Special Mode 1 (REJ05B1349)
- I^{2}C-bus Interface Using UARTi Special Mode 1 (Slave Transmit/Receive) (REJ05B1423)

❖ When you are not sure if this is the cause of the issue:
Examine pins SCLi and SDAi using an oscilloscope.
If the restart condition is issued in slave mode, this item is the cause of the issue.
Also refer to the following section in 4. Analysis Methods:

- 4.5 Examining Signals when Communicating in I^{2}C Mode

4. Analysis Methods

This chapter describes the analysis methods with the following development environment:

- Integrated development environment: HEW
- Debugger: E8a emulator, E100 emulator

4.1 Examining a Register Setting

This section introduces methods to confirm that a register or variable is set as expected.

4.1.1 Examining a Register Setting Using a Debugger

With a debugger, the value in a register or variable can be checked at a given point. Examine a value in a register before and after setting the register to see if the value is set as expected.

Procedure

(1) Set a breakpoint on the code for setting the register to be examined.
However if there is a code to set the PRC2 bit in the PRCR register to 1 before the code for setting the register, set breakpoints on the code for setting the PRC2 bit and after the code for setting the register.

(2) Run the program and halt at the breakpoint.

(3) Use the memory widow or I/O window to check the value before setting the register.

(4) Step through the code to set the register.
However if the register is protected with the PRC2 bit, the register setting cannot be checked with this procedure. Do not halt the program between the code to set the PRC2 bit and the next code.

(5) Check the value after setting the register using the memory widow or I/O window to see if the value is set as expected.

Figure 4.1 shows the Examining the Register Setting Using a Debugger and Figure 4.2 shows Examining the Setting of the Register Protected by the PRC2 Bit.

Explanation

With step (5) above, if the value is not set as expected, the register may be write protected or the specific setting procedure may have to be followed. Refer to the User's Manual: Hardware to confirm the condition for the register setting and its setting procedure.

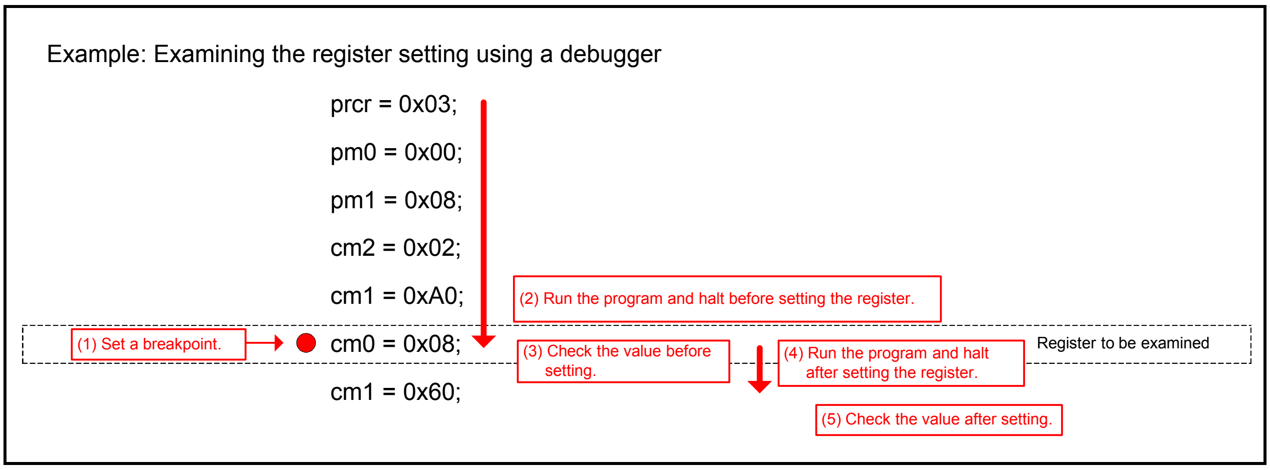

Example: Examining the register setting using a debugger
prcr = 0x03;
pm0 = 0x00;
pm1 = 0x08;
cm2 = 0x02;
cm1 = 0xA0;
(2) Run the program and halt before setting the register.
(1) Set a breakpoint.
cm0 = 0x08;
(3) Check the value before setting.
(4) Run the program and halt after setting the register.
Register to be examined
cm1 = 0x60;
(5) Check the value after setting.

Figure 4.1 Examining the Register Setting Using a Debugger

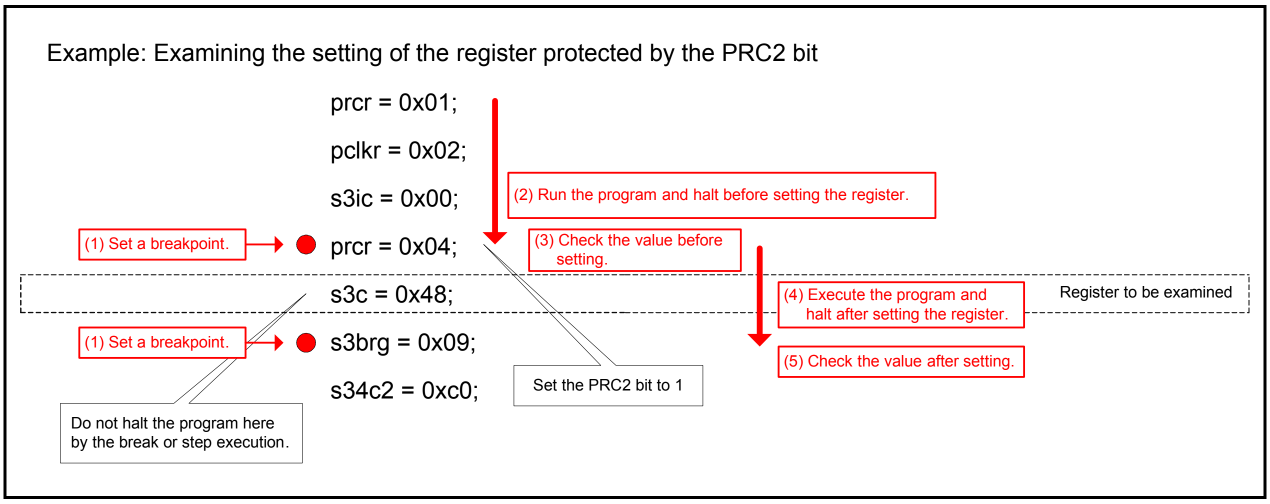

Example: Examining the setting of the register protected by the PRC2 bit
prcr = 0x01;
pclkr = 0x02;
s3ic = 0x00;
prcr = 0x04;
s3c = 0x48;
s3brg = 0x09;
s34c2 = 0xc0;
(1) Set a breakpoint.
(1) Set a breakpoint.
(2) Run the program and halt before setting the register.
(3) Check the value before setting.
(4) Execute the program and halt after setting the register.
(5) Check the value after setting.
Register to be examined
Set the PRC2 bit to 1
Do not halt the program here by the break or step execution.

Figure 4.2 Examining the Setting of the Register Protected by the PRC2 Bit

4.1.2 Examining a Register Setting Using an Oscilloscope

Add test code to read the register after the code for setting the register, and if the register value read is an expected value, then output a high level signal from the test port. Check the test port state with the oscilloscope to see if an expected value is set.
For test ports, use ports which do not affect the system by setting the direction to output.

Procedure

(1) Add code to output the test result on the test ports in the user program.
Figure 4.3 shows an Example of Adding Test Codes.
(2) Run the program and check the test port states with an oscilloscope.

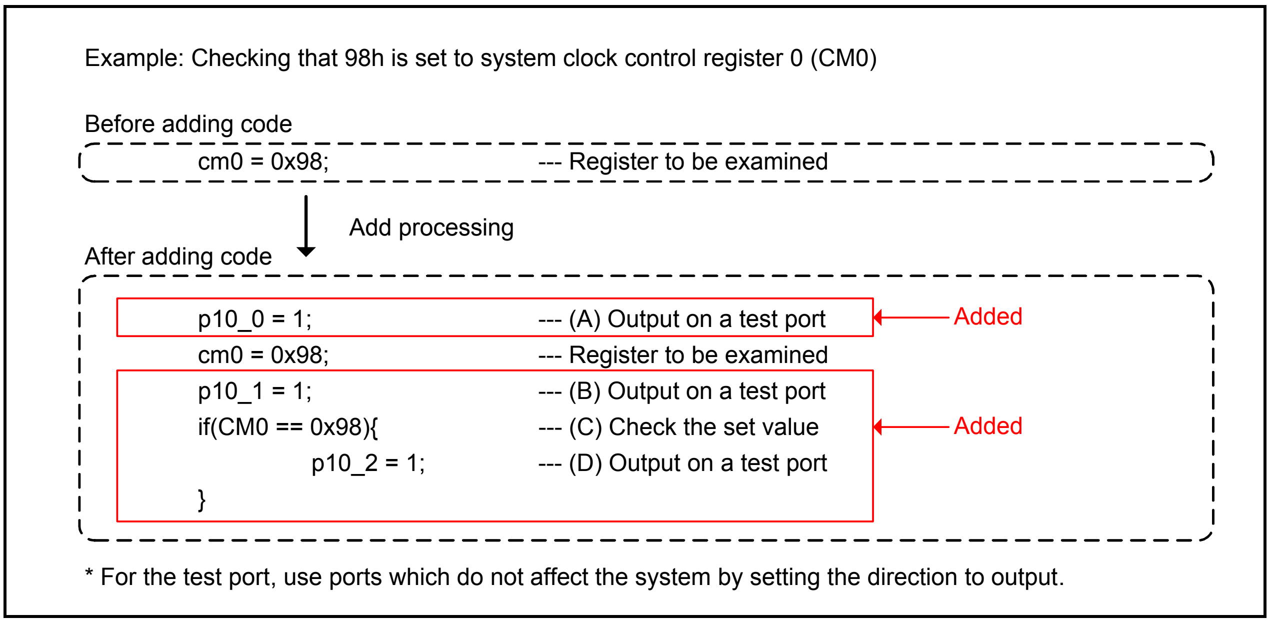
Explanation

Examine changes of test ports in step (2) above with an oscilloscope.
The following are criteria for judging the result when executing the test code shown in Figure 4.3.

When the output is as expected:
High level signals are output from all test ports as shown in Figure 4.4. The register is set correctly. Thus this is not the cause of the issue.

When the output is not as expected:
A high level signal is not output from test port (D) as shown in Figure 4.5. The register may be write protected or the specific setting procedure may have to be followed. Refer to the User's Manual: Hardware to confirm the condition for the register setting and its setting procedure.

When the program stopped or ran out of control after setting the register
High level signals may not output from test ports (B) and (D) as shown in Figure 4.6. Refer to the User's Manual: Hardware to check whether the register setting procedure, recommended operating conditions of electrical characteristics, and related notes are correctly followed.


Example: Checking that 98h is set to system clock control register 0 (CM0)

Before adding code

```
          cm0 = 0x98;                     --- Register to be examined
```

Add processing

After adding code

```
          p10_0 = 1;                      --- (A) Output on a test port
          cm0 = 0x98;                     --- Register to be examined
          p10_1 = 1;                      --- (B) Output on a test port
          if(CM0 == 0x98){                --- (C) Check the set value
                    p10_2 = 1;            --- (D) Output on a test port
          }
```

* For the test port, use ports which do not affect the system by setting the direction to output.

Figure 4.3 Example of Adding Test Codes

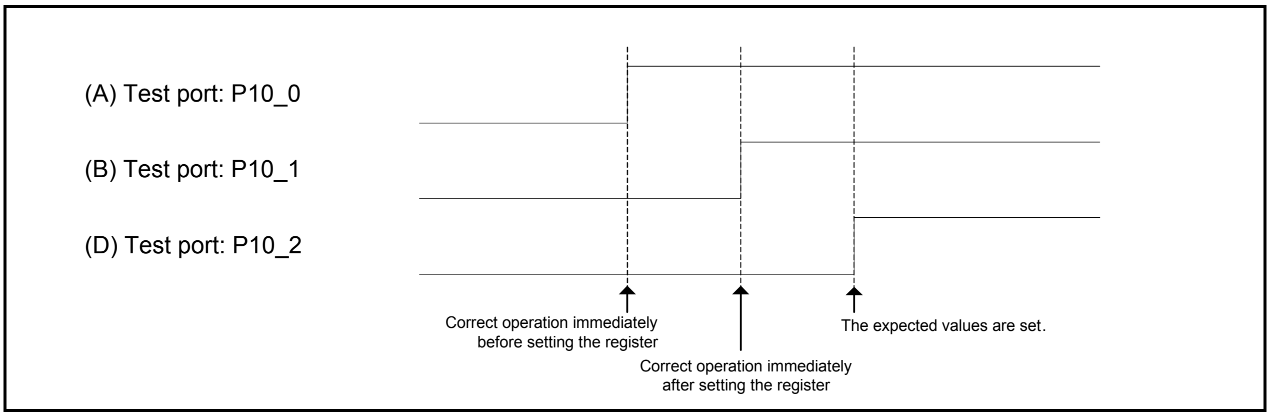

(A) Test port: P10_0
(B) Test port: P10_1
(D) Test port: P10_2
Correct operation immediately before setting the register
Correct operation immediately after setting the register
The expected values are set.

Figure 4.4 Waveform when the Value is Set as Expected

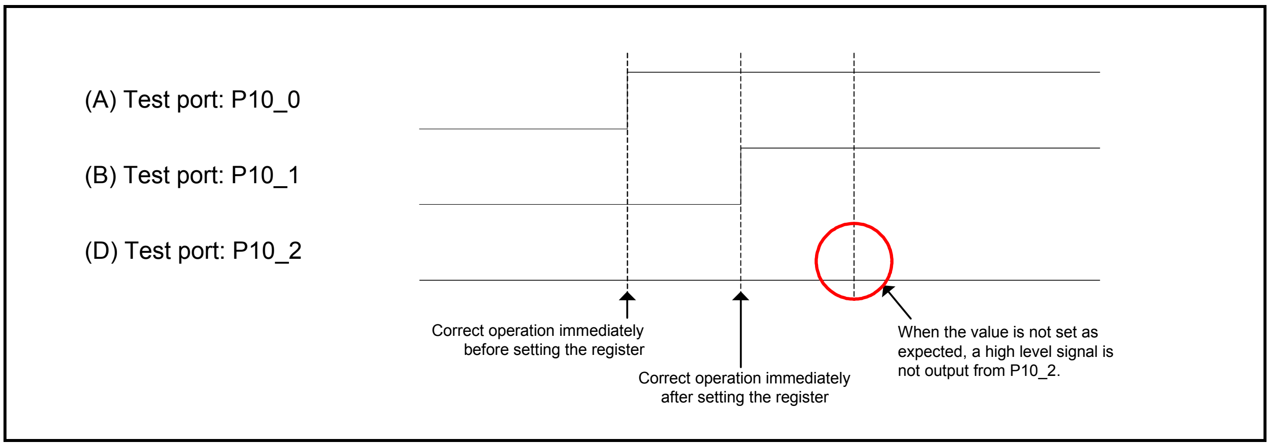

(A) Test port: P10_0
(B) Test port: P10_1
(D) Test port: P10_2
Correct operation immediately before setting the register
Correct operation immediately after setting the register
When the value is not set as expected, a high level signal is not output from P10_2.

Figure 4.5 Waveform when the Value is Not Set as Expected

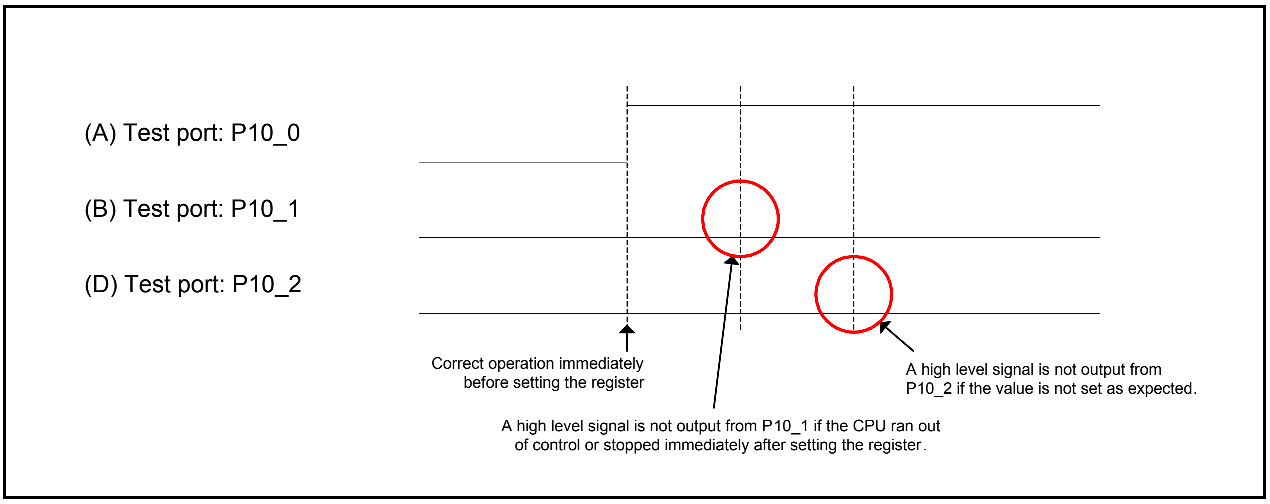

(A) Test port: P10_0
(B) Test port: P10_1
(D) Test port: P10_2
Correct operation immediately before setting the register
A high level signal is not output from P10_1 if the CPU ran out of control or stopped immediately after setting the register.
A high level signal is not output from P10_2 if the value is not set as expected.

Figure 4.6 Waveform when the Program Stopped or Ran Out of Control after Setting a Register

4.2 Checking the Frequency of Operating Peripheral Function Clock f1

This section describes methods to see whether peripheral function clock f1 is configured correctly.

4.2.1 Using an Oscilloscope to Check f1 Output from CLKOUT with the Clock Output Function

The clock output function is the function to output f1, f8, f32, or fC from the CLKOUT pin in single-chip mode. f1 is the same as the clock before the CPU clock passes the divider.
Specify the clock output function to output f1 from the CLKOUT pin and use an oscilloscope to check the frequency output from the CLKOUT pin.
The clock output from the CLKOUT pin should be 25 MHz or less. If it exceeds 25 MHz, use f8.
Make sure that the output from the CLKOUT pin does not affect the system.

Procedure

(1) Add code to output f1 from the CLKOUT pin (set the PCLK5 bit in the PCLKR register to 1) to the program. When f1 exceeds 25 MHz, output f8 instead of f1 (set bits CM01 and CM00 in the CM0 register to 10b).
(2) Run the program and examine the CLKOUT pin using an oscilloscope.

Explanation

With step (2) above, consider the checked frequency with the division selected. If the frequency is not as expected, check whether the registers associated with the system clock are set as expected. For details on the method, refer to “4.1 Examining a Register Setting”.

4.2.2 Checking Timer A Pulse Output Using an Oscilloscope

When f1 is used as the timer A count source and pulse is output with the counter value set to 0, half of the f1 frequency is output from the TAiOUT pin.
f1 is the same as the clock before the CPU clock passes the divider.
Use timer A to output half of the f1 frequency and check the frequency output from the TAiOUT pin with an oscilloscope.
Make sure that the output from the TAiOUT pin does not affect the system.

Procedure

(1) Add code to output half of the f1 frequency using timer A.
Setting for timer A
- Operating mode: Timer mode
- Pulse output function: Pulse output
- Count source: f1
- Timer register: 0000h
- Count operation: Started

(2) Run the program and check the TAiOUT (i = channel to output the timer) pin using an oscilloscope.

Explanation

With step (2) above, double the checked frequency and consider it with the division selected. If the frequency is not as expected, check whether the registers associated with the system clock are set as expected. For the method, refer to “4.1 Examining a Register Setting”.

4.3 Examining Code where an Interrupt Occurs

This section describes methods to examine code to see if an unexpected interrupt occurs.

4.3.1 Examining Code Using the Trace Function of the ICE (E100)

The ICE can trace code when running the program. Examine the traced code to see where an interrupt occurred.

Procedure

(1) Set a breakpoint on the first code of the interrupt handler to be examined.
(2) Run the program and halt at the breakpoint.
(3) Open the trace window and see if the timing the interrupt occurred is as expected.

Explanation

With step (3) above, if the interrupt occurred at an unexpected point, check the following:

- A few instructions immediately before the interrupt occurred may have problems. Refer to the User's Manual: Hardware to check the conditions and the procedure for setting the register.
- When using an interrupt that occurs when an external signal is received, a signal may be input at an unexpected timing. Use an oscilloscope to see if an unexpected signal is input on the receive pin.

4.3.2 Examining Code Using a Debugger

By stepping through return processing (REIT) from the interrupt handler, the program can be halted at the next code of the code where the interrupt occurred. Then the point where the interrupt occurred can be determined.

Procedure

(1) Set a breakpoint on the return instruction (REIT) of the interrupt handler.
(2) Run the program and halt at the breakpoint.
(3) Step through the return processing (REIT) to check the code immediately before the address to return.

Explanation

It is determined that the interrupt occurred immediately after the code checked in step (3) above was executed. If the point where the interrupt occurred is an unexpected point, check the following:

- A few instructions immediately before the interrupt occurred may have a problem. Refer to the User's Manual: Hardware to check the conditions and the procedure for setting the register.
- If using an interrupt which occurs when an external signal is received, a signal may be input at an unexpected timing. Use an oscilloscope to see if an unexpected signal is input on the receive pin.

4.4 Determining the Device that Holds Pins SDAi and SCLi Low

This section describes methods to determine the device that outputs low when the communication is canceled due to pins SDAi and SCLi being held low.

4.4.1 Determining by Resetting the M16C MCU

When the M16C is reset, the SFR states become the initial setting values and pin states become input ports. Examine pins SDAi and SCLi after a reset to determine whether the M16C MCU holds these pins low.

Procedure

(1) Input a reset and examine the states of pins SDAi and SCLi using an oscilloscope.

Explanation

With step (1) above, examine whether pins SDAi and SCLi become high.
The following are criteria for judging the result.

When pins SDAi and SCLi become high
The M16C MCU holds the pins low. Check if the communication procedure is correct. Also check if bit slippage occurred due to noise or the other factor.

When pins SDAi and SCLi remain low
The external device holds the pins low. Check if bit slippage occurred due to noise or the other factor.

4.4.2 Determining by Connecting Resistors to the Communication Lines

When resistors are connected in series between the M16C MCU and the external device (e.g. EEPROM), and the observation points are placed between the M16C MCU and the resistors, the low levels which are output from the external device will appear slightly higher than the ground levels. Examine pins SDAi and SCLi during communication to determine the device that holds these pins low.

Procedure

(1) Connect resistors in series between the M16C MCU and the external device.
Figure 4.7 shows an Example of a Circuit with Resistors Connected to the Communication Lines and Criteria for Judging the Result.
(2) Run the program and examine the SDAi and SCLi pin states using an oscilloscope.

Explanation

With step (2) above, examine changes of the SDAi and SCLi pin states.
If low levels of pins SDAi and SCLi appear slightly higher than the ground levels, the device, which has resistors between the observation points and its own SCLi and SDAi pins, outputs low levels.
If low levels of pins SDAi and SCLi are at the ground levels, the device, which has no resistors between the observation points and its own SDAi and SCLi pins, outputs low levels.
The criteria for judging the result are shown in Figure 4.7.

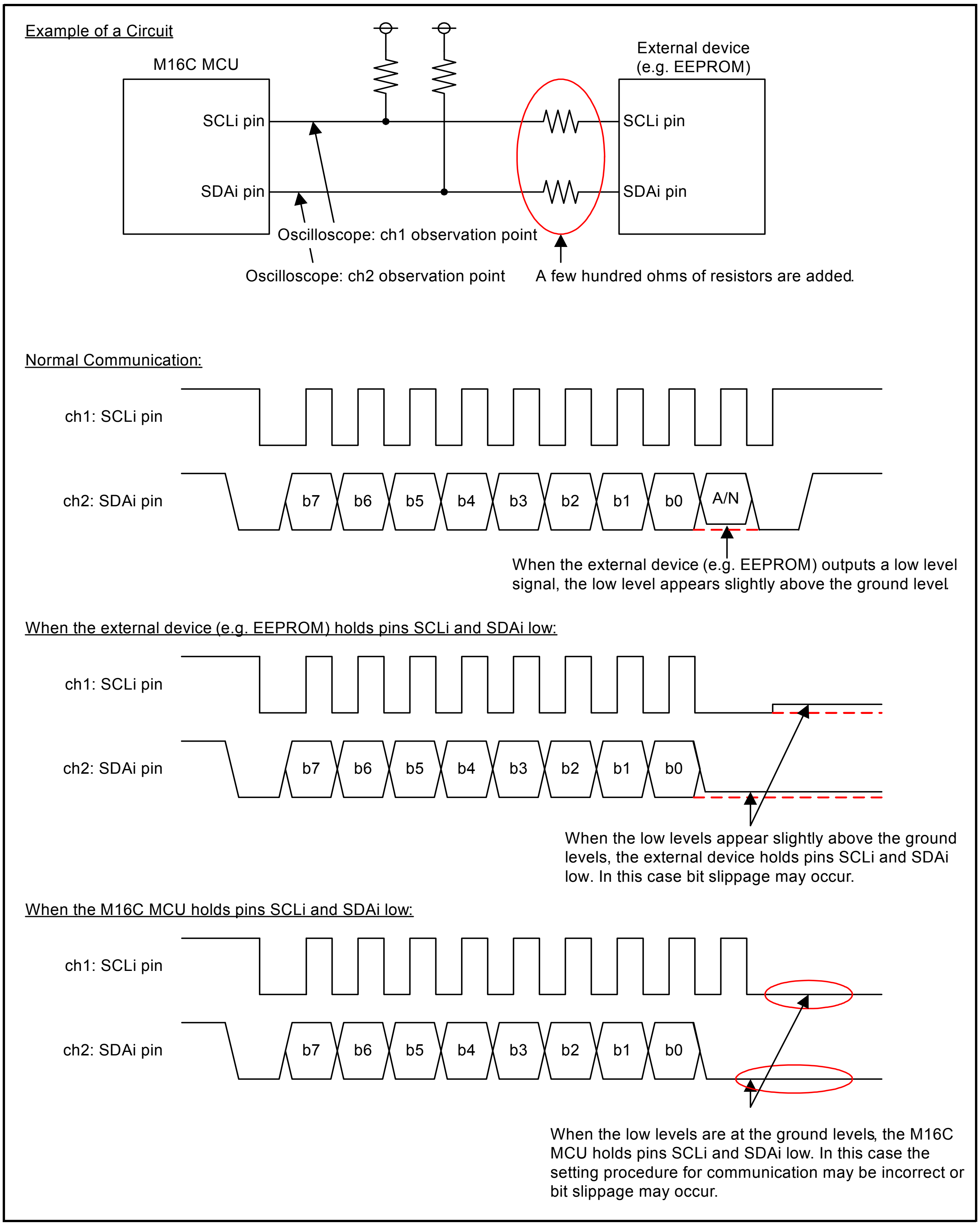

Example of a Circuit
M16C MCU
SCLi pin
SDAi pin
External device
(e.g. EEPROM)
SCLi pin
SDAi pin
Oscilloscope: ch1 observation point
Oscilloscope: ch2 observation point
A few hundred ohms of resistors are added.
Normal Communication:
ch1: SCLi pin
ch2: SDAi pin
b7
b6
b5
b4
b3
b2
b1
b0
A/N
When the external device (e.g. EEPROM) outputs a low level signal, the low level appears slightly above the ground level.
When the external device (e.g. EEPROM) holds pins SCLi and SDAi low:
ch1: SCLi pin
ch2: SDAi pin
When the low levels appear slightly above the ground levels, the external device holds pins SCLi and SDAi low. In this case bit slippage may occur.
When the M16C MCU holds pins SCLi and SDAi low:
ch1: SCLi pin
ch2: SDAi pin
When the low levels are at the ground levels, the M16C MCU holds pins SCLi and SDAi low. In this case the setting procedure for communication may be incorrect or bit slippage may occur.

Figure 4.7 Example of a Circuit with Resistors Connected to the Communication Lines and Criteria for Judging the Result

4.5 Examining Signals when Communicating in I2C Mode

This section describes methods to examine signals of the output clock or data.

The methods are described for each communication operation of I2C mode in the following order.

- Master operation in I2C mode
- Generating conditions in master operation of I2C mode
- Slave operation in I2C mode

4.5.1 Examining Signals in Master Operation of I2C Mode Using an Oscilloscope

Add code to invert a signal on the test port immediately before the code to set the data for master transmission or reception. The inversion is used as a trigger for the oscilloscope to capture the waveform in the master operation.
Examine the data output from the SDAi pin and the clock output from the SCLi pin at the timing to invert a signal on the test port using an oscilloscope. By doing this, problems you have encountered can be speculated.
For the test port, use a port which does not affect the system by setting the direction to output.

Procedure

(1) Add code to invert a signal on the test port immediately before the code to set the data to the UiTB register.
(2) Run the program to see the test port, SDAi pin, and SCLi pin states using an oscilloscope.
    - Set an inversion of a signal on the test port as the trigger to capture the waveform.

Explanation

Examine changes of the test port, SDAi pin, and SCLi pin in step (2) above with an oscilloscope.

This section introduces a normal pattern and error patterns of signals in master operation of I2C mode.
With error patterns, possible causes of issues are described.
The signal patterns introduced here are based on the following condition:

- CKPH bit in the UiSMR3 register is 1 (with clock delay).

- **Signal pattern in a normal operation [A-1]**

  Transmit data is output from the SDAi pin and the clock is output from the SCLi pin immediately after a signal on the test port is inverted.
  In this case, the transmit data from the SDAi pin is equal to the data set in the UiTB register and the clock is output from the SCLi pin in an expected period. Therefore output signals are normal and this is not the cause of the issue.
  Make sure signals satisfy the operating conditions (rated values) in both communication devices.

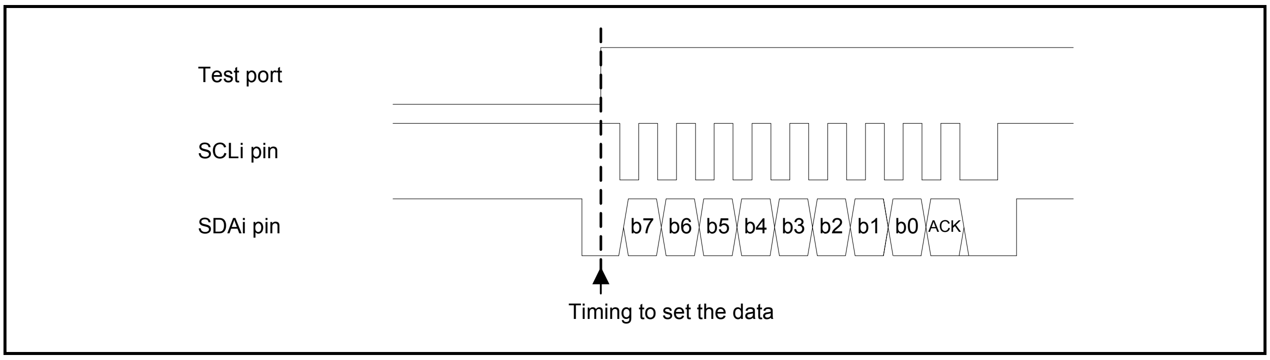


Figure 4.8 Signal Pattern in a Normal Operation [A-1]

- **Signal pattern in an error operation [A-2]**

  A signal on the test port is inverted after the start condition is output, then signals on pins SDAi and SCLi remain low. In this case the transmit condition is not satisfied. Check whether the transmission is enabled (TE bit in the UiC1 register is 1).

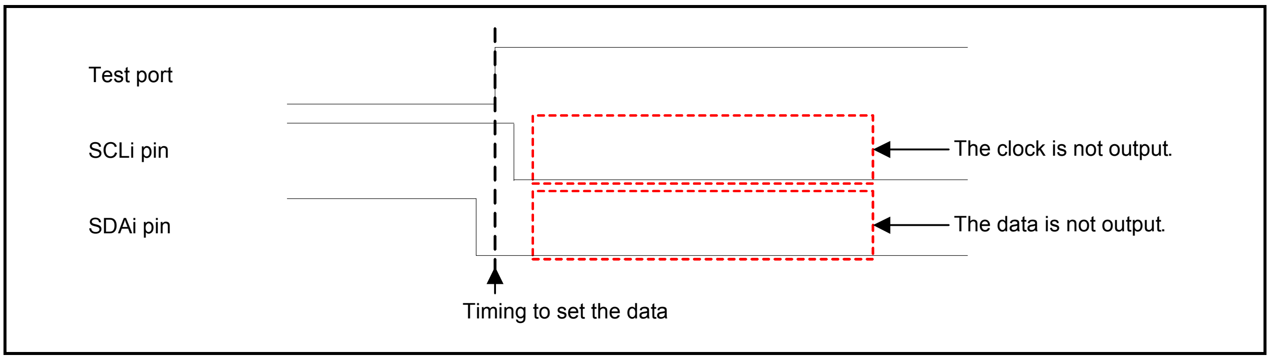


Figure 4.9 Signal Pattern in an Error Operation [A-2]

- **Signal pattern in an error operation [A-3]**

  Low signals on pins SDAi and SCLi do not change after a signal on the test port is inverted. In this case pins SDAi and SCLi are not pulled up. Pull them up with an external circuit.

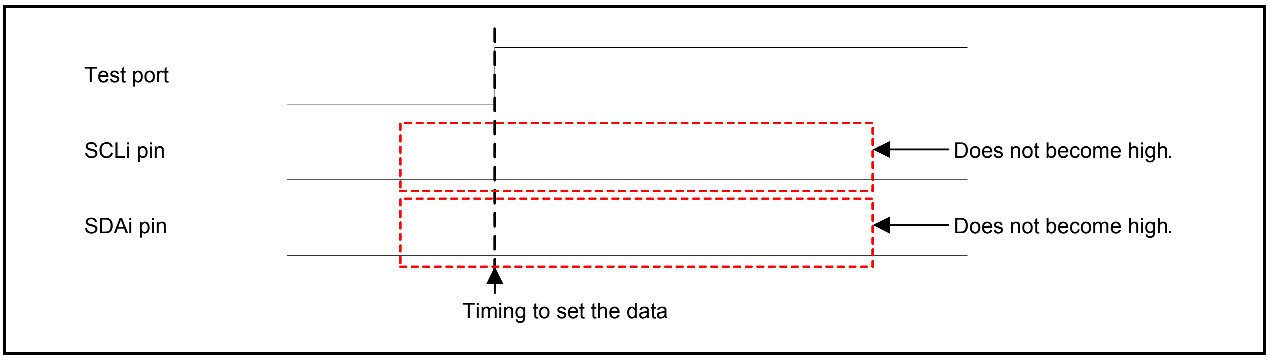


Figure 4.10 Signal Pattern in an Error Operation [A-3]

- **Signal pattern in an error operation [A-4]**

  Short low pulses appear on pins SDAi and SCLi. In this case noise may be introduced from peripheral circuits or external sources. If a pulse appears on the SCLi pin, bit slippage may occur in the target device.

  When noise occurs, add a few hundred ohms of resistors in series on pins SCLi and SDAi or move the signal lines farther from the cause of the noise.

  When using resistors in series, the I2C-bus parameter in the system must be within the allowable range.

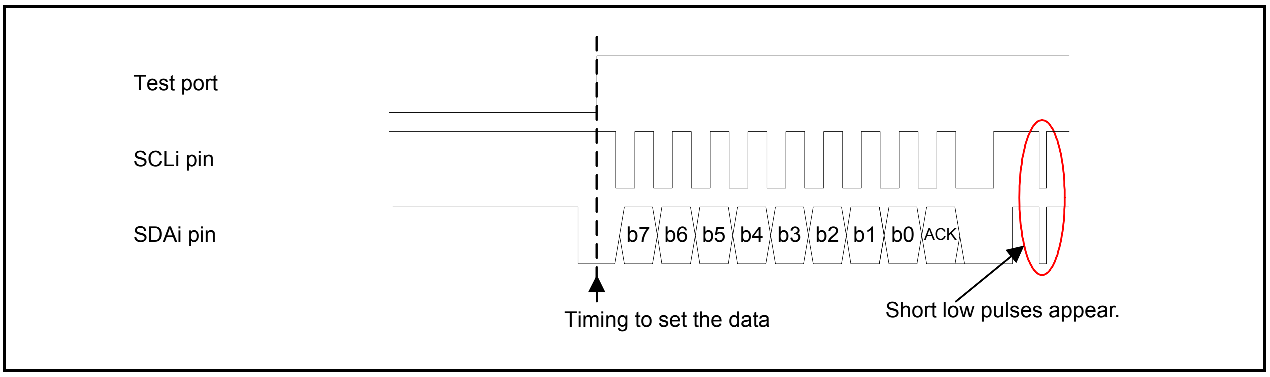


Figure 4.11 Signal Pattern in an Error Operation [A-4]

- **Signal pattern in an error operation [A-5]**

  Low level output from pins SDAi and SCLi does not fall down to 0 V or high level does not rise up to VCC. In this case the target device or other pins connected to pins SDAi and SCLi may output signals, and signal collision may occur. The operation conditions VIH or VIL in the target device are not satisfied, and low and high levels may not be recognized correctly.
  Furthermore, if adding approximately 100 ohms of resistors does not cause pull-up or pull-down, and pin levels of SDAi and SCLi do not change, then CMOS output may be performed.
  Examine the circuit and settings whether signals are output from the target device or pins connected to pins SDAi and SCLi.

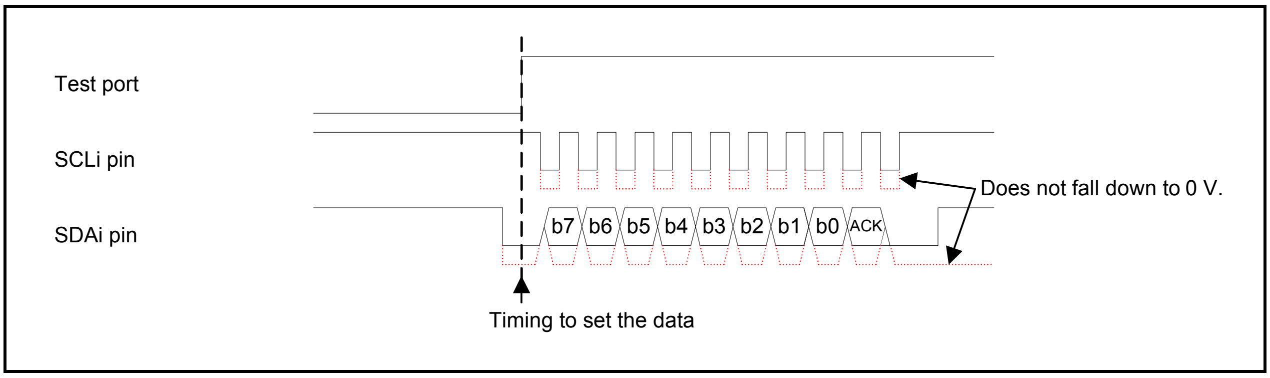


Figure 4.12 Signal Pattern in an Error Operation [A-5]

- **Signal pattern in an error operation [A-6]**

  High level output from pins SDAi and SCLi does not rise up to VCC. In this case the standards may not be satisfied such that the signal level may not rise up to the VIH of the target device or high period is not long enough due to large wiring capacity or too high pull-up resistor connected to pins SDAi and SCLi. Or bit slippage may occur.
  Design the circuit with enough margin so the wiring capacity or the pull-up resistor connected to pins SDAi and SCLi becomes appropriate.

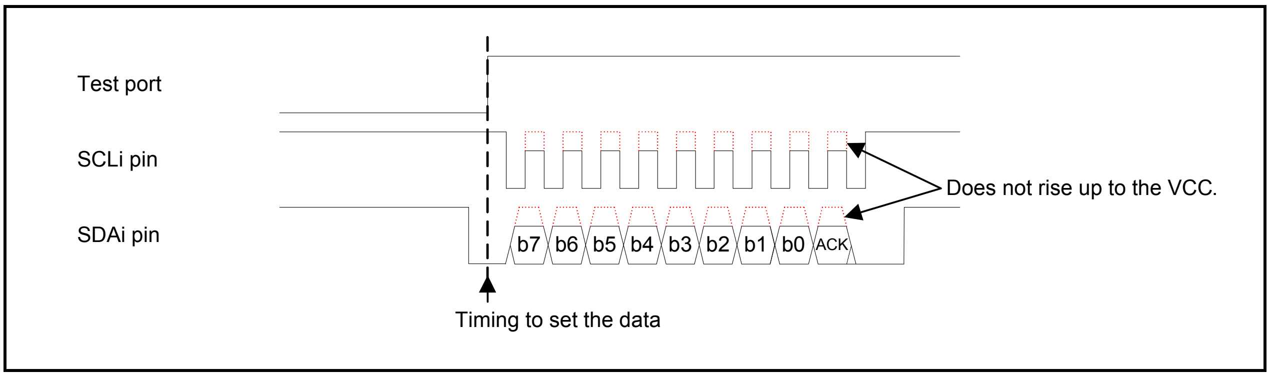
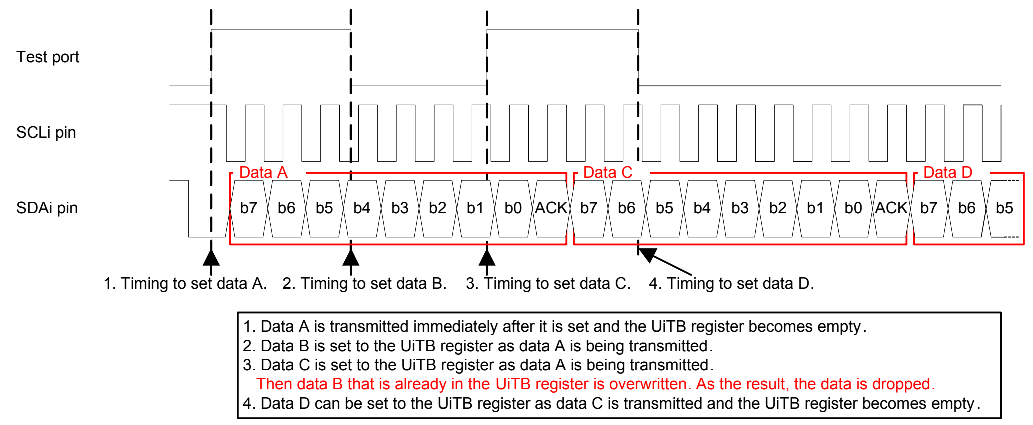


Figure 4.13 Signal Pattern in an Error Operation [A-6]

- **Signal pattern in an error operation [A-7]**

  The test port is inverted multiple times while 1-byte transmit data is output. In this case the next data is set to the UiTB register while the transmit buffer is full (TI bit in the UiC1 register is 0). Then the data in the transmit buffer is overwritten and the data may be dropped.
  Set the next transmit data in the interrupt handler for completion of transmission.



1. Data A is transmitted immediately after it is set and the UiTB register becomes empty.
2. Data B is set to the UiTB register as data A is being transmitted.
3. Data C is set to the UiTB register as data A is being transmitted.
   Then data B that is already in the UiTB register is overwritten. As the result, the data is dropped.
4. Data D can be set to the UiTB register as data C is transmitted and the UiTB register becomes empty.

Figure 4.14 Signal Pattern in an Error Operation [A-7]

- **Signal pattern in an error operation [A-8]**

  A NACK is received with the ACK/NACK bit when transmitting the slave address. In this case the transmitted slave address does not match the slave address of the target device. Check the slave address with the specification document for the target device and try to transmit with the right address.
  The target device is performing write operation and a NACK is returned. Wait until the target device completes write operation and try to transmit the slave address again.

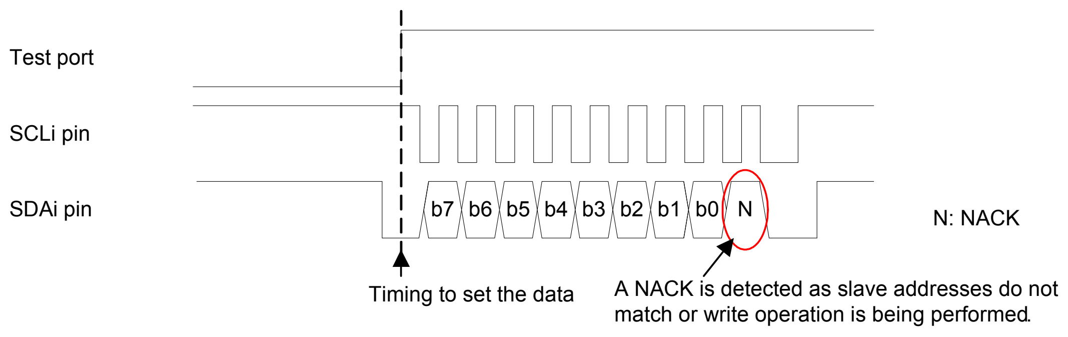

Test port
SCLi pin
SDAi pin
b7 b6 b5 b4 b3 b2 b1 b0 N
N: NACK
Timing to set the data
A NACK is detected as slave addresses do not match or write operation is being performed.

Figure 4.15 Signal Pattern in an Error Operation [A-8]

4.5.2 Examining Signals when Generating a Condition in Master Operation

Add code to invert a signal on the test port immediately before the code to generate the start condition. The inversion is used as a trigger for oscilloscope to capture the waveform when generating the condition in master operation of I^{2}C mode.
Examine the data output from the SDAi pin and the clock output from the SCLi pin at the timing to invert a signal on the test port using an oscilloscope. By doing this, problems you have encountered can be speculated.
For the test port, use a port which does not affect the system by setting the direction to output.

Procedure

(1) Add code to invert a signal on the test port immediately before the code to set the STSPSEL bit in the UiSMR4 register to 1 (select condition generate circuit).
(2) Run the program to see the test port, SDAi pin, and SCLi pin states using an oscilloscope.
- Set an inversion of a signal on the test port as the trigger to capture the waveform.

Explanation

Examine changes of the test port, SDAi pin, and SCLi pin in step (2) above with an oscilloscope.

This section introduces a normal pattern and error patterns of signals when performing master operation in I^{2}C mode. With error patterns, possible causes of issues are described.
Signal patterns introduced here are based on the following condition:
- CKPH bit in the UiSMR3 register is 1 (with clock delay).

- **Signal pattern in a normal operation [B-1]**

  The start condition is output from pins SDAi and SCLi immediately after a signal on the test port is inverted. Also pins SDAi and SCLi are driven high until immediately before the test port is inverted. In this case output signals are normal and this is not the cause of the issue.
  Make sure signals satisfy the operating conditions (rated values) in both communication devices.

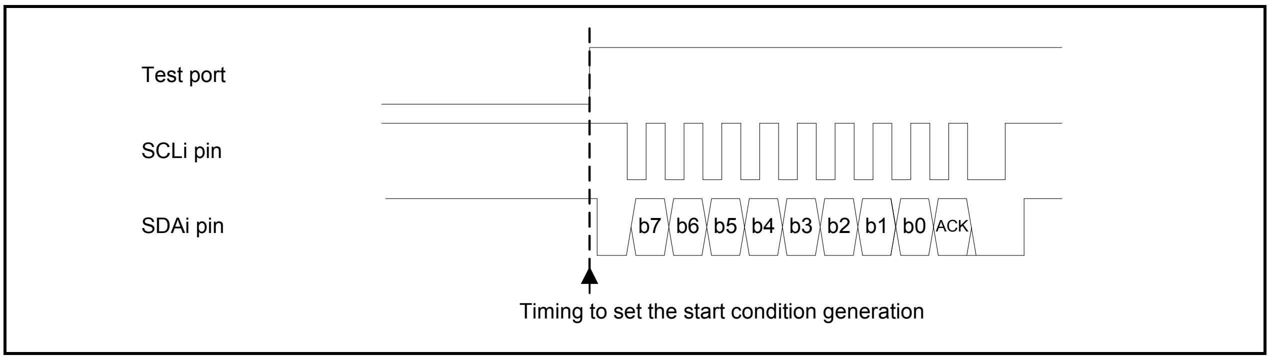


Figure 4.16 Signal Pattern in a Normal Operation [B-1]

- **Signal pattern in an error operation [B-2]**

  Pin levels of SDAi and SCLi become low for some period before the test port is inverted. In this case the procedure to set the SFRs for generating the start condition may not be followed properly.
  Refer to the following application notes for details on the setting procedure.
  - I2C-bus Interface Using UARTi Special Mode 1 (REJ05B1349)
  - I2C-bus Interface Using UARTi Special Mode 1 (Master Transmit/Receive) (REJ05B1422)
  - I2C-bus Interface Using UARTi Special Mode 1 (Slave Transmit/Receive) (REJ05B1423)

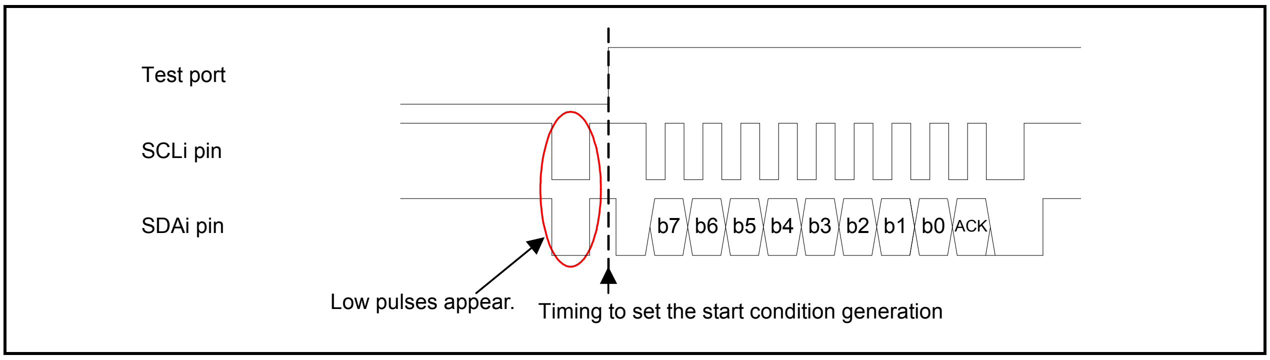


Figure 4.17 Signal Pattern in an Error Operation [B-2]

- **Signal Pattern in an Error Operation [B-3]**

  The SCLi pin becomes low before the SDAi pin becomes low at the start condition generation immediately after the test port is inverted. In this case the delay length of the SDA digital delay function (setting value in bits DL2 to DL0 in the UiSMR3 register) may be too much. Check whether the delay length is set with the appropriate value.

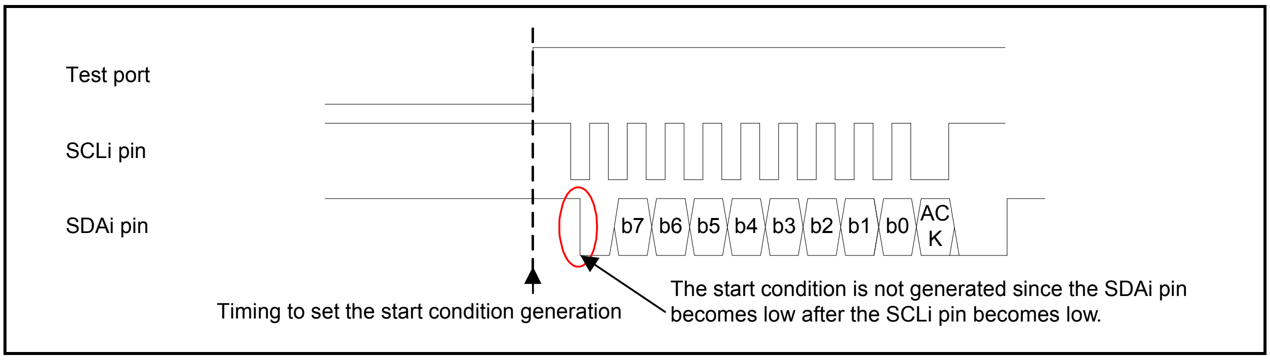

Test port
SCLi pin
SDAi pin
b7
b6
b5
b4
b3
b2
b1
b0
ACK
Timing to set the start condition generation
The start condition is not generated since the SDAi pin becomes low after the SCLi pin becomes low.

Figure 4.18 Signal Pattern in an Error Operation [B-3]

4.5.3 Examining Signals in Slave Operation

Add code to invert a signal on the test port immediately before reading or writing data that is transmitted or received by the slave device. The inversion is used as a trigger for the oscilloscope to capture the waveform when performing slave operation.
Examine the data output from the SDAi pin and the clock output from the SCLi pin at the timing to invert a signal on the test port using an oscilloscope. By doing this, problems you have encountered can be speculated.
For the test port, use a port which does not affect the system by setting the direction to output.

Procedure

(1) Add code to invert a signal on the test port immediately before the code to set the data to the UiTB register or read the data from the UiRB register.
(2) Run the program to see the test port, SDAi pin, and SCLi pin states using an oscilloscope.
  • Set an inversion of a signal on the test port as the trigger to capture the waveform.

Explanation

Examine changes of the test port, SDAi pin, and SCLi pin in step (2) above with an oscilloscope.

This section introduces a normal pattern and error patterns of signals when performing slave operation in I2C mode. With error patterns, possible causes of issues are described.
Signal patterns introduced here are based on the following condition:
• CKPH bit in the UiSMR3 register is 1 (with clock delay).

- **Signal pattern in a normal operation [C-1]**

  The test port is inverted at the falling edge of the eighth bit on the SCLi pin when receiving the slave address and then inverted at the falling edge of the ninth bit on the SCLi pin when transmitting or receiving data. In this case the received data for the SDAi pin can be read from the UiRB register and the transmit data from the SDAi pin is equal to the data set in the UiTB register. Thus input/output signals are normal and this is not the cause of the issue.
  Make sure signals satisfy the operating conditions (rated values) in both communication devices.

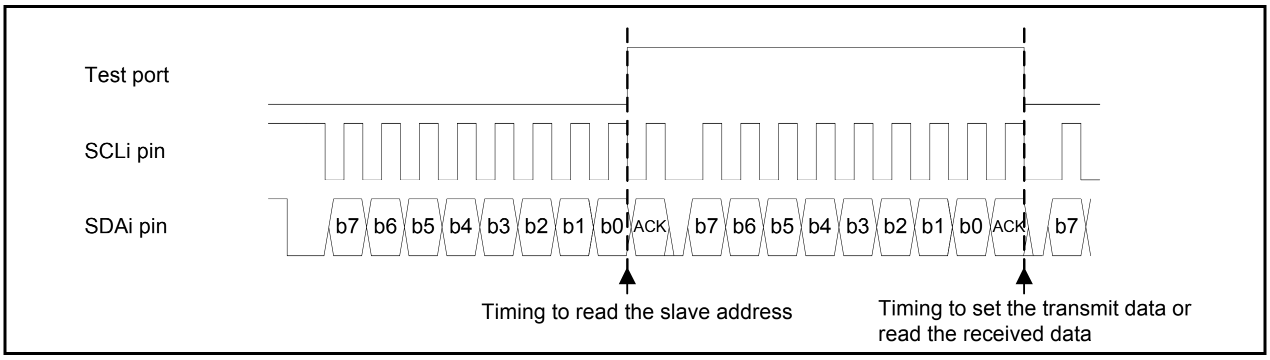

Test port
SCLi pin
SDAi pin
b7 b6 b5 b4 b3 b2 b1 b0 ACK b7 b6 b5 b4 b3 b2 b1 b0 ACK b7
Timing to read the slave address
Timing to set the transmit data or read the received data

Figure 4.19 Signal Pattern in a Normal Operation [C-1]

- **Signal pattern in an error operation [C-2]**

  The test port is inverted at the falling edge of the eighth bit on the SCLi pin when receiving the slave address and then inverted at the falling edges of the eighth and ninth bits on the SCLi pin when transmitting or receiving data. In this case the received data may be read with the receive interrupt.
  Orders to store data in the UiRB register are different between the receive interrupt and the transmit interrupt. Check whether the transmit interrupt is used to read data after the slave address reception. If data is read with the receive interrupt, the order to store data has to be changed.

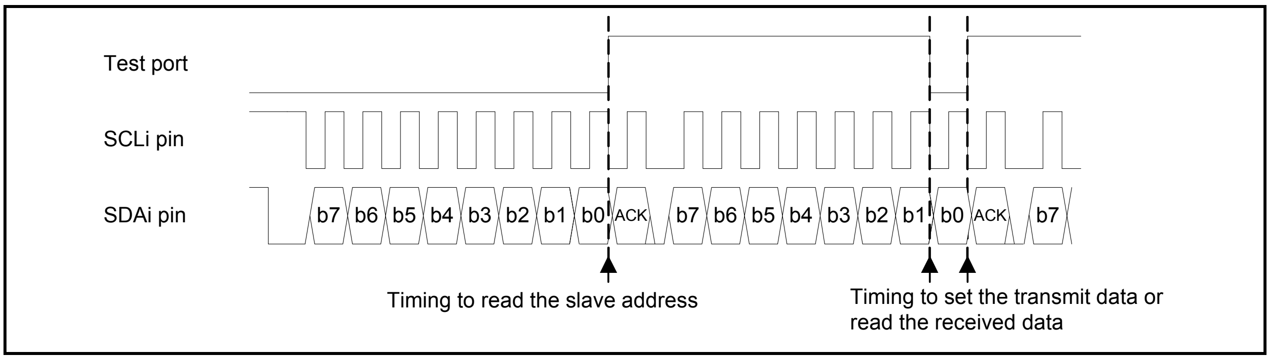

Test port
SCLi pin
SDAi pin
b7 b6 b5 b4 b3 b2 b1 b0 ACK b7 b6 b5 b4 b3 b2 b1 b0 ACK b7
Timing to read the slave address
Timing to set the transmit data or read the received data

Figure 4.20 Signal Pattern in an Error Operation [C-2]

4.6 Checking Pull-Up for the N-Channel Open Drain Output Pin

This section describes the method to check if the N-channel open drain output pin is correctly pulled up.

4.6.1 Checking with an Oscilloscope

Use an oscilloscope to check whether the N-channel open drain output pin is pulled up correctly.

Procedure

(1) Connect the pin to be checked to VSS (pull-down) through the same resistor as the one used for pull-up.
(2) Run the program to see the pin state with an oscilloscope.

Explanation

The possible cases are provided below to determine the result from step (2) above.

When the N-channel open drain output pin is pulled up correctly:
When the pin does not output (during reset), the pin level becomes half of the VCC. When the pin outputs high, the level becomes half of the VCC, and when the pin outputs low, the level becomes VSS. In this case the N-channel open drain output pin is pulled up correctly.

When communicating with an external device, communication may not be performed correctly since the high level of the waveform output from the N-channel open drain output pin is half of the VCC.
When checking communication with an external device, remove the pull-down resistor connected to the N-channel open drain output pin.

When the N-channel open drain output pin is not pulled up correctly:
When the pin does not output (during reset), the pin level becomes VSS.
When the pin outputs high, the level becomes VSS, and when the pin outputs low, the level becomes VSS.

Check the circuit if the pin pulled up is correct.
A low signal may be input from the external device. If so, avoid the low output by resetting the external device, for instance, then check the pin level again.

When the pin is not the N-channel open drain output pin:
When the pin does not output (during reset), the pin level becomes VSS.
When the pin outputs high, the level becomes VCC, and when the pin outputs low, the level becomes VSS.

The pin may be the CMOS output pin or switching to the N-channel open drain output pin may fail.
Check if the pin tested is the right pin. Also examine the program whether the setting for N-channel open drain output is specified correctly.

4.7 Investigating the Cause of Interrupt Problems

This section describes a method to investigate what causes an interrupt to be skipped or not to be generated.

4.7.1 Examining the Program Using ICE (E100)

An interrupt may be skipped or may not to be generated in the following cases:

- Address 00000h is read.
- After an interrupt request is generated, the interrupt request is cleared before transition to the interrupt handler is made.

This section introduces the method to examine the program if address 00000h is read.

Procedure

(1) Specify the HEW to have a break point when address 00000h is read.
- Select View >> Event >> Hardware break from the menu bar. The Hardware Break window opens.
- In the Hardware Break window, click the Add button.
- In the Event window, specify the settings as follows and click OK.
  - Event type: Data access
  - Access type: MCU bus, CPU, and DMAC selected
  - Address condition: Specified value (=) Start: 0000h
  - Read/write: Read/Write
- Click the Apply button in the Hardware Break window to apply the settings.

(2) Run the program to see if the break occurs.

Explanation

Check whether the break occurs in step (2) above. If the break occurred, address 00000h is read. Open the trace window and determine the point where address 00000h is read in the program.

5. When the Cause of the Issue Cannot be Found/Determined

5.1 When No Solution Can be Found

When you cannot find any solution for the encountered issue, contact the Renesas distributor or sales representative in your area. The support team will investigate the cause of the issue based on the information provided.

Please provide the following information when making an inquiry:

(1) MCU part number
(2) Goal to be achieved
(3) Issue encountered
(4) Operating frequency (CPU clock)
(5) Frequency of crystals/ceramic resonators connected
(6) Power-supply voltage
(7) Temperature
(8) Reproducibility
(9) Dependencies (voltage dependence, frequency dependence, board dependence)
(10)Number of chips with the issue (pcs/pcs)
(11)Frequency of the issue occurrence (times/hour)
(12)Peripheral function in which the issue occurred
- Communication mode
- Transfer rate

(13)Development phase (development, mass production)
(14)Setting values of related registers
(15)Simulator and emulator information
(16)Compiler version

6. Reference Documents

M16C/63 Group User’s Manual: Hardware Rev.2.20
M16C/64A Group User’s Manual: Hardware Rev.2.10
M16C/64C Group User’s Manual: Hardware Rev.1.10
M16C/65 Group User’s Manual: Hardware Rev.2.10
M16C/65C Group User’s Manual: Hardware Rev.1.10
M16C/6C Group User’s Manual: Hardware Rev.2.10
M16C/5LD Group, M16C/56D Group User’s Manual: Hardware Rev.1.20
M16C/5L Group, M16C/56 Group User’s Manual: Hardware Rev.1.10
M16C/5M Group, M16C/57 Group User’s Manual: Hardware Rev.1.10
The latest versions can be downloaded from the Renesas Electronics website.

Technical Update/Technical News
The latest information can be downloaded from the Renesas Electronics website.

Website and Support

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

Inquiries
http://www.renesas.com/inquiry

Revision History	M16C/63, 64A, 64C, 65, 65C, 6C, 5LD, 56D, 5L, 56, 5M, and 57 Groups Troubleshooting for I^{2}C Mode of Serial Interface in Development

Rev.	Date	Description	
		Page	Summary
1.10	Dec. 1, 2014	—	First edition issued

General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU 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 an MPU or MCU 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.

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2801 Scott Boulevard Santa Clara, CA 95050-2549, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

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
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
Room 1709, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100191, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, P. R. China 200333
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022/9044

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.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics Korea Co., Ltd.
12F., 234 Teheran-ro, Gangnam-Ku, Seoul, 135-920, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 4.0