

---

# SPI 教程

——疯壳·开发板系列

**Wolverine-Team**

2015/7/16

## 目录

第一节 SPI Flash 硬件电路.....	3
第二节 SPI+寄存器.....	4
2.1 SPI+介绍.....	4
2.2 寄存器介绍.....	4
2.2.1 SPI 控制寄存器 0.....	4
2.2.2 SPI 接收/发送寄存器 0.....	5
2.2.3 SPI 接收/发送寄存器 1.....	5
2.2.4 SPI 清除中断寄存器.....	5
2.2.5 SPI 控制寄存器 1.....	5
2.3 寄存器配置讲解.....	6
第三节 SPI 实验.....	7

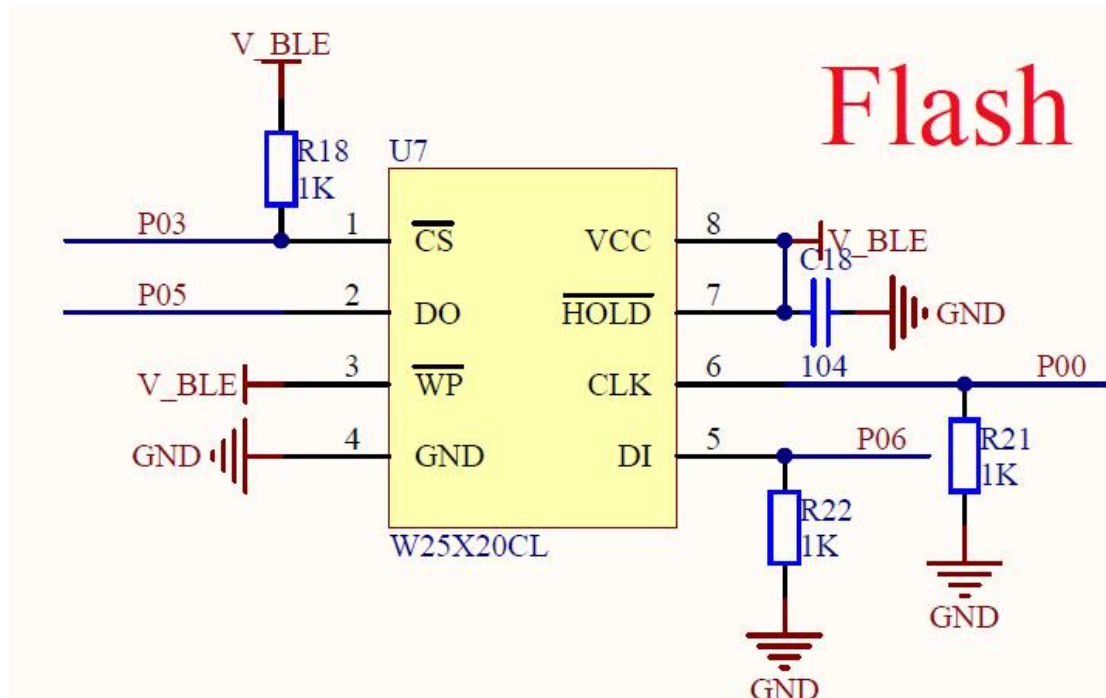
官网地址: <http://www.fengke.club>

购买链接: <http://shop115904315.taobao.com/>

官方 QQ 群: 193836402

## 第一节 SPI Flash 硬件电路

SPI\_Flash 可用于存储程序以及一些数据,如果需要存储程序则必须连接在规定的引脚,手环上选用的 MCU 引脚分别为: P00, P03, P05, P06, 如下图所示:



## 第二节 SPI+寄存器

### 2.1 SPI+介绍

这个接口支持 SPI 总线的一个子集。这个串行接口在主/从模式可以发送和接收 8、16 或 32 位，并且在主模式可以发送 9 位。SPI+接口有双向的 2×16 位字的 FIFO，功能得到了增强。

该接口可以工作在主或从模式；有 8、9、16、32 位的操作方式；SPI 控制器的时钟达到 16MHz，SPI 时钟源可以通过编程进行 1、2、4、8 分频；SPI 的时钟线达到 8MHz；支持 SPI 的 0、1、2、3 四种工作模式；SPI\_DO 的空闲电平可以通过编程设置；可屏蔽的中断发生器；单向读和写模式降低总线负载。

### 2.2 寄存器介绍

#### 2.2.1 SPI 控制寄存器 0

Table 161: SPI\_CTRL\_REG (0x50001200)

Bit	Mode	Symbol	Description	Reset
15	R/W	SPI_EN_CTRL	0 = SPI_EN pin disabled in slave mode. Pin SPI_EN is don't care. 1 = SPI_EN pin enabled in slave mode.	0x0
14	R/W	SPI_MINT	0 = Disable SPI_INT_BIT to ICU 1 = Enable SPI_INT_BIT to ICU. Note that the SPI_INT interrupt is shared with AD_INT interrupt	0x0
13	R	SPI_INT_BIT	0 = RX Register or FIFO is empty. 1 = SPI interrupt. Data has been transmitted and received-Must be reset by SW by writing to SPI_CLEAR_INT_REG.	0x0
12	R	SPI_DI	Returns the actual value of pin SPI_DIN (delayed with two internal SPI clock cycles)	0x0
11	R	SPI_TXH	0 = TX-FIFO is not full, data can be written. 1 = TX-FIFO is full, data can not be written.	0x0
10	R/W	SPI_FORCE_DO	0 = normal operation 1 = Force SPIDO output level to value of SPI_DO.	0x0
9	R/W	SPI_RST	0 = normal operation 1 = Reset SPI. Same function as SPI_ON except that internal clock remain active.	0x0
8:7	R/W	SPI_WORD	00 = 8 bits mode, only SPI_RX_TX_REG0 used 01 = 16 bit mode, only SPI_RX_TX_REG0 used 10 = 32 bits mode, SPI_RX_TX_REG0 & SPI_RX_TX_REG1 used 11 = 9 bits mode. Only valid in master mode.	0x0
6	R/W	SPI_SMN	Master/slave mode 0 = Master, 1 = Slave(SPI1 only)	0x0
5	R/W	SPI_DO	Pin SPI_DO output level when SPI is idle or when SPI_FORCE_DO=1	0x0
4:3	R/W	SPI_CLK	Select SPI_CLK clock frequency in master mode:00 = (XTAL) / (CLK_PER_REG *8) 01 = (XTAL) / (CLK_PER_REG *4) 10 = (XTAL) / (CLK_PER_REG *2) 11 = (XTAL) / (CLK_PER_REG *14)	0x0
2	R/W	SPI_POL	Select SPI_CLK polarity. 0 = SPI_CLK is initially low. 1 = SPI_CLK is initially high.	0x0
1	R/W	SPI_PHA	Select SPI_CLK phase. See functional timing diagrams in SPI chapter	0x0
0	R/W	SPI_ON	0 = SPI Module switched off (power saving). Everything is reset except SPI_CTRL_REG0 and SPI_CTRL_REG1. When this bit is cleared the SPI will remain active in master mode until the shift register and holding register are both empty. 1 = SPI Module switched on. Should only be set after all control bits have their desired values. So two writes are needed!	0x0

- 15 位：SPI\_EN 引脚是否有效，'0'表示无效，'1'表示有效；
- 14 位：SPI 中断使能位，'0'表示无效，'1'表示有效；
- 13 位：'0'表示接收寄存器或 FIFO 为空，'1'表示 SPI 中断，数据已经被发送或接收，该位必须通过软件写 SPI\_CLR\_INT\_REG 清零；
- 12 位：返回 SPI\_DI 引脚的状态值；
- 11 位：'0'表示发送 FIFO 满，'1'表示发送 FIFO 空；
- 10 位：'0'正常模式，'1'使 SPIDO 的值等于 SPI\_DO 的值；
- 9 位：'0'正常操作，'1'复位 SPI；
- 8:7 位：SPI 数据格式；
- 6 位：SPI 工作模式，'0'为主设备，'1'为从设备；
- 5 位：当 SPI 处于空闲模式或者当 SPI\_FORCE\_DO=1 时引脚 SPI\_DO 的值；
- 4:3 位：SPI 在主模式下的时钟频率选择；
- 2 位：SPI 时钟初始电平的高低，'0'为低，'1'为高；
- 1 位：SPI 时钟的相位选择，具体看 SPI 的时序图；(SPI 的资料)
- 0 位：SPI 的开关。

## 2.2.2 SPI 接收/发送寄存器 0

Table 162: SPI\_RX\_TX\_REG0 (0x50001202)

Bit	Mode	Symbol	Description	Reset
15:0	R0/W	SPI_DATA0	Write: SPI_TX_REG0 output register 0 (TX-FIFO) Read: SPI_RX_REG0 input register 0 (RX-FIFO) In 8 or 9 bits mode bits 15 to 8 are not used, they contain old data.	0x0

15:0 位：SPI 发送或接收的数据，写操作是，存储发送的数据，读操作时，为接收到的数据。

## 2.2.3 SPI 接收/发送寄存器 1

Table 163: SPI\_RX\_TX\_REG1 (0x50001204)

Bit	Mode	Symbol	Description	Reset
15:0	R0/W	SPI_DATA1	Write: SPI_TX_REG1 output register 1 (MSB's of TX-FIFO) Read: SPI_RX_REG1 input register 1 (MSB's of RX-FIFO) In 8 or 9 or 16 bits mode bits this register is not used.	0x0

15:0 位：SPI 发送或接收的数据，写操作是，存储发送的数据，读操作时，为接收到的数据；为 32 位模式的高 16 位数据。

## 2.2.4 SPI 清除中断寄存器

Table 164: SPI\_CLEAR\_INT\_REG (0x50001206)

Bit	Mode	Symbol	Description	Reset
15:0	R0/W	SPI_CLEAR_INT	Writing any value to this register will clear the SPI_CTRL_REG[SPI_INT_BIT] Reading returns 0.	0x0

15:0 位：写任意值到该寄存器清除 SPI 的中断标志。

## 2.2.5 SPI 控制寄存器 1

Table 165: SPI\_CTRL\_REG1 (0x50001208)

Bit	Mode	Symbol	Description	Reset
15:5	-	-	Reserved	0x0
4	R/W	SPI_9BIT_VAL	Determines the value of the first bit in 9 bits SPI mode.	0x0



3	R	SPI_BUSY	0 = The SPI is not busy with a transfer. This means that either no TX-data is available or that the transfers have been suspended due to a full RX-FIFO. The SPIx_CTRL_REG0[SPI_INT_BIT] can be used to distinguish between these situations. 1 = The SPI is busy with a transfer.	0x0
2	R/W	SPI_PRIORITY	0 = The SPI has low priority, the DMA request signals are reset after the corresponding acknowledge. 1 = The SPI has high priority, DMA request signals remain active until the FIFOs are filled/emptied, so the DMA holds the AHB bus.	0x0
1:0	R/W	SPI_FIFO_MODE	0: TX-FIFO and RX-FIFO used (Bidirectional mode). 1: RX-FIFO used (Read Only Mode) TX-FIFO single depth, no flow control 2: TX-FIFO used (Write Only Mode), RX-FIFO single depth, no flow control 3: No FIFOs used (backwards compatible mode)	0x3

15:5 位：保留不使用；

4 位：决定在 9 位模式下的第一位的值；

3 位：SPI 忙标志位，‘0’表示 SPI 空闲，‘1’表示 SPI 忙；

2 位：SPI 优先级选择位，‘0’低优先级，‘1’高优先级；

1:0 位：SPI\_FIFO 模式。

## 2.3 寄存器配置讲解

```
#define CLK_PER_REG          (* ( volatile uint16*)0x50000004)
#define SPI_CTRL_REG        (* ( volatile uint16*)0x50001200)
#define SPI_RX_TX_REG0      (* ( volatile uint16*)0x50001202)
#define SPI_RX_TX_REG1      (* ( volatile uint16*)0x50001204)
#define SPI_CLEAR_INT_REG   (* ( volatile uint16*)0x50001206)
#define SPI_CTRL_REG1       (* ( volatile uint16*)0x50001208)
```

启动 SPI 模块的时钟：CLK\_PER\_REG |= 0x0800;

SPI 的初始化配置寄存器：

先关闭 SPI， SPI\_CTRL\_REG = 0x8000;

SPI 配置为 8 位模式，主模式，时钟空闲为低电平，相位模式 0，关闭 SPI 中断，时钟 8 分频 (0x1000000000000000)，则 SPI\_CTRL\_REG = 0x8000;

开 SPI， SPI\_CTRL\_REG |= 0x0001;

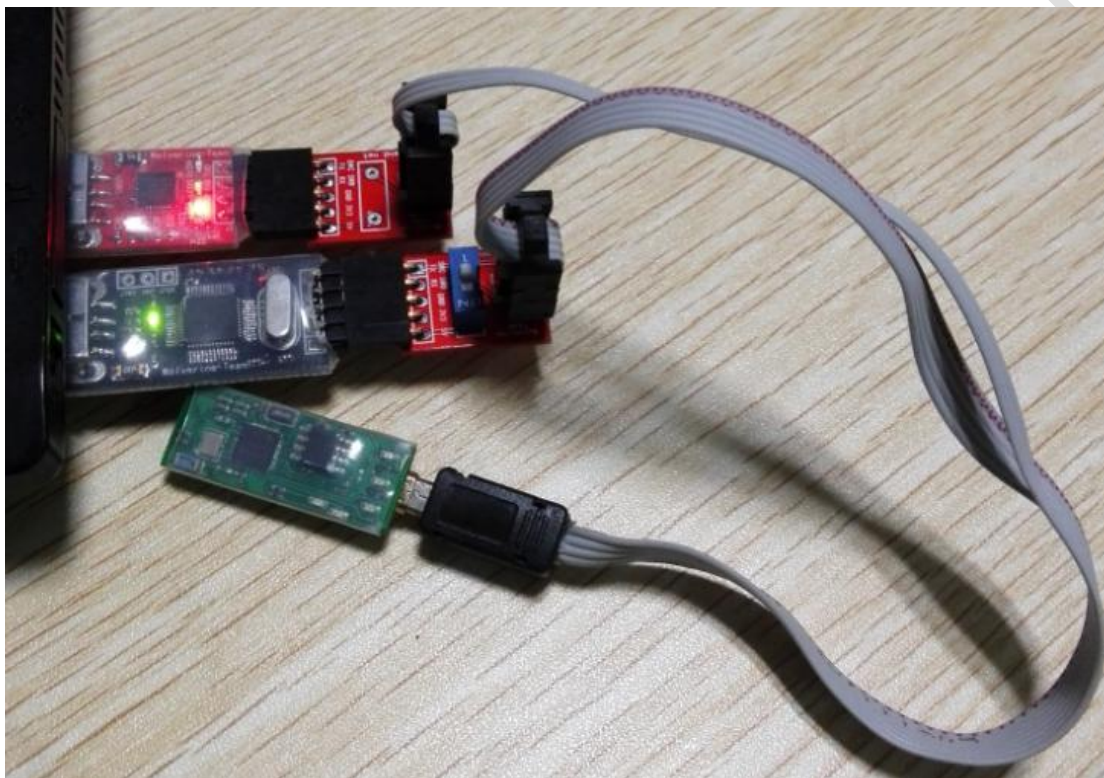
发送一个字节 0x55，将数据填充进发送寄存器 SPI\_RX\_TX\_REG0 = 0x55；等待发送寄存器为空 while(SPI\_CTRL\_REG & 0x2000);

接收一个字节，读取接收寄存器 rx\_data = SPI\_RX\_TX\_REG1;

## 第三节 SPI 实验

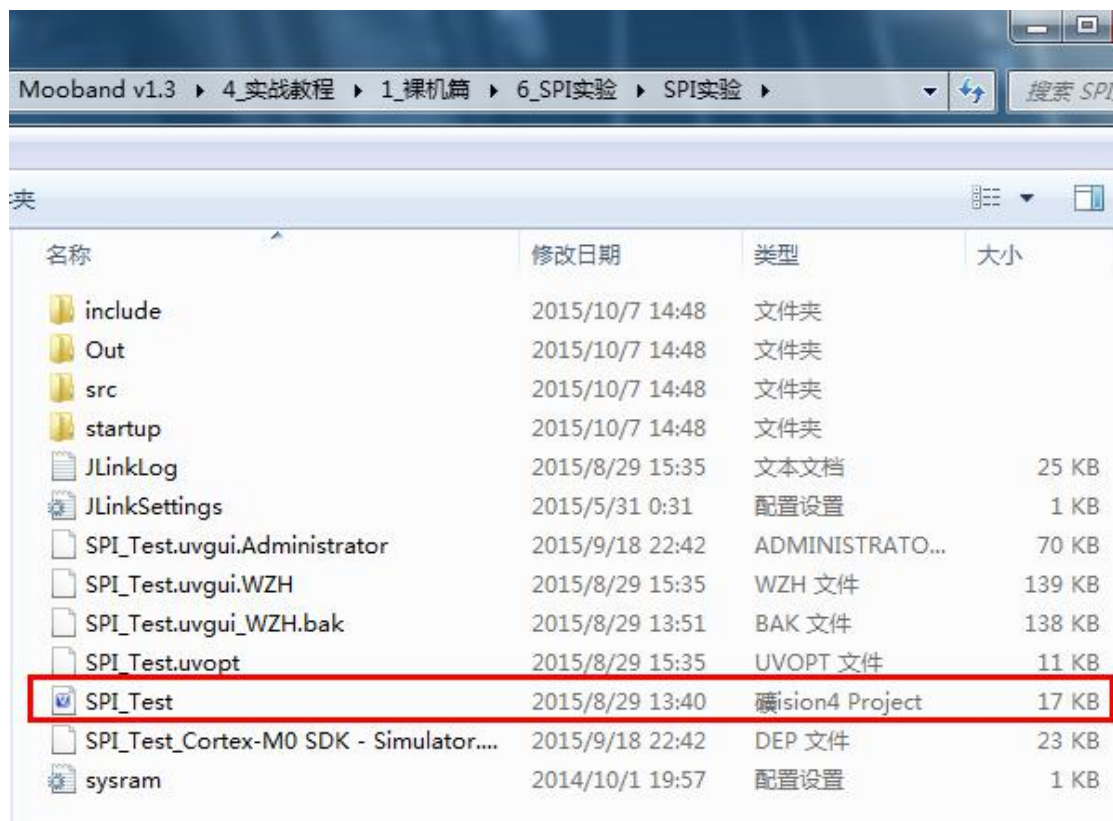
实验需要使用的模块有：手环，Jlink 调试工具，USB 转串模块，一根手环下载调试线。

将 JLINK 通过下载调试线连接到手环的 USB 调试接口，JLINK 插在有拨码开关的一端，注意丝印标注一一一对应，将 JLINK 插上电脑的 USB 口。将 USB 转串模块插在手环现在调试线的另一端，注意丝印标注一一一对应，然后将 USB 转串模块插在电脑的 USB 接口。如下图所示：



SPI 实验的 Keil 工程为 SPI\_Test.uvproj，位于目录：..\Mooband v1.3\4\_实战教程\1\_裸机篇\6\_SPI 实验\SPI 实验，如下图所示：





打开串口调试助手连接串口模块。在 KEIL 中编译源代码，点击 **DEBUG**，然后点击全速运行，就看到串口打印出的读写 Flash 的信息，如下图所示：



```

*****
*****      金刚狼团队      *****
*****      Mooband v1.3      *****
*****      官网:www.mooband.net      *****
*****      淘宝:shop115904315.taobao.com      *****
*****      官方Q Q群: 193836402      *****
*****
*****      SPI测试      *****
*****

```

```

SPI flash JEDEC ID is EF3012

You are using W25X20 (2-MBit) SPI flash device.

```

```

SPI flash Manufacturer/Device ID is EF11

```

```

is writing 256 Bytes...

```

```

Finish writing!

```

```

Read the 256 Bytes!...

```

```

00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20 2
1 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F 30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F 40 41 42
43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F 50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F 60 61 62 63
64 65 66 67 68 69 6A 6B 6C 6D 6E 6F 70 71 72 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F 80 81 82 83 84 8
5 86 87 88 89 8A 8B 8C 8D 8E 8F 90 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F A0 A1 A2 A3 A4 A5 A6
A7 A8 A9 AA AB AC AD AE AF B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB BC BD BE BF C0 C1 C2 C3 C4 C5 C6 C7
C8 C9 CA CB CC CD CE CF D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 DA DB DC DD DE DF E0 E1 E2 E3 E4 E5 E6 E7 E8 E
9 EA EB EC ED EE EF F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB FC FD FE FF

```

```

Finish reading the 256 Bytes, are they 00~ff?

```