零死角玩转STM32

SPI—读写串行FLASH

淘宝：fire-stm32.taobao.com

论坛： www.firebbs.cn

扫描进入淘宝店铺

主讲内容

01 SPI协议简介

02 STM32的SPI特性及架构

03 SPI初始化结构体详解

04 SPI—读写串行FLASH实验

参考资料:《零死角玩转STM32》

“SPI—读写串行FLASH”章节

SPI—读写串行FLASH

SPI初始化结构体详解

跟其它外设一样，STM32标准库提供了SPI初始化结构体及初始化函数来配置SPI外设。初始化结构体及函数定义在库文件“stm32f4xx_spi.h”及“stm32f4xx_spi.c”中，编程时我们可以结合这两个文件内的注释使用或参考库帮助文档。

```
1  typedef struct
 2  {
 3     uint16_t SPI_Direction;           /*设置 SPI 的单双向模式 */
 4     uint16_t SPI_Mode;               /*设置 SPI 的主/从机端模式 */
 5     uint16_t SPI_DataSize;            /*设置 SPI 的数据帧长度，可选 8/16 位 */
 6     uint16_t SPI_CPOL;               /*设置时钟极性 CPOL，可选高/低电平*/
 7     uint16_t SPI_CPHA;               /*设置时钟相位，可选奇/偶数边沿采样 */
 8     uint16_t SPI_NSS;               /*设置 NSS 引脚由 SPI 硬件控制还是软件控制*/
 9     uint16_t SPI_BaudRatePrescaler;   /*设置时钟分频因子，fpclk/分频数=fSCK */
10     uint16_t SPI_FirstBit;           /*设置 MSB/LSB 先行 */
11     uint16_t SPI_CRCPolynomial;       /*设置 CRC 校验的表达式 */
12  } SPI_InitTypeDef;
```

SPI—读写串行FLASH

```
1 typedef struct
 2 {
 3     uint16_t SPI_Direction;          /*设置 SPI 的单双向模式 */
 4     uint16_t SPI_Mode;               /*设置 SPI 的主/从机端模式 */
 5     uint16_t SPI_DataSize;           /*设置 SPI 的数据帧长度，可选 8/16 位 */
 6     uint16_t SPI_CPOL;               /*设置时钟极性 CPOL，可选高/低电平*/
 7     uint16_t SPI_CPHA;               /*设置时钟相位，可选奇/偶数边沿采样 */
 8     uint16_t SPI_NSS;                /*设置 NSS 引脚由 SPI 硬件控制还是软件控制*/
 9     uint16_t SPI_BaudRatePrescaler;  /*设置时钟分频因子，fpclk/分频数=fSCK */
10     uint16_t SPI_FirstBit;           /*设置 MSB/LSB 先行 */
11     uint16_t SPI_CRCPolynomial;      /*设置 CRC 校验的表达式 */
12 } SPI_InitTypeDef;
```

- SPI_Direction

本成员设置SPI的通讯方向，可设置为双线全双工(SPI_Direction_2Lines_FullDuplex)，双线只接收(SPI_Direction_2Lines_RxOnly)，单线只接收(SPI_Direction_1Line_Rx)、单线只发送模式(SPI_Direction_1Line_Tx)。

SPI—读写串行FLASH

```
1  typedef struct
2  {
3      uint16_t SPI_Direction;           /*设置 SPI 的单双向模式 */
4      uint16_t SPI_Mode;               /*设置 SPI 的主/从机端模式 */
5      uint16_t SPI_DataSize;            /*设置 SPI 的数据帧长度，可选 8/16 位 */
6      uint16_t SPI_CPOL;               /*设置时钟极性 CPOL，可选高/低电平*/
7      uint16_t SPI_CPHA;               /*设置时钟相位，可选奇/偶数边沿采样 */
8      uint16_t SPI_NSS;               /*设置 NSS 引脚由 SPI 硬件控制还是软件控制*/
9      uint16_t SPI_BaudRatePrescaler;   /*设置时钟分频因子，fpclk/分频数=fSCK */
10     uint16_t SPI_FirstBit;            /*设置 MSB/LSB 先行 */
11     uint16_t SPI_CRCPolynomial;       /*设置 CRC 校验的表达式 */
12 } SPI_InitTypeDef;
```

- SPI_Mode

本成员设置SPI工作在主机模式(SPI_Mode_Master)或从机模式(SPI_Mode_Slave)，这两个模式的最大区别为SPI的SCK信号线的时序，SCK的时序是由通讯中的主机产生的。若被配置为从机模式，STM32的SPI外设将接受外来的SCK信号。

SPI—读写串行FLASH

```
1  typedef struct
2  {
3      uint16_t SPI_Direction;           /*设置 SPI 的单双向模式 */
4      uint16_t SPI_Mode;               /*设置 SPI 的主/从机端模式 */
5      uint16_t SPI_DataSize;            /*设置 SPI 的数据帧长度，可选 8/16 位 */
6      uint16_t SPI_CPOL;               /*设置时钟极性 CPOL，可选高/低电平*/
7      uint16_t SPI_CPHA;               /*设置时钟相位，可选奇/偶数边沿采样 */
8      uint16_t SPI_NSS;               /*设置 NSS 引脚由 SPI 硬件控制还是软件控制*/
9      uint16_t SPI_BaudRatePrescaler;   /*设置时钟分频因子，fpclk/分频数=fSCK */
10     uint16_t SPI_FirstBit;            /*设置 MSB/LSB 先行 */
11     uint16_t SPI_CRCPolynomial;       /*设置 CRC 校验的表达式 */
12 } SPI_InitTypeDef;
```

- SPI_DataSize

本成员可以选择SPI通讯的数据帧大小是为8位(SPI_DataSize_8b)还是16位(SPI_DataSize_16b)。

SPI—读写串行FLASH

```
1 typedef struct
 2 {
 3     uint16_t SPI_Direction;           /*设置 SPI 的单双向模式 */
 4     uint16_t SPI_Mode;               /*设置 SPI 的主/从机端模式 */
 5     uint16_t SPI_DataSize;            /*设置 SPI 的数据帧长度，可选 8/16 位 */
 6     uint16_t SPI_CPOL;               /*设置时钟极性 CPOL，可选高/低电平*/
 7     uint16_t SPI_CPHA;               /*设置时钟相位，可选奇/偶数边沿采样 */
 8     uint16_t SPI_NSS;               /*设置 NSS 引脚由 SPI 硬件控制还是软件控制*/
 9     uint16_t SPI_BaudRatePrescaler;  /*设置时钟分频因子，fpclk/分频数=fSCK */
10     uint16_t SPI_FirstBit;           /*设置 MSB/LSB 先行 */
11     uint16_t SPI_CRCPolynomial;       /*设置 CRC 校验的表达式 */
12 } SPI_InitTypeDef;
```

- SPI_CPOL和SPI_CPHA

这两个成员配置SPI的时钟极性CPOL和时钟相位CPHA，这两个配置影响到SPI的通讯模式，

时钟极性CPOL成员，可设置为高电平(SPI_CPOL_High)或低电平(SPI_CPOL_Low)。

时钟相位CPHA 则可以设置为SPI_CPHA_1Edge(在SCK的奇数边沿采集数据) 或SPI_CPHA_2Edge (在SCK的偶数边沿采集数据) 。

SPI—读写串行FLASH

```
1  typedef struct
2  {
3      uint16_t SPI_Direction;           /*设置 SPI 的单双向模式 */
4      uint16_t SPI_Mode;               /*设置 SPI 的主/从机端模式 */
5      uint16_t SPI_DataSize;            /*设置 SPI 的数据帧长度，可选 8/16 位 */
6      uint16_t SPI_CPOL;               /*设置时钟极性 CPOL，可选高/低电平*/
7      uint16_t SPI_CPHA;               /*设置时钟相位，可选奇/偶数边沿采样 */
8      uint16_t SPI_NSS;               /*设置 NSS 引脚由 SPI 硬件控制还是软件控制*/
9      uint16_t SPI_BaudRatePrescaler;   /*设置时钟分频因子，fpclk/分频数=fSCK */
10     uint16_t SPI_FirstBit;            /*设置 MSB/LSB 先行 */
11     uint16_t SPI_CRCPolynomial;       /*设置 CRC 校验的表达式 */
12 } SPI_InitTypeDef;
```

- SPI_NSS

本成员配置NSS引脚的使用模式，可以选择为硬件模式(SPI_NSS_Hard)与软件模式(SPI_NSS_Soft)，在硬件模式中的SPI片选信号由SPI硬件自动产生，而软件模式则需要亲自把相应的GPIO端口拉高或置低产生非片选和片选信号。

实际中软件模式应用比较多。

SPI—读写串行FLASH

```
1  typedef struct
2  {
3      uint16_t SPI_Direction;           /*设置 SPI 的单双向模式 */
4      uint16_t SPI_Mode;               /*设置 SPI 的主/从机端模式 */
5      uint16_t SPI_DataSize;            /*设置 SPI 的数据帧长度，可选 8/16 位 */
6      uint16_t SPI_CPOL;               /*设置时钟极性 CPOL，可选高/低电平*/
7      uint16_t SPI_CPHA;               /*设置时钟相位，可选奇/偶数边沿采样 */
8      uint16_t SPI_NSS;               /*设置 NSS 引脚由 SPI 硬件控制还是软件控制*/
9      uint16_t SPI_BaudRatePrescaler;   /*设置时钟分频因子，fpclk/分频数=fSCK */
10     uint16_t SPI_FirstBit;            /*设置 MSB/LSB 先行 */
11     uint16_t SPI_CRCPolynomial;       /*设置 CRC 校验的表达式 */
12 } SPI_InitTypeDef;
```

- SPI_BaudRatePrescaler

本成员设置波特率分频因子，分频后的时钟即为SPI的SCK信号线的时钟频率。这个成员参数可设置为fpclk的2、4、6、8、16、32、64、128、256分频。

SPI—读写串行FLASH

```
1  typedef struct
2  {
3      uint16_t SPI_Direction;           /*设置 SPI 的单双向模式 */
4      uint16_t SPI_Mode;               /*设置 SPI 的主/从机端模式 */
5      uint16_t SPI_DataSize;            /*设置 SPI 的数据帧长度，可选 8/16 位 */
6      uint16_t SPI_CPOL;               /*设置时钟极性 CPOL，可选高/低电平*/
7      uint16_t SPI_CPHA;               /*设置时钟相位，可选奇/偶数边沿采样 */
8      uint16_t SPI_NSS;               /*设置 NSS 引脚由 SPI 硬件控制还是软件控制*/
9      uint16_t SPI_BaudRatePrescaler;   /*设置时钟分频因子，fpclk/分频数=fSCK */
10     uint16_t SPI_FirstBit;            /*设置 MSB/LSB 先行 */
11     uint16_t SPI_CRCPolynomial;       /*设置 CRC 校验的表达式 */
12 } SPI_InitTypeDef;
```

- SPI_FirstBit

所有串行的通讯协议都会有MSB先行(高位数据在前)还是LSB先行(低位数据在前)的问题，而STM32的SPI模块可以通过这个结构体成员，对该特性编程控制。

SPI—读写串行FLASH

```
1  typedef struct
2  {
3      uint16_t SPI_Direction;           /*设置 SPI 的单双向模式 */
4      uint16_t SPI_Mode;               /*设置 SPI 的主/从机端模式 */
5      uint16_t SPI_DataSize;            /*设置 SPI 的数据帧长度，可选 8/16 位 */
6      uint16_t SPI_CPOL;               /*设置时钟极性 CPOL，可选高/低电平*/
7      uint16_t SPI_CPHA;               /*设置时钟相位，可选奇/偶数边沿采样 */
8      uint16_t SPI_NSS;               /*设置 NSS 引脚由 SPI 硬件控制还是软件控制*/
9      uint16_t SPI_BaudRatePrescaler;   /*设置时钟分频因子，fpclk/分频数=fSCK */
10     uint16_t SPI_FirstBit;            /*设置 MSB/LSB 先行 */
11     uint16_t SPI_CRCPolynomial;       /*设置 CRC 校验的表达式 */
12 } SPI_InitTypeDef;
```

- SPI_CRCPolynomial

这是SPI的CRC校验中的多项式，若我们使用CRC校验时，就使用这个成员的参数(多项式)，来计算CRC的值。

配置完这些结构体成员后，要调用SPI_Init函数把这些参数写入到寄存器中，实现SPI的初始化，然后调用SPI_Cmd来使能SPI外设。

THANKS

论坛： www.firebbs.cn

淘宝：fire-stm32.taobao.com

扫描进入淘宝店铺