MT793X IoT SDK for I2C Master User Guide

Version: 1.0
Release date: 2021-08-02

Use of this document and any information contained therein is subject to the terms and conditions set forth in Exhibit 1. This document is subject to change without notice.

Version History

Version	Date	Description
1.0	2021-08-02	Official release

Table of Contents

1 I2C Master

1.1 Introduction

There are two I2C master channels in the MT793X with the same hardware architecture. I2C (Inter-Integrated Circuit) is a two-wire serial interface with two signals, SCL and SDA. SCL (Serial Clock Line) is a clock signal driven by the master, and SDA (Serial Data Line) is a bi-directional data signal that can be driven either by the master or by the slave. This generic controller supports the master role and conforms to the I2C specification.

1.2 Features

The main supported features of I2C master are as follows:

- I2C compliant master mode operation
- Adjustable clock speed for Fast-mode Plus
- 7-bit address
- Clock stretching feature
- START/STOP/repeated START conditions
- I2C_FIFO mode
- DMA transfer mode
- Multi-write per transfer
- Multi-read per transfer
- Multi-transfer per transaction
- Combined format transfer with length change capability
- Multi-transfer with repeated START condition

2 Driver Introduction

2.1 Driver API Reference

hal_i2c_status_t	**hal_i2c_master_init** (**hal_i2c_port_t** i2c_port, **hal_i2c_config_t** *i2c_config) This function initializes the I2C master before starting a transaction.
hal_i2c_status_t	**hal_i2c_master_deinit** (**hal_i2c_port_t** i2c_port) This function releases the I2C master after the transaction is over.
hal_i2c_status_t	**hal_i2c_master_set_frequency** (**hal_i2c_port_t** i2c_port, **hal_i2c_frequency_t** frequency) This function sets the transaction speed.
hal_i2c_status_t	**hal_i2c_master_register_callback** (**hal_i2c_port_t** i2c_port, **hal_i2c_callback_t** i2c_callback, void *user_data) This function registers a callback function while using DMA mode.
hal_i2c_status_t	**hal_i2c_master_send_polling** (**hal_i2c_port_t** i2c_port, uint8_t slave_address, const uint8_t *data, uint32_t size) This function sends data to I2C slave in polling mode.
hal_i2c_status_t	**hal_i2c_master_send_dma** (**hal_i2c_port_t** i2c_port, uint8_t slave_address, const uint8_t *data, uint32_t size) This function sends data to I2C slave in DMA mode.
hal_i2c_status_t	**hal_i2c_master_receive_polling** (**hal_i2c_port_t** i2c_port, uint8_t slave_address, uint8_t *buffer, uint32_t size) This function receives data from I2C slave in a polling mode.
hal_i2c_status_t	**hal_i2c_master_receive_dma** (**hal_i2c_port_t** i2c_port, uint8_t slave_address, uint8_t *buffer, uint32_t size) This function receives data from I2C slave in a DMA mode.
hal_i2c_status_t	**hal_i2c_master_send_to_receive_polling** (**hal_i2c_port_t** i2c_port, **hal_i2c_send_to_receive_config_t** *i2c_send_to_receive_config) This function sends data to and then receives data from I2C slave in a polling mode.
hal_i2c_status_t	**hal_i2c_master_send_to_receive_dma** (**hal_i2c_port_t** i2c_port, **hal_i2c_send_to_receive_config_t** *i2c_send_to_receive_config) This function sends data to and then receives data from I2C slave in a DMA mode.
hal_i2c_status_t	**hal_i2c_master_get_running_status** (**hal_i2c_port_t** i2c_port, **hal_i2c_running_status_t** *running_status) This function gets running status of the I2C master.

1.1. Sample Code

2.1.1 Using I2C Master Polling Mode

- Step1: Call **hal_gpio_init()** to initialize the pin. For mode details about hal_gpio_init please refer to GPIO module in HAL.
- Step2: Call **hal_pinmux_set_function()** to set the GPIO pinmux or use the EPT tool to apply the pinmux settings.
- Step3: Call **hal_i2c_master_init()** to initialize the I2C master.
- Step4: Call **hal_i2c_master_send_polling()** to send data in a polling mode.
- Step5: Call **hal_i2c_master_receive_polling()** to receive data in a polling mode.
- Step6: Call **hal_i2c_master_deinit()** to de-allocate the I2C master if it is no longer in use.

```
hal_i2c_config_t i2c_config;
uint8_t slave_address = 0X50;
const uint8_t send_data[8] = {0x00, 0x00, 0xFF, 0xAA, 0x14, 0x15, 0x16, 0x17, 0x18, 0x19}; //the data that used to send
uint8_t receive_data[8] = {0};
volatile uint32_t size = 8;
i2c_config.frequency = HAL_I2C_FREQUENCY_400K;

hal_gpio_init(gpio_pin); //init the gpio pin
hal_pinmux_set_function(gpio_pin, function_index);//Set the pinmux
if(HAL_I2C_STATUS_OK == hal_i2c_master_init(HAL_I2C_MASTER_0, &i2c_config)) {
    //Send data
    if (HAL_I2C_STATUS_OK != hal_i2c_master_send_polling(HAL_I2C_MASTER_0, slave_address, send_data, size)) {
        //Error handler;
    }
    //Receive data
    if (HAL_I2C_STATUS_OK != hal_i2c_master_receive_polling(HAL_I2C_MASTER_0, slave_address, receive_data, size)) {
        //Error handler;
    }
    hal_i2c_master_deinit(HAL_I2C_MASTER_0); //Call this function if the I2C is no longer in use.

} else {
    //Error handler;
}
```

2.1.2 Using I2C Master DMA Mode

- Step1: Call **hal_gpio_init()** to initialize the pin. For mode details about hal_gpio_init please refer to GPIO module in HAL.
- Step2: Call **hal_pinmux_set_function()** to set the GPIO pinmux or use the EPT tool to apply the pinmux settings.
- Step3: Call **hal_i2c_master_init()** to initialize the I2C master.
- Step4: Call **hal_i2c_master_register_callback()** to register a user callback.
- Step5: Call **hal_i2c_master_send_dma()** to send data within a DMA mode.
- Step6: Call **hal_i2c_master_receive_dma()** to receive data in a DMA mode.
- Step7: Call **hal_i2c_master_deinit()** to de-initialize the I2C master if it is no longer in use.

```
hal_i2c_config_t i2c_config;
uint8_t slave_address = 0X50;
const uint8_t send_data[8] = {0x00, 0x00, 0xFF, 0xAA, 0x14, 0x15, 0x16, 0x17, 0x18, 0x19}; //the data that used to send
uint8_t receive_data[8] = {0};
volatile uint32_t size = 8;
i2c_config.frequency = HAL_I2C_FREQUENCY_400K;

hal_gpio_init(gpio_pin); //init the gpio pin
hal_pinmux_set_function(gpio_pin, function_index);//Set the pinmux
if(HAL_I2C_STATUS_OK == hal_i2c_master_init(HAL_I2C_MASTER_0, &i2c_config)) {
    hal_i2c_master_register_callback(HAL_I2C_MASTER_0, i2c_user_callback, (void *) &user_data)//Register a user callback.
    //Send data
    if (HAL_I2C_STATUS_OK != hal_i2c_master_send_dma(HAL_I2C_MASTER_0, slave_address, send_data, size)) {
        //Error handler;
    }
    //Receive data
    if (HAL_I2C_STATUS_OK != hal_i2c_master_receive_dma(HAL_I2C_MASTER_0, slave_address, receive_data, size)) {
        //Error handler;
    }
} else {
    //Error handler;
}
//hal_i2c_master_deinit(i2c_port);// Call this function if the I2C is no longer in use.
// Callback function sample code. This function should be passed to driver while calling the function #hal_i2c_master_register_callback().
void i2c_user_callback (uint8_t slave_address,hal_i2c_callback_event_t event,void *user_data)
{
    if(HAL_I2C_EVENT_ACK_ERROR == event) {
        //ACK error handler;
    } else if(HAL_I2C_EVENT_NACK_ERROR == event) {
        //NACK error handler;
    } else if(HAL_I2C_EVENT_TIMEOUT_ERROR == event) {
        // Timeout handler;
    } else if(HAL_I2C_EVENT_SUCCESS == event) {
        // Do something;
        // The slave_address indicates which device is using I2C.
    }
}
```

Exhibit 1 Terms and Conditions

Your access to and use of this document and the information contained herein (collectively this "Document") is subject to your (including the corporation or other legal entity you represent, collectively "You") acceptance of the terms and conditions set forth below ("T&C"). By using, accessing or downloading this Document, You are accepting the T&C and agree to be bound by the T&C. If You don't agree to the T&C, You may not use this Document and shall immediately destroy any copy thereof.

This Document contains information that is confidential and proprietary to MediaTek Inc. and/or its affiliates (collectively "MediaTek") or its licensors and is provided solely for Your internal use with MediaTek's chipset(s) described in this Document and shall not be used for any other purposes (including but not limited to identifying or providing evidence to support any potential patent infringement claim against MediaTek or any of MediaTek's suppliers and/or direct or indirect customers). Unauthorized use or disclosure of the information contained herein is prohibited. You agree to indemnify MediaTek for any loss or damages suffered by MediaTek for Your unauthorized use or disclosure of this Document, in whole or in part.

MediaTek and its licensors retain titles and all ownership rights in and to this Document and no license (express or implied, by estoppels or otherwise) to any intellectual propriety rights is granted hereunder. This Document is subject to change without further notification. MediaTek does not assume any responsibility arising out of or in connection with any use of, or reliance on, this Document, and specifically disclaims any and all liability, including, without limitation, consequential or incidental damages.

THIS DOCUMENT AND ANY OTHER MATERIALS OR TECHNICAL SUPPORT PROVIDED BY MEDIATEK IN CONNECTION WITH THIS DOCUMENT, IF ANY, ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE. MEDIATEK SPECIFICALLY DISCLAIMS ALL WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, FITNESS FOR A PARTICULAR PURPOSE, COMPLETENESS OR ACCURACY AND ALL WARRANTIES ARISING OUT OF TRADE USAGE OR OUT OF A COURSE OF DEALING OR COURSE OF PERFORMANCE. MEDIATEK SHALL NOT BE RESPONSIBLE FOR ANY MEDIATEK DELIVERABLES MADE TO MEET YOUR SPECIFICATIONS OR TO CONFORM TO A PARTICULAR STANDARD OR OPEN FORUM.

Without limiting the generality of the foregoing, MediaTek makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does MediaTek assume any liability arising out of the application or use of any product, circuit or software. You agree that You are solely responsible for the designing, validating and testing Your product incorporating MediaTek's product and ensure such product meets applicable standards and any safety, security or other requirements.

The above T&C and all acts in connection with the T&C or this Document shall be governed, construed and interpreted in accordance with the laws of Taiwan, without giving effect to the principles of conflicts of law.