M3H Group(2)
Application Note
I²C Interface
(I2C-B)
 arbitration

Outlines
This application note is a reference material for developing products using the arbitration function in I2C interface (I2C) function of M3H Group (2).
This document helps the user check operation of the product and develop its program.

Target sample program: I2C_EEPROM_arbitration
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outlines</td>
<td>1</td>
</tr>
<tr>
<td>Table of Contents</td>
<td>2</td>
</tr>
<tr>
<td>1. Preface</td>
<td>3</td>
</tr>
<tr>
<td>2. Reference Document</td>
<td>4</td>
</tr>
<tr>
<td>3. Function to Use</td>
<td>4</td>
</tr>
<tr>
<td>4. Target Device</td>
<td>4</td>
</tr>
<tr>
<td>5. Conditions for Correct Operation</td>
<td>5</td>
</tr>
<tr>
<td>6. Evaluation Board Setting</td>
<td>6</td>
</tr>
<tr>
<td>7. Outline of I²C Interface function</td>
<td>8</td>
</tr>
<tr>
<td>7.1. Clock Supply</td>
<td>8</td>
</tr>
<tr>
<td>8. Sample Program</td>
<td>9</td>
</tr>
<tr>
<td>8.1. Initialization</td>
<td>9</td>
</tr>
<tr>
<td>8.2. Sample program main operation</td>
<td>9</td>
</tr>
<tr>
<td>8.3. Output Example of Sample Program</td>
<td>10</td>
</tr>
<tr>
<td>8.3.1. Setting Example of Terminal Software</td>
<td>11</td>
</tr>
<tr>
<td>8.4. Operating Flow of Sample Program</td>
<td>12</td>
</tr>
<tr>
<td>9. Precaution</td>
<td>21</td>
</tr>
<tr>
<td>10. Revision History</td>
<td>21</td>
</tr>
<tr>
<td>RESTRICTIONS ON PRODUCT USE</td>
<td>22</td>
</tr>
</tbody>
</table>
1. Preface

This sample program is used to check the operation of the I2C communication function. First write data to I2C EEPROM. The function to read the written data via two CPUs is executed by the terminal software on the host PC via USB-UART.

Structure diagram of Sample program
2. Reference Document

- Datasheet
  TMPM3H group (2) datasheet Rev2.0 (Japanese edition)
- Reference manual
  I2C interface (I2C-B) Rev2.0 (Japanese edition)
- Other reference document
  TMPM3H(2) Group Peripheral Driver User Manual (Doxygen)

3. Function to Use

<table>
<thead>
<tr>
<th>IP</th>
<th>Channel</th>
<th>Port</th>
<th>Function / operation mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>I2C interface</td>
<td>ch1</td>
<td>PA4 (I2C1SCL)</td>
<td>I2C mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PA5 (I2C1SDA)</td>
<td></td>
</tr>
<tr>
<td>Asynchronous</td>
<td>ch0</td>
<td>PA1 (UT0TXDA)</td>
<td>UART mode</td>
</tr>
<tr>
<td>communication</td>
<td></td>
<td>PA2 (UT0RXD)</td>
<td></td>
</tr>
<tr>
<td>Input and Output ports</td>
<td>-</td>
<td>PV0 (Input Port)</td>
<td>Input</td>
</tr>
</tbody>
</table>

4. Target Device

The target devices of this application note are as follows;

<table>
<thead>
<tr>
<th>TMPM3HQFDFG</th>
<th>TMPM3HQFZFG</th>
<th>TMPM3HQFYFG</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMPM3HPFDFG</td>
<td>TMPM3HPFZFG</td>
<td>TMPM3HPFYFG</td>
</tr>
<tr>
<td>TMPM3HNDFDG</td>
<td>TMPM3HNFZFG</td>
<td>TMPM3HNFYFG</td>
</tr>
<tr>
<td>TMPM3HNDDFG</td>
<td>TMPM3HNFZDGF</td>
<td>TMPM3HNFYDFG</td>
</tr>
<tr>
<td>TMPM3HMFDFG</td>
<td>TMPM3HMFZFG</td>
<td>TMPM3HMFYFG</td>
</tr>
</tbody>
</table>

* This sample program operates on the evaluation board of TMPM3HQFDFG. If other function than the TMPM3HQ one is checked, it is necessary that CMSIS Core related files (C startup file and I/O header file) should be changed properly.

The BSP related file is dedicated to the evaluation board (TMPM3HQ). If other function than the TMPM3HQ one is checked, the BSP related file should be changed properly.
5. Conditions for Correct Operation

<table>
<thead>
<tr>
<th></th>
<th>TMPM3HQFDFG Evaluation Board (Product of Sensyst)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Used microcontroller</td>
<td>TMPM3HQFDFG</td>
</tr>
<tr>
<td>Used board</td>
<td>TMPM3HQFDFG Evaluation Board (Product of Sensyst)</td>
</tr>
<tr>
<td>Unified development environment</td>
<td>IAR Embedded Workbench for ARM 8.11.2.13606</td>
</tr>
<tr>
<td>Unified development environment</td>
<td>μVision MDK Version 5.24.2.0</td>
</tr>
<tr>
<td>Terminal software</td>
<td>Tera Term V4.96</td>
</tr>
<tr>
<td>Sample program</td>
<td>V1100</td>
</tr>
</tbody>
</table>

For purchasing the board, refer to the following homepage. (http://www.chip1stop.com/)
6. Evaluation Board Setting

Two evaluation boards should be prepared.

Prepare board
Evaluation Board A: TMPM3HQFDFG Evaluation Board
Evaluation Board B: TMPM3HQFDFG Evaluation Board

The following connections should be done.
Note 1: Evaluation Board A
Write the program created with the Default setting of the project for TMPM3HQ.
Connection method details

<table>
<thead>
<tr>
<th>CN12 Board function</th>
<th>Through hole No.</th>
<th>Through hole No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>USB UART conversion</td>
<td>53 : PA1</td>
<td>54 : USB_TXD</td>
</tr>
<tr>
<td>USB UART conversion</td>
<td>55 : PA2</td>
<td>56 : USB_RXD</td>
</tr>
<tr>
<td>I2C (SCL)</td>
<td>57 : PA4</td>
<td>58 : I2C1_SCL</td>
</tr>
<tr>
<td>I2C (SDA)</td>
<td>59 : PA5</td>
<td>60 : I2C1_SDA</td>
</tr>
</tbody>
</table>

Note 2: Evaluation Board B
Write the program created by defining #define BOARD_B in main.c of the project for TMPM3HQ.
Connection method details

<table>
<thead>
<tr>
<th>CN12 Board function</th>
<th>Through hole No.</th>
<th>Through hole No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>USB UART conversion</td>
<td>53 : PA1</td>
<td>54 : USB_TXD</td>
</tr>
<tr>
<td>USB UART conversion</td>
<td>55 : PA2</td>
<td>56 : USB_RXD</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CN5 Board function</th>
<th>Through hole No.</th>
<th>Through hole No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Push SW (S4)</td>
<td>49 : PORT_PSW0</td>
<td>50 : PV0</td>
</tr>
</tbody>
</table>

Connections between the boards
Connect “Board A CN12 No 57” and “Board B CN12 No 57”
Connect “Board A CN12 No 59” and “Board B CN12 No 59”
Connect “Board A CN5 No 50” and “Board B CN5 No 50”
7. Outline of I²C Interface function

The I²C can operate as a transceiver circuit of 1ch (SCL, SDA) in 1 unit circuit.

7.1. Clock Supply

When using I²C, please set a clock enabling bit corresponding with the fsys supply on/off register A ([CGFSYSENA]) or B ([CGFSYSENB]) and fc supply on/off register ([CGFCEN]) as “1” (clock supply). Please refer to “Clock Control and Operation Mode” of the reference manual for the details.
8. Sample Program
When switch (S4) is ON, an event procedure which uses the I2C function is executed. Writing data to the I2C EEPROM via the I2C interface, reading data from the I2C EEPROM, checking the occurrence of the bus busy error, and checking the occurrence of the bus arbitration error. Each event procedure can be checked by displaying its status on the terminal software.

8.1. Initialization
The following initialization is done after power is supplied. The port setting is executed after the initialization of each clock setting, the watchdog timer setting and the clock setting.

8.2. Sample program main operation
Connect board A to read from address 0 and board B to read from address 4. At first, when the switch (S4) on Board B is pushed down, the characters "toshibaABCDEFGHIJKLMNOPQRSTUVWXYZ" are stored to the EEPROM device on Board A through the I2C interface. By pressing switch (S4) on Board B for the second time or later, read the EEPROM data of board A via I2C. The CPU of board A tries to read address 0 and the board B CPU tries to read from address 4.

One of the four following operations is done according to the timing to write the I2C interface after the push-down of the switch (S4) on Board B:
(1) Board A: Read successful. Board B: Read successful.
(2) Board A: Bus busy error. Board B: Read successful.
(3) Board A: Read successful. Board B: Bus busy error.
(4) Board A: Read successful. Board B: Bus arbitration error.
"command >" is displayed on Tera Term at start-up. Then the switch (S4) should be pushed down on Board B to write the data. After that, every push-down of the switch (S4) on Board B executes the read and display of the data.
8.3. Output Example of Sample Program

When the sample program operates, the command results are shown as follows;

**Board A**

command >
write data > toshibaABCDEFGHIJKLMNOPQRST
read data > toshibaABCDEFGHIJKLMNOPQRST
read data >
  bus busy error !!
read data > toshibaABCDEFGHIJKLMNOPQRST
read data > toshibaABCDEFGHIJKLMNOPQRST

**Board B**

command >
write data > toshibaABCDEFGHIJKLMNOPQRST
read data > ibaABCDEFGHIJKLMNOPQRST
read data > ibaABCDEFGHIJKLMNOPQRST
read data >
  bus busy error !!
read data >
  arbitration error !!
8.3.1. Setting Example of Terminal Software

The operation of the terminal software (Tera Term) has been checked with the following settings.

![Tera Term: Serial port setup window](image1)

![Tera Term: Terminal setup window](image2)
8.4. Operating Flow of Sample Program

The operating flows of the sample program are shown in the following:
Creation and Initialization

EEPROM_i2c_init() → bsp_i2c

Port:
I2C1SCL and I2C1SDA settings

CG: FSYSENA
FSYSENA00 (PortA) Enable
FSYSENB12 (I2C) Enable

Register assignment

I2C_init(Instance address)

I2C_get_clock_setting(Instance address)

I2C_reset(Instance address)

I2C_frequency(100KHz)

result = I2C_init(Instance address): Successful

I2C_init(Instance address)
**“write” command procedure**

In the case of “write”,

- EEPROM_i2c_WritePage (Transmission data, Count of Transmission Bytes)

Generation of Transmission data:
- Byte0: Address High Byte
- Byte1: Address Low Byte
- Byte2 to n: Data

- **i2c_write** (Instance handle, Slave address, Transmission data, Count of Transmission Bytes, stop = 1)

**Result**: `strncmp()`

- **Terminal display**: Write data is displayed.
EEPROM(i2c) ReadData
(Reception buffer, Byte count)

Generation of Transmission data
Byte0: Address High Byte
Byte1: Address Low Byte

i2c_write
(Instance handle, Slave address, Transmission data, Count of Transmission Bytes, stop = 0)

Count of Reception Bytes = EEPROM_i2c_ReadData(-):

"read" command procedure

"read" command procedure

EEPROM(i2c) ReadData
(Reception buffer, Byte count)

Generation of Transmission data
Byte0: Address High Byte
Byte1: Address Low Byte

i2c_write
(Instance handle, Slave address, Transmission data, Count of Transmission Bytes, stop = 0)

Count of Reception Bytes = EEPROM_i2c_ReadData(-):

"read" command procedure

In the case of "read":

EEPROM(i2c) ReadData
(Reception buffer, Byte count)

Generation of Transmission data
Byte0: Address High Byte
Byte1: Address Low Byte

i2c_write
(Instance handle, Slave address, Transmission data, Count of Transmission Bytes, stop = 0)

Count of Reception Bytes = EEPROM_i2c_ReadData(-):

"read" command procedure

In the case of "read":

EEPROM(i2c) ReadData
(Reception buffer, Byte count)

Generation of Transmission data
Byte0: Address High Byte
Byte1: Address Low Byte

i2c_write
(Instance handle, Slave address, Transmission data, Count of Transmission Bytes, stop = 0)

Count of Reception Bytes = EEPROM_i2c_ReadData(-):

"read" command procedure

In the case of "read":

EEPROM(i2c) ReadData
(Reception buffer, Byte count)

Generation of Transmission data
Byte0: Address High Byte
Byte1: Address Low Byte

i2c_write
(Instance handle, Slave address, Transmission data, Count of Transmission Bytes, stop = 0)

Count of Reception Bytes = EEPROM_i2c_ReadData(-):

"read" command procedure

In the case of "read":

EEPROM(i2c) ReadData
(Reception buffer, Byte count)

Generation of Transmission data
Byte0: Address High Byte
Byte1: Address Low Byte

i2c_write
(Instance handle, Slave address, Transmission data, Count of Transmission Bytes, stop = 0)

Count of Reception Bytes = EEPROM_i2c_ReadData(-):

"read" command procedure

In the case of "read":

EEPROM(i2c) ReadData
(Reception buffer, Byte count)

Generation of Transmission data
Byte0: Address High Byte
Byte1: Address Low Byte

i2c_write
(Instance handle, Slave address, Transmission data, Count of Transmission Bytes, stop = 0)

Count of Reception Bytes = EEPROM_i2c_ReadData(-):
i2c_check_bus_free

result = :Successful

bus_free=1*
(Only update of the internal information)
i2c_start(Instance handle)

result = i2c_start(); Successful

Start=1*
(Only update of the internal information)
i2c_byte_write(Instance handle and Transmission data)

ACK_result = i2c_byte_write(-):I2C_ACK

I2C_interrupt_request is present.

I2C_int_status(Instance handle)

I2C_get_ack(Instance handle)

ACK_result = I2C_get_ack(-):ACK

I2C_write_data(Instance handle and Transmission data)

I2C_start_condition(Instance handle and Slave address)

I2C_start_condition(-)

I2C_clear_int_status(Instance handle)

I2C_clear_int_status(-):

Bus free check

i2c_active(Instance handle)

i2c_active_status

I2C_start_condition(Instance handle and Slave address)

I2C_slave

I2C_master status

I2C_active

I2C_start(Instance handle and Slave address)

I2C_Master status

CR2 = (I2CxCR2_INIT | I2CxCR2_PIN_CLEAR);

bus busy error

終了

Loop(I2C interrupt request is present)

I2C_int_status(Instance handle)

I2C_interrupt_status=I2C_int_status()
i2c_byte_read

bsp_i2c

i2c

Loop[I2C interrupt request is present]

i2c_clear_int_status(Instance handle)

i2c_write_data(Instance handle, Dummy data)

i2c_clear_int_status(-)

i2c_read_data(Instance handle)

Reception data=i2c_byte_read(-)

Reception data=i2c_read_data(-)

i2c_write_data(-)

i2c_clear_int_status(Instance handle)

I2C Interrupt status=i2c_int_status(-)

I2C Interrupt status=i2c_int_status(-)
The diagram shows the I2C stop function in the context of the i2c_stop process. The steps are as follows:

1. **i2c_stop** is triggered, possibly by an external event or condition.
2. The function `i2c_stop` is called with the instance handle as a parameter.
3. **I2C_stop_condition** is then called with the instance handle as a parameter. This condition checks if the I2C bus is busy.
4. The `I2C_stop_condition` function returns a status.
5. If the I2C is busy, the function `I2C_status_busy` is called with the instance handle as a parameter, indicating the bus is still active.
6. If the I2C is not busy, the function `I2C_stop_condition` is called again, and the process continues until the bus becomes free.
7. Once the bus is free, the `i2c_stop` function completes its operation.
9. Precaution

When using the sample program with CPU other than TMPM3HQ, please check operation sufficiently.

10. Revision History

<table>
<thead>
<tr>
<th>Rev</th>
<th>Date</th>
<th>Page</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>2018-05-17</td>
<td>-</td>
<td>First release</td>
</tr>
</tbody>
</table>
RESTRICTIONS ON PRODUCT USE

Toshiba Corporation and its subsidiaries and affiliates are collectively referred to as “TOSHIBA”. Hardware, software and systems described in this document are collectively referred to as “Product”.

- TOSHIBA reserves the right to make changes to the information in this document and related Product without notice.

- This document and any information herein may not be reproduced without prior written permission from TOSHIBA. Even with TOSHIBA’s written permission, reproduction is permissible only if reproduction is without alteration/omission.

- Though TOSHIBA works continually to improve Product’s quality and reliability, Product can malfunction or fail. Customers are responsible for complying with safety standards and for providing adequate designs and safeguards for their hardware, software and systems which minimize risk and avoid situations in which a malfunction or failure of Product could cause loss of human life, bodily injury or damage to property, including data loss or corruption. Before customers use the Product, create designs including the Product, or incorporate the Product into their own applications, customers must also refer to and comply with (a) the latest versions of all relevant TOSHIBA information, including without limitation, this document, the specifications, the data sheets and application notes for Product and the precautions and conditions set forth in the "TOSHIBA Semiconductor Reliability Handbook" and (b) the instructions for the application with which the Product will be used with or for. Customers are solely responsible for all aspects of their own product design or applications, including but not limited to (a) determining the appropriateness of the use of this Product in such design or applications; (b) evaluating and determining the applicability of any information contained in this document, or in charts, diagrams, programs, algorithms, sample application circuits, or any other referenced documents; and (c) validating all operating parameters for such designs and applications. TOSHIBA ASSUMES NO LIABILITY FOR CUSTOMERS’ PRODUCT DESIGN OR APPLICATIONS.

- PRODUCT IS NEITHER INTENDED NOR WARRANTED FOR USE IN EQUIPMENTS OR SYSTEMS THAT REQUIRE EXTRAORDINARILY HIGH LEVELS OF QUALITY AND/OR RELIABILITY, AND/OR A MALFUNCTION OR FAILURE OF WHICH MAY CAUSE LOSS OF HUMAN LIFE, BODILY INJURY, SERIOUS PROPERTY DAMAGE AND/OR SERIOUS PUBLIC IMPACT (“UNINTENDED USE”). Except for specific applications as expressly stated in this document, Unintended Use includes, without limitation, equipment used in nuclear facilities, equipment used in the aerospace industry, medical equipment, equipment used for automobiles, trains, ships and other transportation, traffic signaling equipment, equipment used to control combustions or explosions, safety devices, elevators and escalators, devices related to electric power, and equipment used in finance-related fields. IF YOU USE PRODUCT FOR UNINTENDED USE, TOSHIBA ASSUMES NO LIABILITY FOR PRODUCT. For details, please contact your TOSHIBA sales representative.

- Product shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable laws or regulations.

- The information contained herein is presented only as guidance for Product use. No responsibility is assumed by TOSHIBA for any infringement of patents or any other intellectual property rights of third parties that may result from the use of Product. No license to any intellectual property right is granted by this document, whether express or implied, by estoppel or otherwise.

- ABSENT A WRITTEN SIGNED AGREEMENT, EXCEPT AS PROVIDED IN THE RELEVANT TERMS AND CONDITIONS OF SALE FOR PRODUCT, AND TO THE MAXIMUM EXTENT ALLOWABLE BY LAW, TOSHIBA (1) ASSUMES NO LIABILITY WHATSOEVER, INCLUDING WITHOUT LIMITATION, INDIRECT, CONSEQUENTIAL, SPECIAL, OR INCIDENTAL DAMAGES OR LOSS, INCLUDING WITHOUT LIMITATION, LOSS OF PROFITS, LOSS OF OPPORTUNITIES, BUSINESS INTERRUPTION AND LOSS OF DATA, AND (2) DISCLAIMS ANY AND ALL EXPRESS OR IMPLIED WARRANTIES AND CONDITIONS RELATED TO SALE, USE OF PRODUCT, OR INFORMATION, INCLUDING WARRANTIES OR CONDITIONS OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, ACCURACY OF INFORMATION, OR NONINFRINGEMENT.

- Do not use or otherwise make available Product or related software or technology for any military purposes, including without limitation, this design, development, use, stockpiling or manufacturing of nuclear, chemical, or biological weapons or missile technology products (mass destruction weapons). Product and related software and technology may be controlled under the applicable export laws and regulations including, without limitation, the Japanese Foreign Exchange and Foreign Trade Law and the U.S. Export Administration Regulations. Export and re-export of Product or related software or technology are strictly prohibited except in compliance with all applicable export laws and regulations.

- Please contact your TOSHIBA sales representative for details as to environmental matters such as the RoHS compatibility of Product. Please use Product in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. TOSHIBA ASSUMES NO LIABILITY FOR DAMAGES OR LOSSES OCCURRING AS A RESULT OF NONCOMPLIANCE WITH APPLICABLE LAWS AND REGULATIONS.