

文档版本	V1.0
发布日期	20210609

APT32S003 SPI 应用指南

APTCHIP



目录

1 概述	1
2. 适用的硬件.....	1
3. 应用方案代码说明	1
3.1 SPI 配置.....	1
3.2 SPI 主机配置.....	3
3.3 SPI 从机配置.....	5
4. 程序下载和运行	8

1 概述

本文介绍了在APT32S003中使用SPI的应用范例。

2. 适用的硬件

该例程使用于 APT32S003 系列学习板

3. 应用方案代码说明

3.1 SPI 配置

基于 APT32S003 完整的库文件系统，可以对 SPI 进行配置。

● 硬件配置：

SPI 是可以配置为主机或者从机接口模块，可以用来跟其它外设进行同步串行通讯，发送和接收都有一个 16 位宽，8 地址深的 FIFO。具有发送和接收中断以及溢出中断，支持内部环回测试模式。

注意 SPI 的主/从模式通信，首先硬件接线正确，CS-CS，CLK-CLK，MISO，MOSI。主机模式频率和从机模式频率要参数一致，比如波特率、数据长度。

1 主 1 从连接方式：

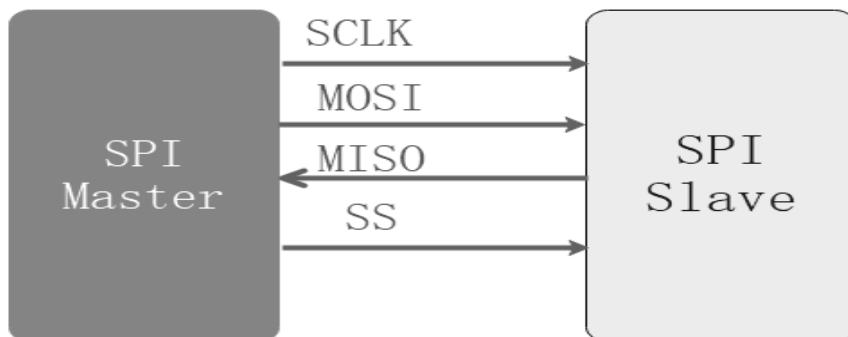


图 3.1.1 SPI 1 主 1 从定义

1 主多从连接方式:

时钟、数据脚都并接，主机通过 CS 脚来控制不同的设备使能。

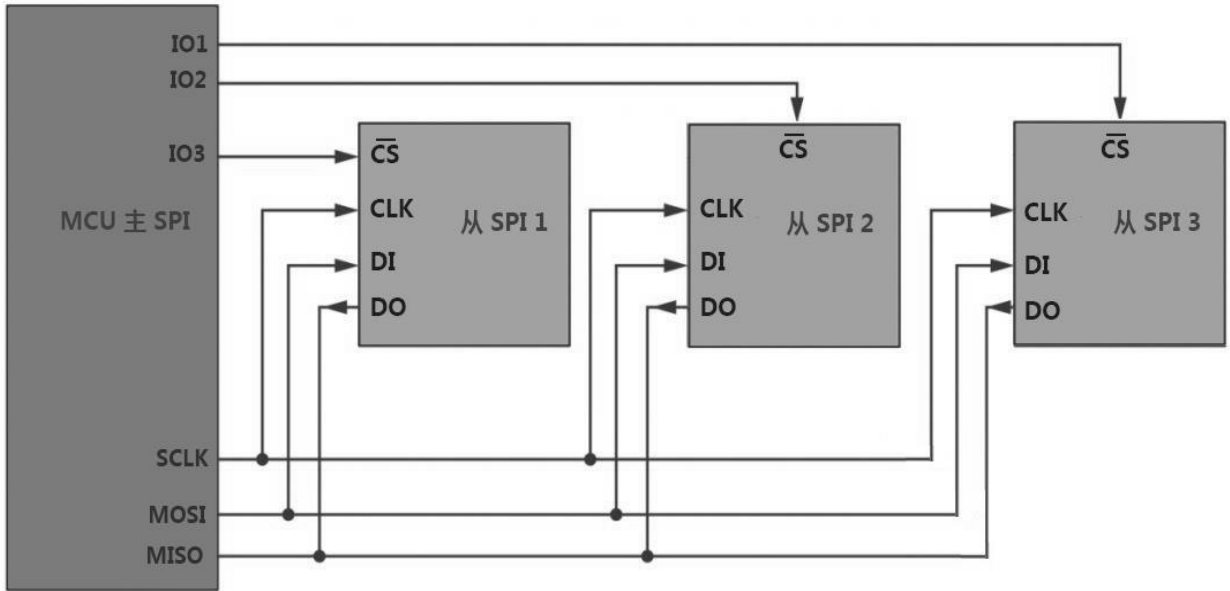


图 3.1.2 SPI 1 主多从定义

● 管脚描述:

管脚名称	功能	I/O类型
SSPCLK	SPI 串行时钟	I/O
SSPMOSI	主机输出从机输入	I/O
SSPMISO	主机输入从机输出	I/O
SSPFSS	帧，从机选择 (作为主机时) 帧输入 (作为从机时)	I/O

图 3.1.3 SPI 管脚定义

SSPCLK: 串行时钟 (主机输出)

SSPMOSI: 主输出从机输入或主机输出从机输入 (主机输出的数据)

SSPMISO: 主输入从输出或主输入从输出 (从输出的数据输出)

SSPNSS: 从机选择 (通常为低电平有效, 主机输出)

● 软件配置:

可在 apt32s003_initial.c 文件中 SPI_MASTER_CONFIG 进行 SPI 主机初始化的配置, SPI_SLAVE_CONFIG 进行 SPI 从机初始化。

```

/*****

```

```

//SPI MASTER Initial
//EntryParameter:NONE
//ReturnValue:NONE
/*****/
void SPI_MASTER_CONFIG(void)
{
    SPI_DeInit();
    SPI_NSS_IO_Init(1);
    SPI_Master_Init(SPI_G2,SPI_DATA_SIZE_8BIT,SPI_SPO_0,SPI_SPH_0,SPI_LBM_0,SPI_RXIFLSEL_1_8,0,10);
    //选择 SPI IO group1; 发送数据大小为 8BIT; SCK 工作时为低电平; SCK 第一个时钟沿捕捉; 串行正常输出; 接收占用 1/8 FIFO 中断触发断点;
    FSSPCLKOUT=20M/10=1M
    SPI_ConfigInterrupt_CMD(ENABLE,SPI_RXIM); //使能 FIFO 接收中断
    SPI_Int_Enable(); //使能 SPI 中断向量
    nop;
}
/*****/
//SPI SLAVE Initial
//EntryParameter:NONE
//ReturnValue:NONE
/*****/
void SPI_SLAVE_CONFIG(void)
{
    SPI_DeInit();
    SPI_NSS_IO_Init(1);
    SPI_Slave_Init(SPI_G1,SPI_DATA_SIZE_8BIT,SPI_SPH_0,SPI_RXIFLSEL_1_8,0,12);
    //选择 SPI IO group1; 发送数据大小为 8BIT;SCK 第一个时钟沿捕捉;接收占用 1/8 FIFO 中断触发断点 ;FSSPCLKOUT=20M/12=1.6M
    SPI_ConfigInterrupt_CMD(ENABLE,SPI_RXIM); //使能 FIFO 接收中断
    SPI_Int_Enable(); //使能 SPI 中断向量
}

```

3.2 SPI 主机配置

系统时钟为内部主频 48MHz，SPI 设置主机模式，进行测试数据发送。

- 管脚配置：

SCLK--PB0.2 / MOSI--PB0.3 / MISO--PA0.8 / NSS--PA0.6

```

/*****/
//SPI MASTER Initial
//EntryParameter:NONE
//ReturnValue:NONE
/*****/

```

```

void SPI_MASTER_CONFIG(void)
{
    SPI_DeInit();

    SPI_NSS_IO_Init(0);

    SPI_Master_Init(SPI_G0,SPI_DATA_SIZE_8BIT,SPI_SPO_0,SPI_SPH_0,SPI_LBM_0,SPI_RXIFLSEL_1_8,0,48);
    //选择 SPI IO group1; 发送数据大小为 8BIT; SCK 工作时为低电平; SCK 第一个时钟沿捕捉; 串行正常输出; 接收占用 1/8 FIFO 中断触发断点;
    FSSPCLKOUT=48M/48=1M

    SPI_ConfigInterrupt_CMD(ENABLE,SPI_RXIM); //使能 FIFO 接收中断
    SPI_Int_Enable(); //使能 SPI 中断向量
}

U8_T LOOP;

/*****/

//main
/*****/

int main(void)
{
    APT32S003_init();

    LOOP = 0;

    while(1)
    {
        SYSCON_IWDCNT_Reload();
        SPI_WRITE_BYTE (LOOP++); //发送数据累加
    }
}

```

● 代码说明:

SPI_DeInit(); ----用于恢复默认设置

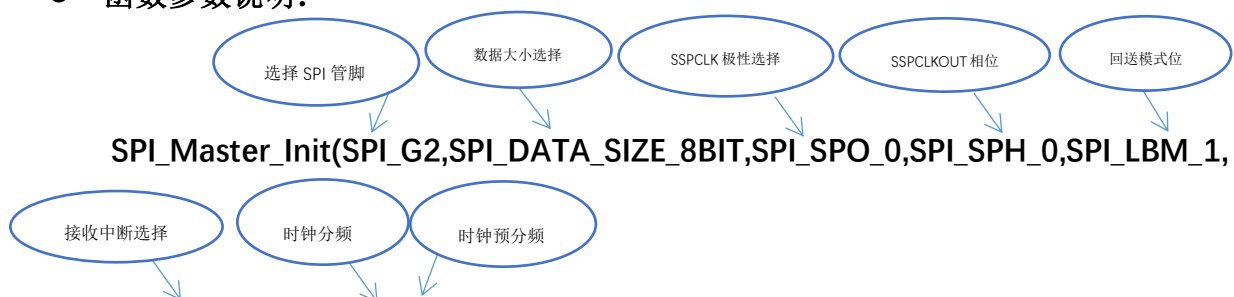
SPI_NSS_IO_Init(); ----用于配置 NSS 引脚

SPI_Master_Init(); ----用于配置主机模式

SPI_ConfigInterrupt_CMD(); ----用于配置 SPI 接收中断

SPI_Int_Enable(); ----用于使能中断

● 函数参数说明:



SPI_RXIFLSEL_1_8,0,10);



SPI_ConfigInterrupt_CMD(ENABLE,SPI_RXIM);

● 测试波形:

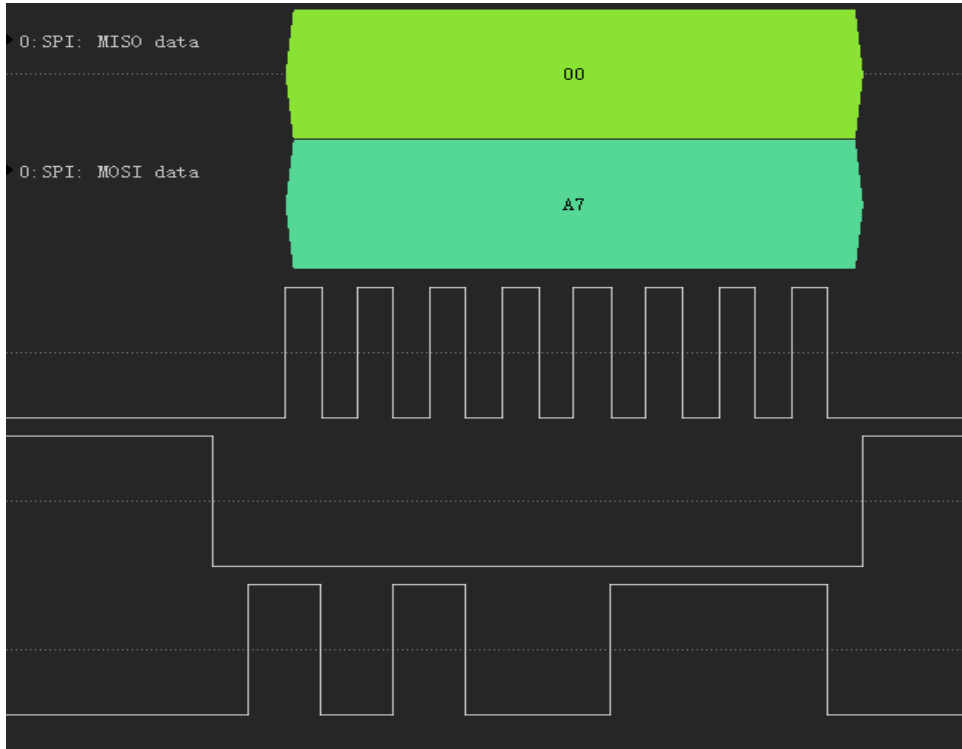


图 3.2.1 SPI 主发从收波形

3.3 SPI 从机配置

系统时钟选用内部主频 48MHZ，在中断中进行接收数据。

● 管脚配置:

SCLK--PB0.2 / MOSI--PB0.3 / MISO--PA0.8 / NSS--PA0.6

```
/*.....*/
//SPI MASTER Initial
//EntryParameter:NONE
//ReturnValue:NONE
/*.....*/
void SPI_MASTER_CONFIG(void)
{
```

```

SPL_DeInit();

SPL_NSS_IO_Init(0);

SPL_Master_Init(SPL_G0,SPL_DATA_SIZE_8BIT,SPL_SPO_0,SPL_SPH_0,SPL_LBM_1,SPL_RXIFLSEL_1_8,0,48);

//选择 SPI IO group1; 发送数据大小为 8BIT; SCK 工作时为低电平; SCK 第一个时钟沿捕捉; 串行正常输出; 接收占用 1/8 FIFO 中断触发断点;
FSSPCLKOUT=48M/48=1M

SPL_ConfigInterrupt_CMD(ENABLE,SPL_RXIM); //使能 FIFO 接收中断

SPL_Int_Enable(); //使能 SPI 中断向量
}

U8_T SPI_DATA_BUF;

/*****/

//SPI Interrupt

//EntryParameter:NONE

//ReturnValue:NONE

/*****/

void SPI0IntHandler(void)
{
// ISR content ...

if((SPI0->MISR&SPL_PORIM)==SPL_PORIM) //Receive Overrun Interrupt
{
SPI0->ICR = SPL_PORIM;
}

else if((SPI0->MISR&SPL_RTIM)==SPL_RTIM) //Receive Timeout Interrupt
{
SPI0->ICR = SPL_RTIM;
}

else if((SPI0->MISR&SPL_RXIM)==SPL_RXIM) //Receive FIFO Interrupt,FIFO can be set 1/8,1/4,1/2 FIFO Interrupt
{
SPI0->ICR = SPL_RXIM;

SPI_DATA_BUF=SPI0->DR;
}

else if((SPI0->MISR&SPL_TXIM)==SPL_TXIM) //Transmit FIFO Interrupt
{
SPI0->ICR = SPL_TXIM;
}

}

/*****/

//main

/*****/

int main(void)
{
APT32S003_init();

while(1)
{

```



```

        SYSCON_IWDCNT_Reload();
    }
}
    
```

● 代码说明:

- SPI_Delinit(); ----用于恢复默认设置
- SPI_NSS_IO_Init();----用于配置 NSS 引脚
- SPI_Slave_Init(); ----用于配置 SPI 从机
- SPI_ConfigInterrupt_CMD(); ----用于配置 SPI 中断
- SPI_Int_Enable(); ----用于使能中断

● 函数参数说明:



中断接收数据:

Expression	Value
SPI0	0x40090000
SPI0->DR	0x00000040

```

528 }
529 extern U8_T SPI_DATA_FLAG;
530 extern U8_T SPI_DATA_BUF;
531 /*****
532 //SPI Interrupt
533 //EntryParameter:NONE
534 //ReturnValue:NONE
535 *****/
536 void SPI0IntHandler(void)
537 {
538     // ISR content ...
539     if((SPI0->MISR&SPI_PORIM)==SPI_PORIM) //Receive Overrun Interrupt
540     {
541         SPI0->ICR = SPI_PORIM;
542     }
543     else if((SPI0->MISR&SPI_RTIM)==SPI_RTIM) //Receive Timeout Interrupt
544     {
545         SPI0->ICR = SPI_RTIM;
546     }
547     else if((SPI0->MISR&SPI_RXIM)==SPI_RXIM) //Receive FIFO Interrupt, FIFO
548     {
549         SPI0->ICR = SPI_RXIM;
550         SPI_DATA_BUF=SPI0->DR;
    
```

图 3.3.1 SPI 从机接收

4. 程序下载和运行

1. 将目标板与仿真器连接，分别为 VDD SCLK SWIO GND
2. 将 SPI 主机功能脚与对应的 SPI 从机设备功能脚进行连接
3. 程序编译后仿真运行
4. 进行读写数据，查看图 3.2.1 图 3.3.1 进行验证