MT793X IoT SDK for SPI Slave User Guide

Version: 1.0
Release date: 2021-07-01

Version History

Version	Date	Description
1.0	2021-08-02	Official release

Table of Contents

List of Figures

List of Tables

1 SPI Slave

1.1 Introduction

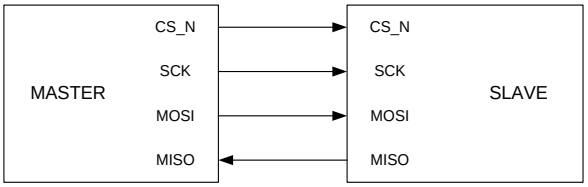

Figure 1-1. Pin connection between SPI master and SPI slave

The SPI interface is a bit-serial, four-pin transmission protocol. Figure 1-1 is an example of the connection between the SPI master and SPI slave.

1.2 Pin Description

Table 1-1. SPI controller interface

Signal name	Type	Description
CS_N	O	Low active chip selection signal
SCK	O	The (bit) serial clock
MOSI	O	Data signal from master output to slave input
MISO	I	Data signal from slave output to master input

1.3 Features

The features of the SPI controller (slave) are listed below:

- The supported SPI_CLK is up to 25 MHz.
- Configurable bit transmitting and receiving order: Two options of bit order – MSB or LSB first
- Four communication modes are available, MODE 0, 1, 2, 3, which basically define the SCLK edge on which the MISO line toggles and the slave samples the MOSI line. They also define the SCLK signal steady level (namely, the clock level, high or low, when the clock is not active). Each mode is formally defined with a pair of parameters called “clock polarity” (CPOL) and “clock phase” (CPHA).

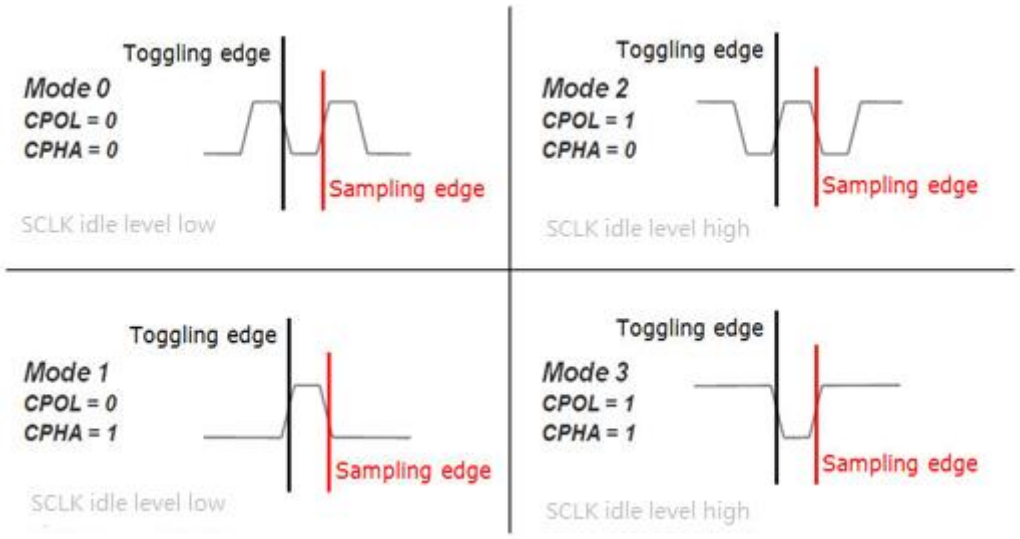

Figure 1-2 Four communication modes waveform

- Enable/Disable Transmit and Receive mode
- Default Tx FIFO data (default value is 0x00): If Tx FIFO is empty and the SPI master wants to get the data from the SPI slave, then the SPI slave outputs configurable constant byte value on MISO, and the default value is 0x00.
- Rx/Tx FIFO data status: Rx/Tx FIFO pointer and number of bytes are transmitted/received in status register. They can be read for status checking.
- Interrupt support: RX full interrupt and transfer done interrupt are used for indication.
- Support PIO mode and DMA mode transfer: Both DMA and PIO modes are supported on SPI slave Tx/Rx channel. (Note: Under DMA mode, the value of SPIS_TX_SRC (SPISn Base address+0x0020)[31:0] and SPIS_RX_DST(SPISn Base address+0x001C)[31:0] must be 4-byte aligned)
- Programmable byte length for transmission: The length of Tx DMA can be programmable from 1 to 1 M bytes.
- Supports Rx DMA byte length from 1 to (1 M-4) bytes.
- Each FIFO depth of Tx/Rx is 32 x 4 bytes.

2 Driver Introduction

2.1 Driver API Reference

API	Description
hal_spi_slave_status_t hal_spi_slave_init(hal_spi_slave_port_t spi_port, hal_spi_slave_config_t *spi_configure)	This function initializes the SPI slave and sets user defined common parameters
hal_spi_slave_status_t hal_spi_slave_deinit(hal_spi_slave_port_t spi_port)	This function resets the SPI slave, gates its clock and disables interrupts.
hal_spi_slave_status_t hal_spi_slave_register_callback(hal_spi_slave_port_t spi_port, hal_spi_slave_callback_t callback_function, void *user_data)	This function registers user's callback in the SPI slave driver.
hal_spi_slave_status_t hal_spi_slave_send_and_receive(hal_spi_slave_port_t spi_port, const uint8_t *tx_buf, uint8_t *rx_buf,uint32_t size)	This function sends data asynchronously with DMA mode, this function should be called from user's callback function.
hal_spi_slave_status_t hal_spi_slave_receive(hal_spi_slave_port_t spi_port, uint8_t *buffer, uint32_t size)	This function receives data asynchronously with DMA mode.

2.2 Sample Code

- Step 1. Call **hal_gpio_init()** to init the pins, if EPT tool hasn't been used to configure the related pinmux.
- Step 2. Call **hal_pinmux_set_function()** to set the GPIO pinmux, if the EPT tool hasn't been used to configure the related pinmux. For more details about **hal_pinmux_set_function()**, please refer to **GPIO**.
- Step 3. Call **hal_spi_slave_init()** to initialize one SPI slave.
- Step 4. Call **hal_spi_slave_register_callback()** to register a user callback.
- Step 5. Call **hal_spi_slave_receive()** to receive data.
- Step 6. Call **hal_spi_slave_send_and_receive()** to send and receive data.
- Step 7. Call **hal_spi_slave_deinit()** to deinitialize the SPI slave that is no longer in use.

- sample code:

```
uint16_t slave_status;
hal_spi_slave_config_t spi_configure;
uint8_t *data, *buffer;
uint32_t size;
```

```
// Initialize the GPIO, set GPIO pinmux (if EPT tool hasn't been used to configure the related pinmux).
hal_gpio_init(HAL_GPIO_25);
hal_gpio_init(HAL_GPIO_26);
hal_gpio_init(HAL_GPIO_27);
hal_gpio_init(HAL_GPIO_28);
// Call hal_pinmux_set_function() to set GPIO pinmux, for more information, please refer to hal_pinmux_define.h
// Not need to configure the pinmux, if EPT is used.
hal_pinmux_set_function(HAL_GPIO_25, 7);// Set the pin to be used as SCK signal of SPI Slave.
hal_pinmux_set_function(HAL_GPIO_26, 7);// Set the pin to be used as CS signal of SPI Slave.
hal_pinmux_set_function(HAL_GPIO_27, 7);// Set the pin to be used as MOSI signal of SPI Slave.
hal_pinmux_set_function(HAL_GPIO_28, 5);// Set the pin to be used as MISO signal of SPI Slave.
```

```
spi_configure.bit_order = HAL_SPI_SLAVE_LSB_FIRST;
spi_configure.phase = HAL_SPI_SLAVE_CLOCK_PHASE0;
spi_configure.polarity = HAL_SPI_SLAVE_CLOCK_POLARITY0;
spi_configure.timeout_threshold = 0xFFFFFFFF;
if (HAL_SPI_SLAVE_STATUS_OK != hal_spi_slave_init(HAL_SPI_SLAVE_0, &spi_configure)) {
    // Error handler;
}
if (HAL_SPI_SLAVE_STATUS_OK != hal_spi_slave_register_callback(HAL_SPI_SLAVE_0, user_spi_slave_callback, NULL)) {
    // Error handler;
}

if (HAL_SPI_SLAVE_STATUS_OK != hal_spi_slave_receive(HAL_SPI_SLAVE_0, buffer, size)) {
    // Error handler;
}

if (HAL_SPI_SLAVE_STATUS_OK != hal_spi_slave_send_and_receive(HAL_SPI_SLAVE_0, data, buffer, size)) {
    // Error handler;
}

hal_spi_slave_deinit(HAL_SPI_SLAVE_0);
```

```
// Callback function sample code. Pass this function to the driver, while calling #hal_spi_slave_register_callback().
hal_spi_slave_status_t user_spi_slave_callback (hal_spi_slave_transaction_status_t status,void *user_data)
{
 //interrupt handler;
}
```

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.