SigmaStar Camera
SPI 使用参考

REVISION HISTORY

Revision No.	Description	Date
{000001}	• {Initial release}	{07/28/2018}

TABLE OF CONTENTS

1. 概述

1.1. 概述

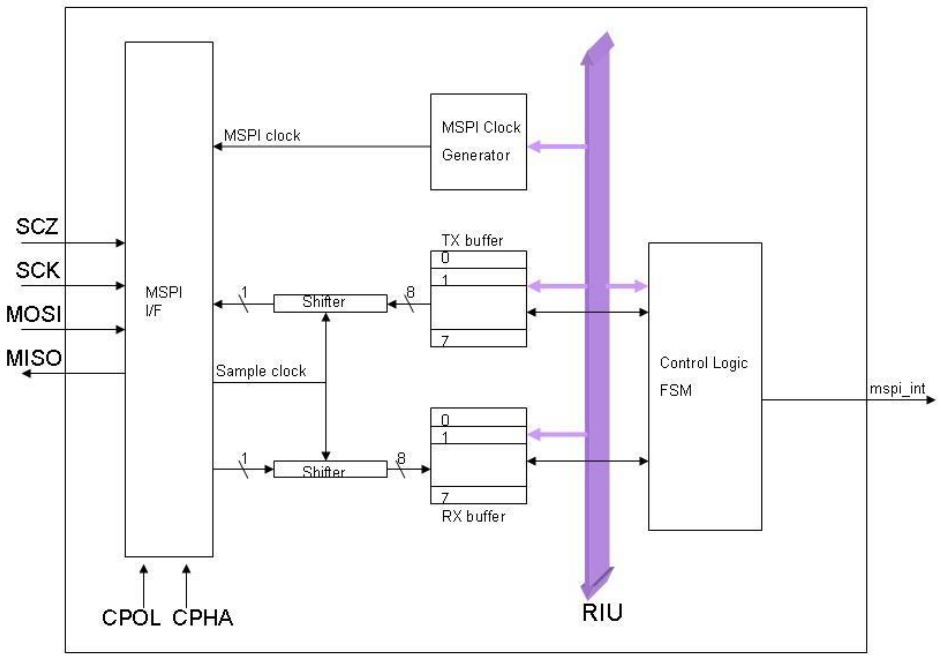
This document describes the master SPI (MSPI) controller. This MSPI is a synchronous serial interface and can connect to a variety of external device.

1.2. Features

- ❑ Generic SPI protocol with half duplex.
- ❑ Supports Motorola SPI compatible timing. (CPHA/CPOL)
- ❑ 8 Byte write buffer and 8-Byte read buffer.
- ❑ Configurable Bit width from one bit to 8bits in a byte transfer.
- ❑ Supports up to 8 slave device select signals.
- ❑ Supports 3-wire mode.
- ❑ Supports an internal RIU (Register Interface Unit) interface. RIU is an in-house protocol of SigmaStar.

1.3. Block Diagram

MSPI Block Diagram



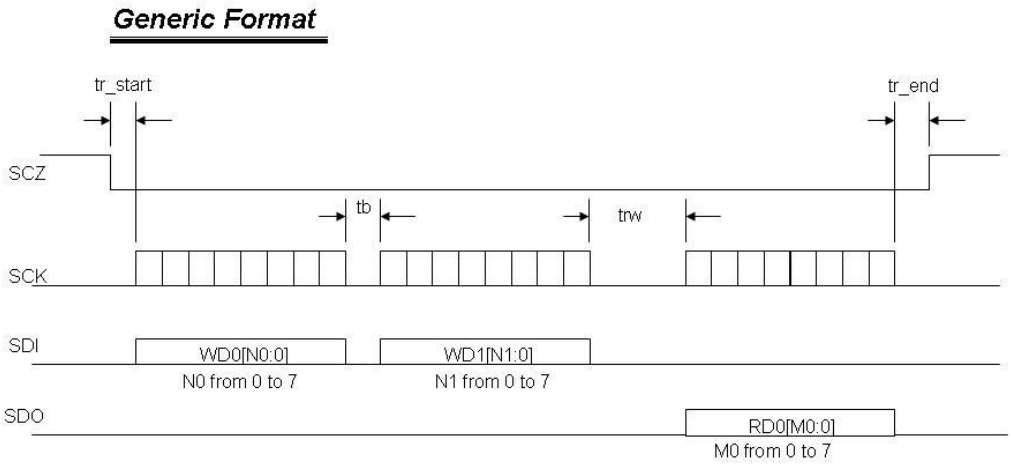
2. SPI 控制

2.1. SPI 控制概述

This section describes which registers should be initialized before enabling MSPI, and how to trigger MSPI and which status should be checked.

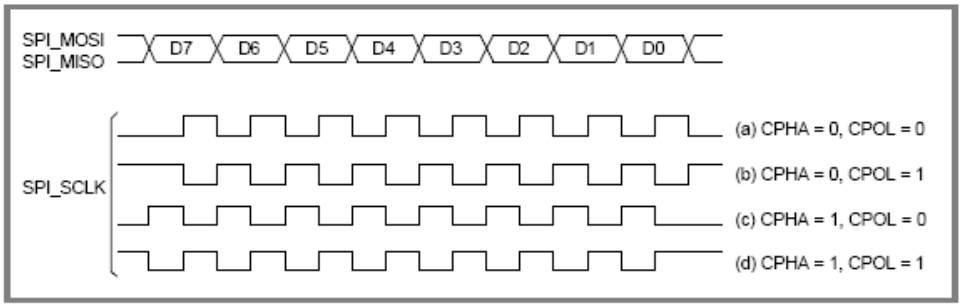
2.2. Initialization

MSPI supports some configurable parameters like below.



(1). DC Timing Setting
"tr_start": SCZ active setup time (relative to SCK)
"tr_end": SCZ active hold time (relative to SCK)
"tb" : The delay cycle between frame transfer
"trw": The delay cycle between last write and first read. (Read turn around cycle)
(2). Write/Read Frame Length Setting, (0x111098, 0x11109C)
Frame Length from 0 ~ 8 Bytes
(3). Read/ Write Bit Length Setting per Frame, (0x111098, 0x11109C)
Bit Length from 1 ~ 8 bit
(4). Clock & Phase Control, (0x111092)
Clock Setting: 3'b000: CPU_CLOCK/2
3'b001: CPU_CLOCK/4
3'b010: CPU_CLOCK/8
3'b011: CPU_CLOCK/16
3'b100: CPU_CLOCK/32
3'b101: CPU_CLOCK/64

3'b110: CPU_CLOCK/128
3'b111: CPU_CLOCK/256
Bit[7]: Clock Polarity, CPOL.
0: The SCK is set to 0 in idle state
1: The SCK is set to 1 in idle state
Bit[6]: Clock Phase, CPHA
0: Date is sampled when the SCK leaves the idle state.
1: Date is sampled when the SCK returns to idle



state.
(5). Set Reset & Enable. (0x111092)
Bit[2]: Enable MSPI interrupt
0: Disable
1: Enable
Bit[1]: Reset
0: Reset
1: Not Reset
Bit[0]: Enable MSPI
0: Disable
1: Enable
(6). Enable chip select. (0x1110BE)
Chip_select for SPI Device1
0: Enable
1: Disable

2.3. Trigger

MSPI Trigger. (0x1110BE, Bit[0])
When this bit is set, MSPI controller will perform operation.
This bit is write-clear register

2.4. MSPI Status

MSPI completed Flag, (0x1110B6, Bit[0])
(1). When MSPI operation is completed, HW set this bit as high, before starting to next data transfer , SW needs to clear this bit via (0x1110A2, Bit[0]),
(2). When MSPI operation is completed, MSPI also issue an interrupt to CPU, SW needs to clear interrupt via (0x1110B8, Bit[0]).

2.5. MSPI Read Data Port

When MSPI operation is completed, SW can get data from the below read port.

0x111088
0x11108A
0x11108C
0x11108E

3. EXAMPLE

3.1. SPI Operation Example

This section describes an example of MSPI operation.

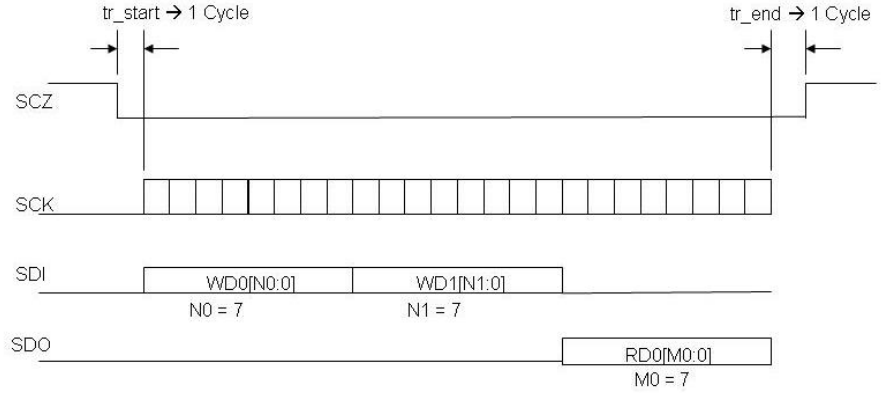
(0). Initial

(1). CS goes low

(2). Write 2Bytes data

(3). Read 1Bytes data

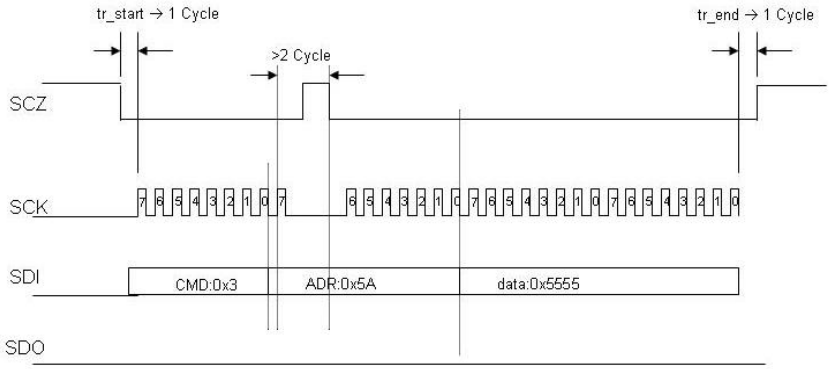
(4). CS goes high




```
WREG[0x111092] = 0x07                    //Enable MSPI, Not Reset, Enable INT
                                        //CPOL =0, CPHA=0
WREG[0x111093] = 0x00                    //MSPI clock = CPU Clock/2
WREG[0x111094] = 0x00                    //tr_start = 1 Cycle
WREG[0x111095] = 0x00                    //tr_end = 1 Cycle
WREG[0x111096] = 0x00                    //tb = 0 cycle
WREG[0x111097] = 0x00                    //trw = 0 Cycle
WREG[0x111080] = DATA0                    //Write buffer0
WREG[0x111081] = DATA1                    //Write buffer1
WREG[0x111090] = 0x02                    //Write length = 2
WREG[0x111091] = 0x01                    //Read length = 1
WREG[0x111098], BIT[2:0] = 3'b111        //Bit length=8Bit for write buffer0
                BIT[5:3] = 3'b111       //Bit length=8Bit for write buffer1
WREG[0x11109c], BIT[2:0] = 3'b111        //Bit length=8Bit for read buffer0
WREG[0x1110BE], BIT[0] = 1'b0            //Select cs0
WREG[0x1110B4], BIT[0] = 1'b1            //Trigger MSPI operation
Done = RREG[0x08B6, Bit[0]]             //Polling MSPI done flag Or interrupt
WREG[0x1110B8], BIT[0] = 1'b1            //Clear done flag or clear interrupt
WREG[0x1110BE], BIT[0] = 1'b1            //Disable cs0
```

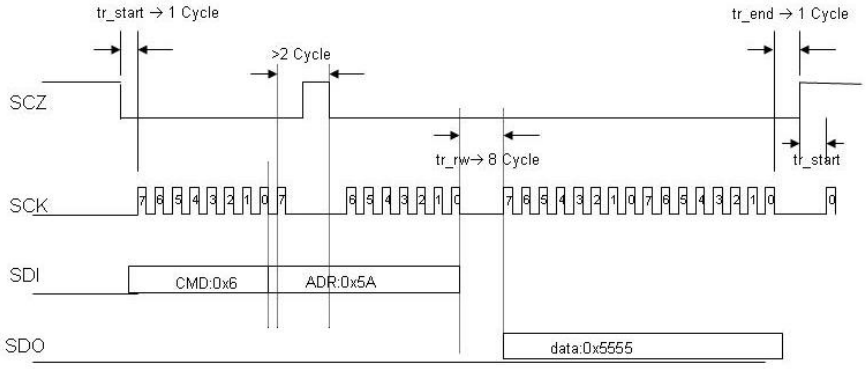
Example code for MSPI Connect to Incense

- Register Write to Incense: WREG[0x5a] = 16'h5555




```
WREG[0x111092] = 0x07                    //Enable MSPI, Not Reset, Enable INT
                                        //CPOL =0, CPHA=0
WREG[0x111093] = 0x00                    //MSPI clock = CPU Clock/2
WREG[0x111094] = 0x00                    //tr_start = 1 Cycle
WREG[0x111095] = 0x01                    //tr_end = 2 Cycle
WREG[0x111096] = 0x00                    //tb = 0 cycle
WREG[0x111097] = 0x00                    //trw = 0 Cycle
WREG[0x111080] = 0x03 (CMD)              //Write buffer0
WREG[0x111081] = 0x5A (ADR)              //Write buffer1
WREG[0x111090] = 0x02                    //Write length = 2
WREG[0x111098], BIT[2:0] = 3'b111        //Bit length=8Bit for write buffer0
                BIT[5:3] = 3'b000        //Bit length=1Bit for write buffer1
WREG[0x1110BE], BIT[0] = 1'b0            //Select cs0
WREG[0x1110B4], BIT[0] = 1'b1            //Trigger MSPI operation
Done = RREG[0x1110B6, Bit[0]]            //Polling MSPI done flag Or interrupt
WREG[0x1110B8], BIT[0] = 1'b1            //Clear done flag or clear interrupt
WREG[0x1110BE], BIT[0] = 1'b1            //Disable cs0
ADR = ADR << 1;
WREG[0x111080] = ADR                     //Write buffer0
WREG[0x111081] = 0x55(DATA0)             //Write buffer1
WREG[0x111082] = 0x55(DATA1)             //Write buffer2
WREG[0x111090] = 0x03                    //Write length = 3
WREG[0x111098], BIT[2:0] = 3'b110        //Bit length=7Bit for write buffer0
                BIT[5:3] = 3'b111        //Bit length=8Bit for write buffer1
                BIT[8:6] = 3'b111        //Bit length=8Bit for write buffer2
WREG[0x1110BE], BIT[0] = 1'b0            //Select cs0
WREG[0x1110B4], BIT[0] = 1'b1            //Trigger MSPI operation
Done = RREG[0x1110B6, Bit[0]]            //Polling MSPI done flag Or interrupt
WREG[0x1110B8], BIT[0] = 1'b1            //Clear done flag or clear interrupt
WREG[0x1110BE], BIT[0] = 1'b1            //Disable cs0
```

- Register Read from Incense: RREG[0x5a]




```
WREG[0x111092] = 0x07                    //Enable MSPI, Not Reset, Enable INT
                                        //CPOL =0, CPHA=0
WREG[0x111093] = 0x00                    //MSPI clock = CPU Clock/2
WREG[0x111094] = 0x00                    //tr_start = 1 Cycle
WREG[0x111095] = 0x01                    //tr_end = 2 Cycle
WREG[0x111096] = 0x00                    //tb = 0 cycle
WREG[0x111097] = 0x00                    //trw = 0 Cycle
WREG[0x111080] = 0x06 (CMD)              //Write buffer0
WREG[0x111081] = 0x5A (ADR)              //Write buffer1
WREG[0x111090] = 0x02                    //Write length = 2
WREG[0x111098], BIT[2:0] = 3'b111        //Bit length=8Bit for write buffer0
                BIT[5:3] = 3'b000        //Bit length=1Bit for write buffer1
WREG[0x1110BE], BIT[0] = 1'b0            //Select cs0
WREG[0x1110B4], BIT[0] = 1'b1            //Trigger MSPI operation
Done = RREG[0x1110B6, Bit[0]]            //Polling MSPI done flag Or interrupt
WREG[0x1110B8], BIT[0] = 1'b1            //Clear done flag or clear interrupt
WREG[0x1110BE], BIT[0] = 1'b1            //Disable cs0

ADR = ADR << 1;
WREG[0x111080] = ADR                     //Write buffer0
WREG[0x111090] = 0x01                    //Write length = 1
WREG[0x111098], BIT[2:0] = 3'b110        //Bit length=7Bit for write buffer0
WREG[0x1110BE], BIT[0] = 1'b0            //Select cs0
WREG[0x1110B4], BIT[0] = 1'b1            //Trigger MSPI operation
Done = RREG[0x1110B6, Bit[0]]            //Polling MSPI done flag Or interrupt
WREG[0x1110B8], BIT[0] = 1'b1            //Clear done flag or clear interrupt

WREG[0x111092], BIT[7:6]= 2'b01          //CPOL =0, CPHA=1
                                        //Change mode for read data
WREG[0x111094] = 0x08                    //tr_start = 8 Cycle
                                        //In this case trw = tr_start
WREG[0x111090] = 0x00                    //Write length = 0
WREG[0x111091] = 0x02                    //Read length = 2
WREG[0x1110B4], BIT[0] = 1'b1            //Trigger MSPI operation
```

```
Done = RREG[0x1110B6, Bit[0]]                 //Polling MSPI done flag Or interrupt
WREG[0x1110B8], BIT[0] = 1'b1                 //Clear done flag or clear interrupt
WREG[0x1110BE], BIT[0] = 1'b1                 //Disable cs0

WREG[0x111091] = 0x01                         //Read length = 1
                                              //Dummy read
WREG[0x11109c], BIT[2:0] = 3'b000             //Bit length=1Bit for read buffer0
WREG[0x1110B4], BIT[0] = 1'b1                 //Trigger MSPI operation
Done = RREG[0x1110B6, Bit[0]]                 //Polling MSPI done flag Or interrupt
WREG[0x1110B8], BIT[0] = 1'b1                 //Clear done flag or clear interrupt
```