

零死角玩转STM32



SPI—读写串行 FLASH

淘宝: firestm32.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

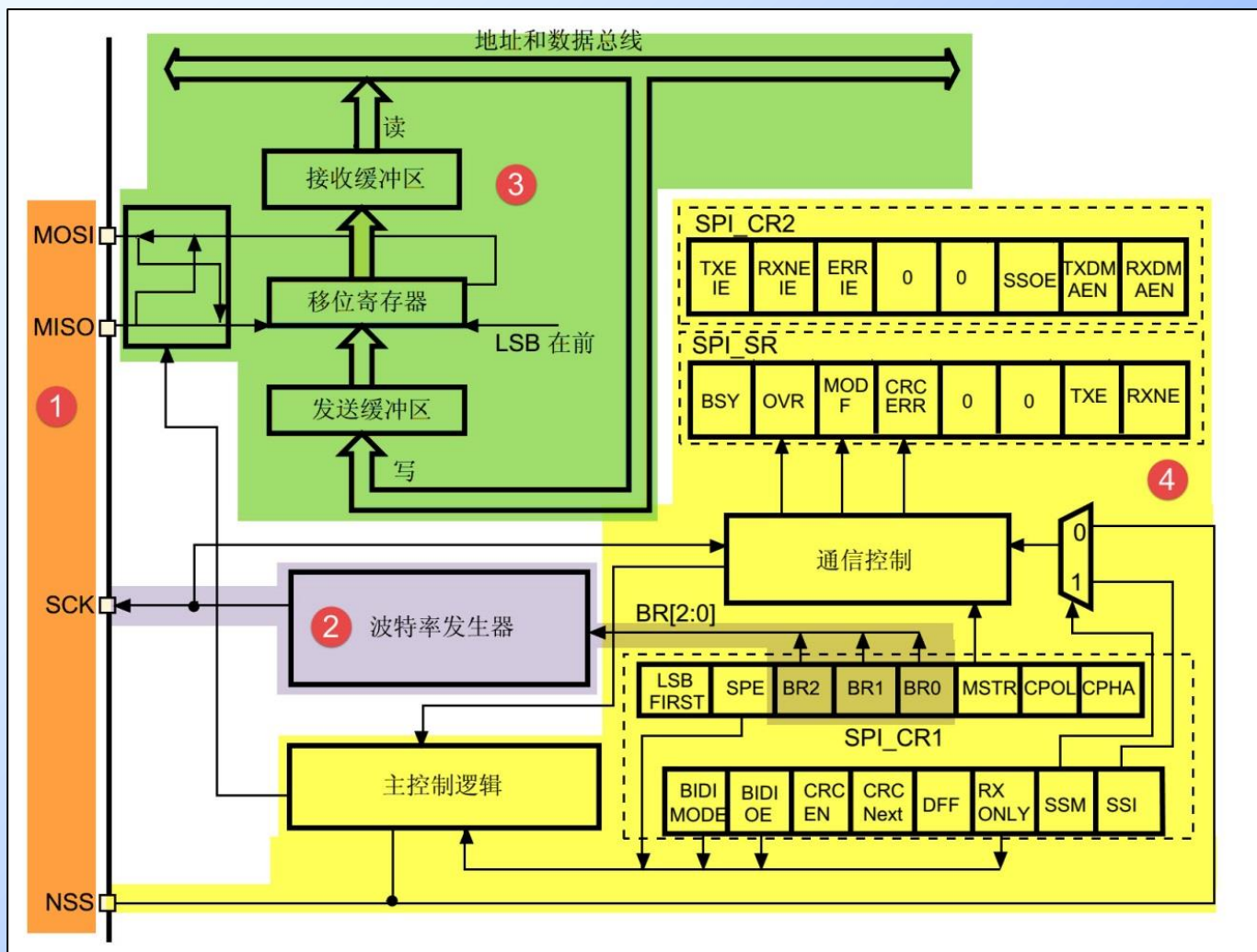


STM32的SPI外设简介

STM32的SPI外设可用作通讯的主机及从机，支持最高的SCK时钟频率为 $f_{\text{pclk}}/2$ (STM32F10x型号的芯片默认 f_{pclk1} 为36MHz， f_{pclk2} 为72MHz)，完全支持SPI协议的4种模式，数据帧长度可设置为8位或16位，可设置数据MSB先行或LSB先行。它还支持双线全双工(前面小节说明的都是这种模式)、双线单向以及单线模式。

SPI—读写串行FLASH

STM32的SPI架构剖析



- 通讯引脚
- 时钟控制逻辑
- 数据控制逻辑
- 整体控制逻辑

SPI—读写串行FLASH



1.通讯引脚

STM32芯片有多个SPI外设，它们的SPI通讯信号引出到不同的GPIO引脚上，使用时必须配置到这些指定的引脚，以《STM32F10x规格书》为准。

引脚	SPI编号		
	SPI1	SPI2	SPI3
NSS	PA4	PB12	PA15下载口的TDI
CLK	PA5	PB13	PB3下载口的TDO
MISO	PA6	PB14	PB4下载口的NTRST
MOSI	PA7	PB15	PB5

其中SPI1是APB2上的设备，最高通信速率达36Mbtis/s，SPI2、SPI3是APB1上的设备，最高通信速率为18Mbits/s。除了通讯速率，在其它功能上没有差异。

SPI—读写串行FLASH



2.时钟控制逻辑

SCK线的时钟信号，由波特率发生器根据“控制寄存器CR1”中的BR[0:2]位控制，该位是对 f_{pclk} 时钟的分频因子，对 f_{pclk} 的分频结果就是SCK引脚的输出时钟频率

BR[0:2]	分频结果(SCK频率)		BR[0:2]	分频结果(SCK频率)
000	$f_{\text{pclk}}/2$		100	$f_{\text{pclk}}/32$
001	$f_{\text{pclk}}/4$		101	$f_{\text{pclk}}/64$
010	$f_{\text{pclk}}/8$		110	$f_{\text{pclk}}/128$
011	$f_{\text{pclk}}/16$		111	$f_{\text{pclk}}/256$

其中的 f_{pclk} 频率是指SPI所在的APB总线频率，APB1为 f_{pclk1} ，APB2为 f_{pclk2} 。

SPI—读写串行FLASH



3.数据控制逻辑

SPI的MOSI及MISO都连接到数据移位寄存器上，数据移位寄存器的数据来源来源于接收缓冲区及发送缓冲区。

- 通过写SPI的“数据寄存器DR”把数据填充到发送缓冲区中。
- 通过读“数据寄存器DR”，可以获取接收缓冲区中的内容。
- 其中数据帧长度可以通过“控制寄存器CR1”的“DFF位”配置成8位及16位模式；配置“LSBFIRST位”可选择MSB先行还是LSB先行。

SPI—读写串行FLASH



4.整体控制逻辑

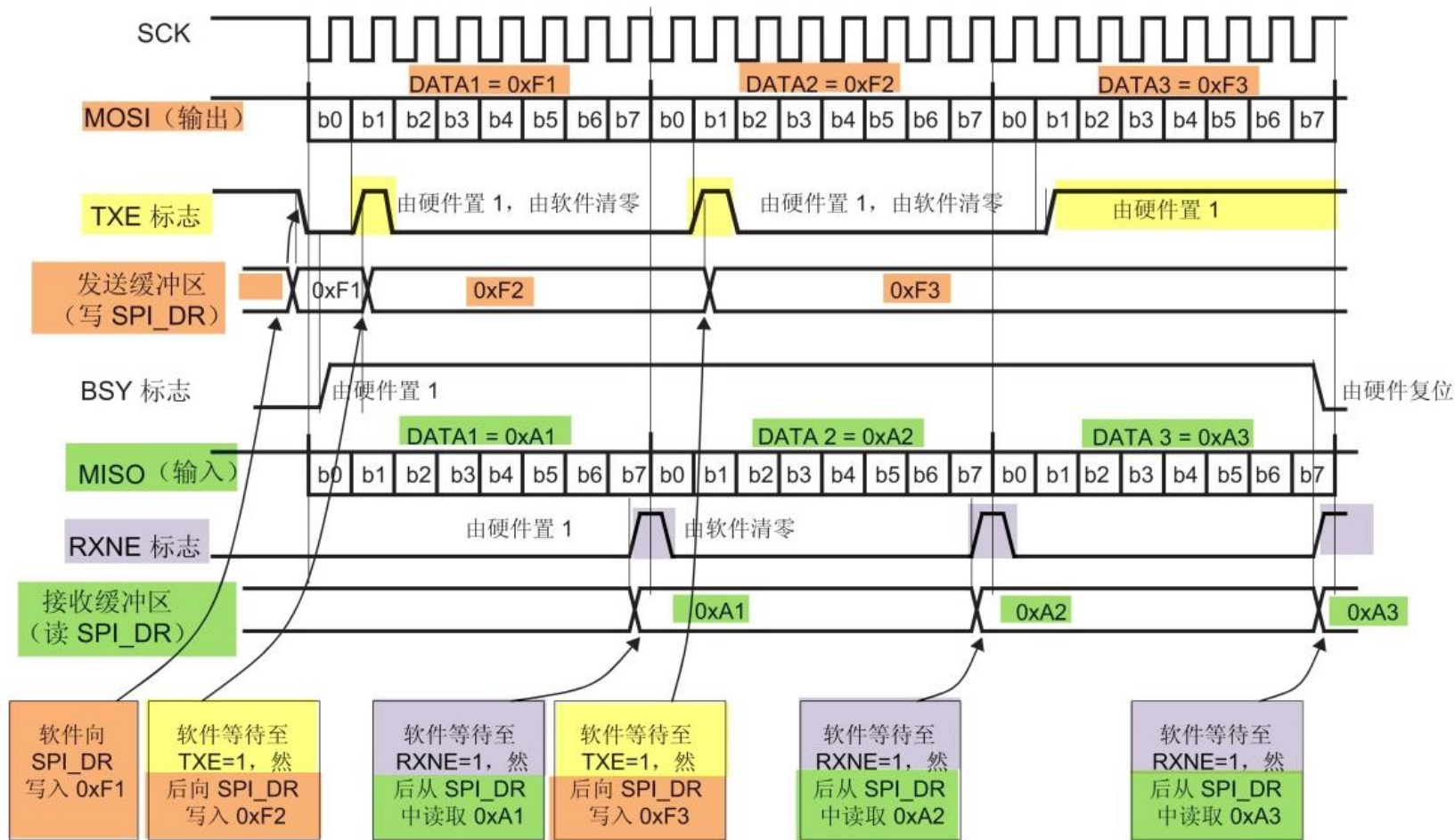
- 整体控制逻辑负责协调整个**SPI**外设，控制逻辑的工作模式根据“控制寄存器(CR1/CR2)”的参数而改变，基本的控制参数包括前面提到的**SPI**模式、波特率、**LSB**先行、主从模式、单双向模式等等。
- 在外设工作时，控制逻辑会根据外设的工作状态修改“状态寄存器(**SR**)”，只要读取状态寄存器相关的寄存器位，就可以了解**SPI**的工作状态了。除此之外，控制逻辑还根据要求，负责控制产生**SPI**中断信号、**DMA**请求及控制**NSS**信号线。
- 实际应用中，一般不使用**STM32 SPI**外设的标准**NSS**信号线，而是更简单地使用普通的**GPIO**，软件控制它的电平输出，从而产生通讯起始和停止信号。

SPI—读写串行FLASH



通讯过程

CPOL=1、CPHA=1 时的主模式示例



SPI—读写串行FLASH



通讯过程

- 控制**NSS**信号线，产生起始信号(图中没有画出)；
- 把要发送的数据写入到“数据寄存器**DR**”中，该数据会被存储到发送缓冲区；
- 通讯开始，**SCK**时钟开始运行。**MOSI**把发送缓冲区中的数据一位一位地传出去；**MISO**则把数据一位一位地存储进接收缓冲区中；
- 当发送完一帧数据的时候，“状态寄存器**SR**”中的“**TXE**标志位”会被置**1**，表示传输完一帧，发送缓冲区已空；类似地，当接收完一帧数据的时候，“**RXNE**标志位”会被置**1**，表示传输完一帧，接收缓冲区非空；
- 等待到“**TXE**标志位”为**1**时，若还要继续发送数据，则再次往“数据寄存器**DR**”写入数据即可；等待到“**RXNE**标志位”为**1**时，通过读取“数据寄存器**DR**”可以获取接收缓冲区中的内容。

假如使能了**TXE**或**RXNE**中断，**TXE**或**RXNE**置**1**时会产生**SPI**中断信号，进入同一个中断服务函数，到**SPI**中断服务程序后，可通过检查寄存器位来了解是哪一个事件，再分别进行处理。也可以使用**DMA**方式来收发“数据寄存器**DR**”中的数据。

零死角玩转STM32



THANKS

论坛: www.firebbs.cn

淘宝: firestm32.taobao.com



扫描进入淘宝店铺