嵌入式系统原理及实验

顾　震

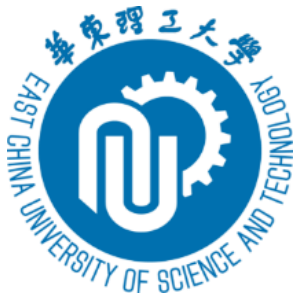
信息科学与工程学院自动化系

华东理工大学

Email: guzhen@ecust.edu.cn

课程大纲

1. 嵌入式系统导论
2. Cortex-M3微处理器
3. STM32最小系统及开发环境
4. 嵌入式C语言
5. 通用输入输出GPIO模块
6. 中断
7. 定时器原理与应用
8. USART通信原理及实现
9. DMA控制器
10. SPI与I2C通信原理及实现
11. 模数转换原理及实现
12. 人工智能辅助的嵌入式项目开发
13. 嵌入式应用前沿



10. SPI与I2C通信原理及实现

本章知识与能力要求

- 理解和掌握SPI、I2C通信协议基本原理；
- 理解通信协议的时序分析方法；
- 熟悉HAL库中有关SPI、I2C通信的库函数及软件模拟；

10. SPI通信协议

10.1 SPI协议基本概念

- SPI 是Serial Peripheral interface的缩写，即**串行外围设备接口**;
- 是Motorola(摩托罗拉)首先在其MC68HCXX系列处理器上定义的一种通信协议。

特点：

- ✓ 高速的，全双工，同步的通信总线
- ✓ 在芯片的管脚上只占用四根线
- ✓ 主要应用: EEPROM、FLASH、实时时钟、AD/DA转换器、数字信号处理器和数字信号解码器等。

10.1 SPI协议基本概念

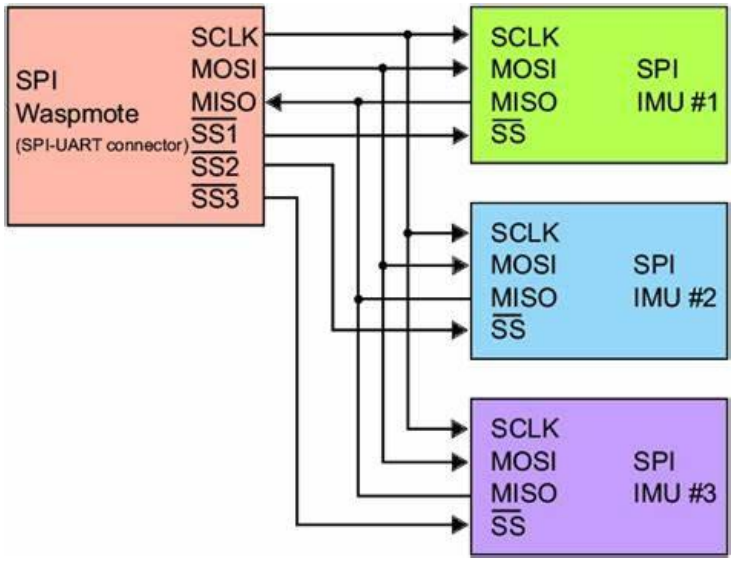
- SPI 是Serial Peripheral interface的缩写，即**串行外围设备接口**;
- 是Motorola(摩托罗拉)首先在其MC68HCXX系列处理器上定义的一种通信协议。

特点：

- ✓ 高速的，全双工，同步的通信总线
- ✓ 在芯片的管脚上只占用四根线
- ✓ 主要应用: EEPROM、FLASH、实时时钟、AD/DA转换器、数字信号处理器和数字信号解码器等。

10.1.1 SPI主从模式

- SPI分为**主、从**两种模式，一个SPI通讯系统需要包含一个且只能是一个主设备，一个或多个从设备。
- 提供时钟的为主设备（Master），接收时钟的设备为从设备（Slave）。
- SPI接口的读写操作，都是由主设备发起。
- 当存在多个从设备时，**通过各自的片选信号进行管理**。
- SPI是全双工且SPI没有定义速度限制，一般的实现通常能达到甚至超过10 Mbps



10.1.2 SPI信号线

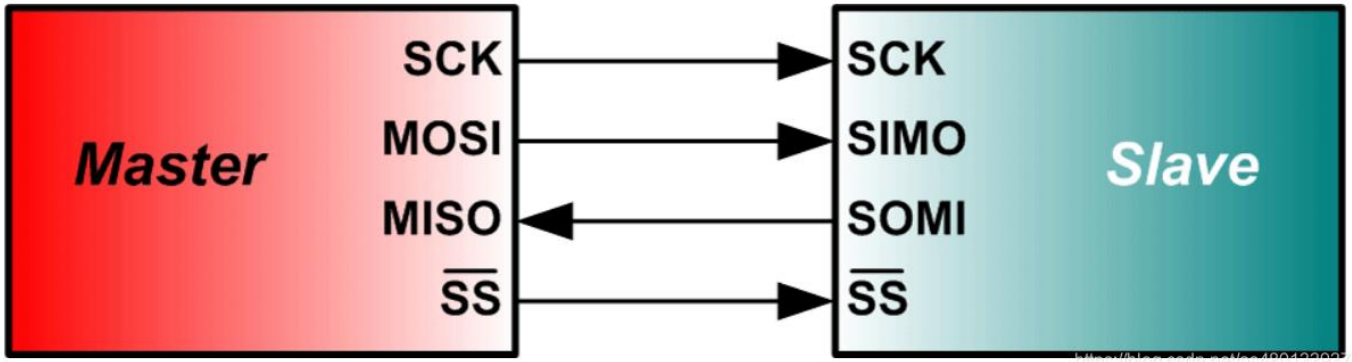
- SPI接口一般使用四条信号线通信：

MISO： 主设备输入/从设备输出引脚。该引脚在从模式下发送数据，在主模式下接收数据。

MOSI： 主设备输出/从设备输入引脚。该引脚在主模式下发送数据，在从模式下接收数据。

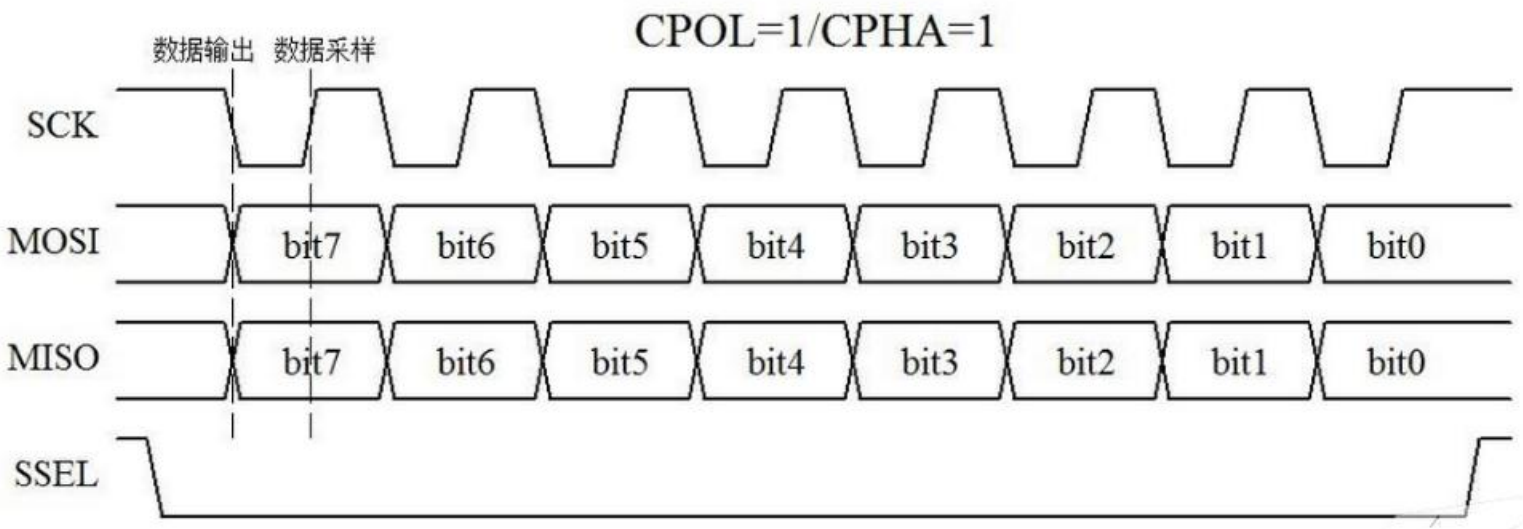
SCLK： 串行时钟信号，由主设备产生。

CS/SS： 从设备片选信号，由主设备控制。让主设备可以单独地与特定从设备通讯，避免数据线上的冲突。



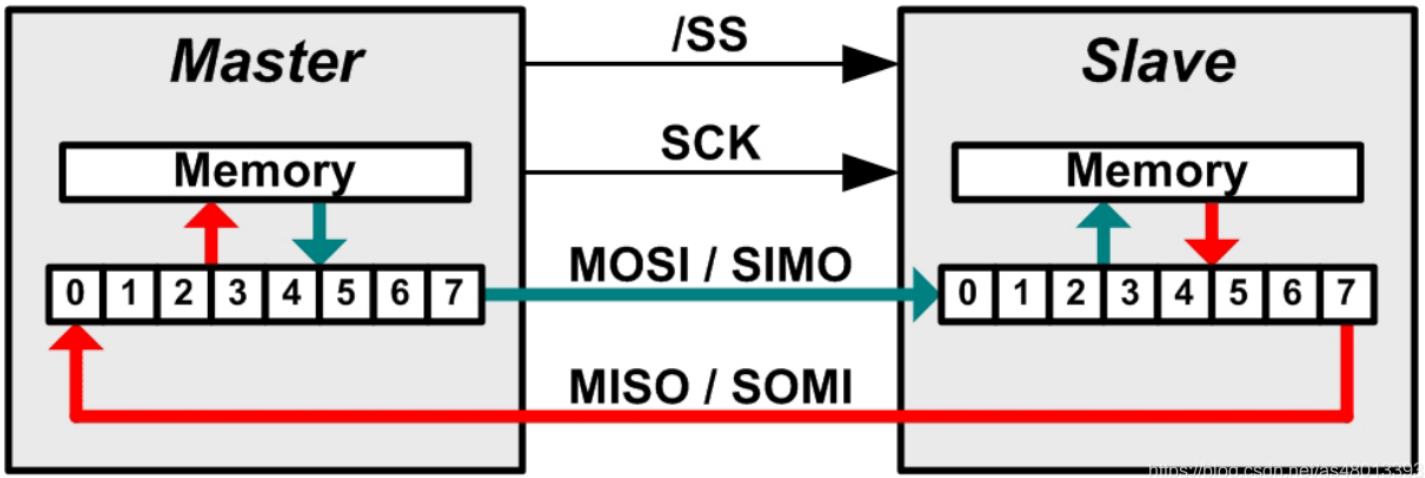
10.1.3 SPI设备选择

- SPI是**单主设备通信协议**，当SPI主设备想读/写从设备时，它首先拉低从设备对应的SS线（SS是低电平有效），接着开始发送工作脉冲到时钟线上，在相应的脉冲时间上，主设备把信号发到MOSI实现“写”，同时可对MISO采样而实现“读”



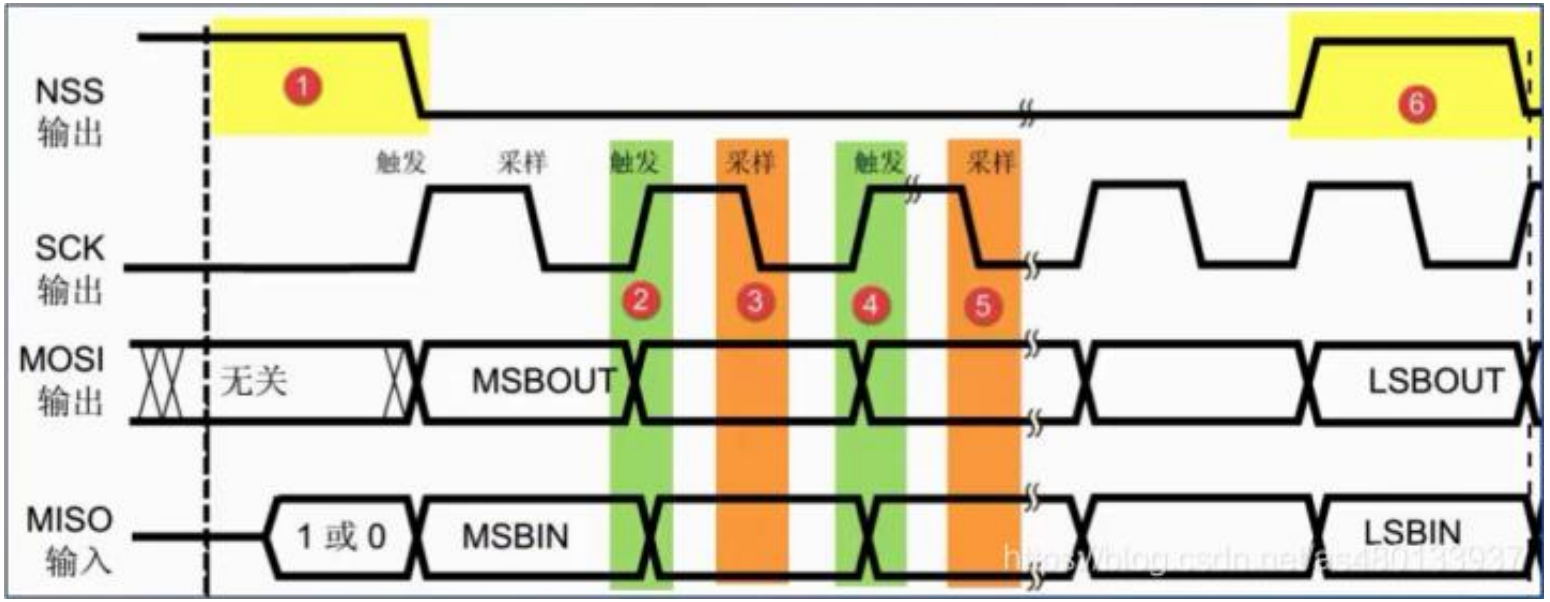
10.1.4 SPI数据发送接收

- SPI主机和从机都有一个**串行移位寄存器**，主机通过向它的SPI串行寄存器写入一个字节来发起一次传输。
- 主机将要发送的数据写到**发送数据缓存区**，缓存区经过移位寄存器(0~7)，串行移位寄存器通过MOSI信号线将字节一位一位的移出去传送给从机，
- MISO接口接收到的数据经过移位寄存器一位一位的移到**接收缓存区**。从机也将自己的串行移位寄存器(0~7)中的内容通过MISO信号线返回给主机。同时通过MOSI信号线接收主机发送的数据，两个移位寄存器中的内容就被交换。



10.1.4 SPI数据发送接收

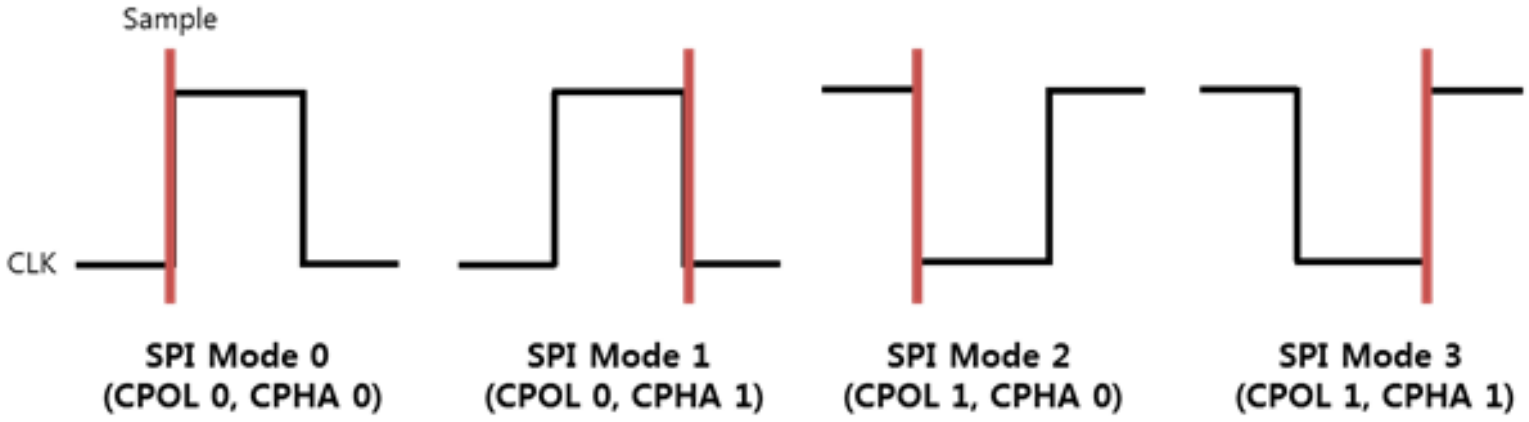
SPI时序图



10.2 SPI通信时序的四种配置

- **SPI通信有4种不同的配置模式**

通信双方必须是工作在同一模式下，通常可以对主设备的模式进行配置，通过CPOL（时钟极性）和CPHA（时钟相位）来定义主设备的通信模式。



10.2 SPI通信时序的四种配置

时钟极性(CPOL)定义了时钟空闲状态电平：

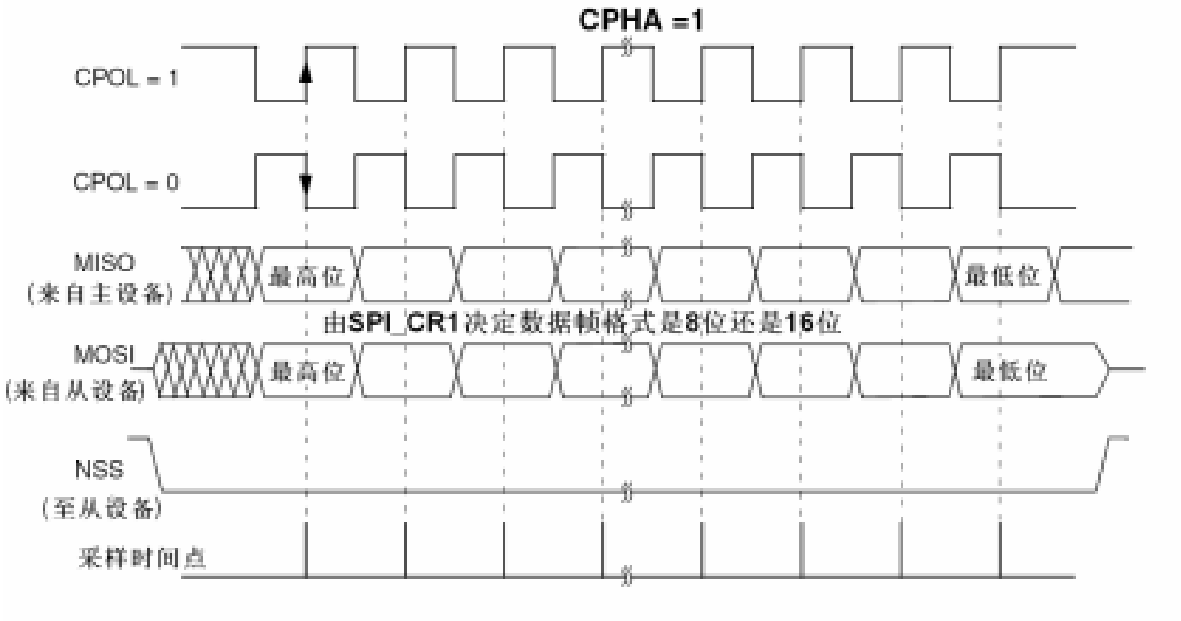
- CPOL=0，表示当SCLK=0时处于空闲态，SCLK=1处于有效状态
- CPOL=1，表示当SCLK=1时处于空闲态，SCLK=0处于有效状态

时钟相位(CPHA)定义数据的采集时间：

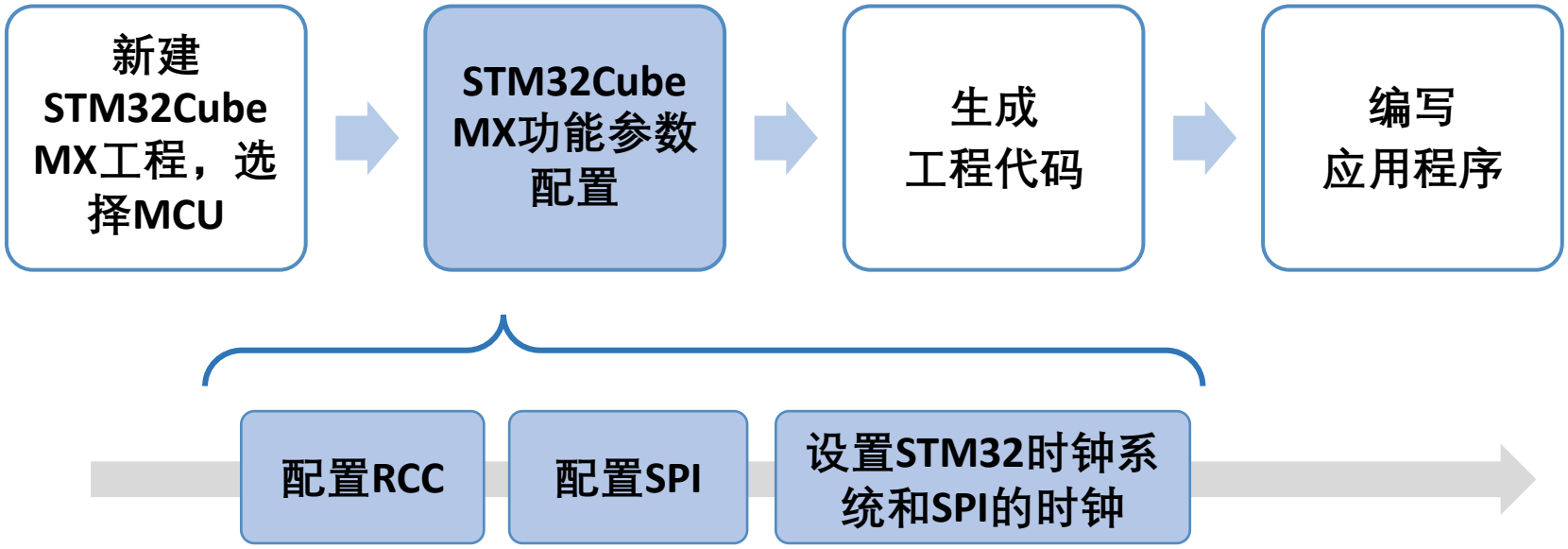
- CPHA=0，在时钟的第一个跳变沿（上升沿或下降沿）进行数据采样。在第2个边沿发送数据
- CPHA=1，在时钟的第二个跳变沿（上升沿或下降沿）进行数据采样。在第1个边沿发送数据

SPI 模式	CPOL	CPHA	空闲时 SCK 时钟	采样时刻
0	0	0	低电平	第 1 个边沿（奇）
1	0	1	低电平	第 2 个边沿（偶）
2	1	0	高电平	第 1 个边沿（奇）
3	1	1	高电平	第 2 个边沿（偶）

10.2 SPI通信时序的四种配置



10.3 基于HAL的SPI开发



10.3 基于HAL的SPI开发

SPI的HAL库常用接口函数

类型		函数及功能描述
引脚功能操作函数	轮询方式	HAL_SPI_Transmit(SPI_HandleTypeDef *hspi, const uint8_t *pData, uint16_t Size, uint32_t Timeout); 功能描述：SPI发送数据
		HAL_SPI_Receive(SPI_HandleTypeDef *hspi, uint8_t *pData, uint16_t Size, uint32_t Timeout); 功能描述：SPI接收数据
		HAL_SPI_TransmitReceive(SPI_HandleTypeDef *hspi, const uint8_t *pTxData, uint8_t *pRxData, uint16_t Size, uint32_t Timeout); 功能描述：SPI发送接收数据
	中断方式	HAL_SPI_Transmit_IT(SPI_HandleTypeDef *hspi, const uint8_t *pData, uint16_t Size); 功能描述：中断方式下SPI发送数据
		HAL_SPI_Receive_IT(SPI_HandleTypeDef *hspi, uint8_t *pData, uint16_t Size); 功能描述：中断方式下SPI接收数据
		HAL_SPI_TransmitReceive_IT(SPI_HandleTypeDef *hspi, const uint8_t *pTxData, uint8_t *pRxData, uint16_t Size); 功能描述：中断方式下SPI发送接收数据
	DMA方式	HAL_SPI_Transmit_DMA(SPI_HandleTypeDef *hspi, const uint8_t *pData, uint16_t Size); 功能描述：DMA方式下SPI发送数据
		HAL_SPI_Receive_DMA(SPI_HandleTypeDef *hspi, uint8_t *pData, uint16_t Size); 功能描述：DMA方式下SPI接收数据
		HAL_SPI_TransmitReceive_DMA(SPI_HandleTypeDef *hspi, const uint8_t *pTxData, uint8_t *pRxData, uint16_t Size); 功能描述：DMA方式下SPI接收数据
		HAL_SPI_GetState(const SPI_HandleTypeDef *hspi); 功能描述：轮询或中断方式下获取SPI状态
		HAL_SPI_GetError(const SPI_HandleTypeDef *hspi); 功能描述：轮询或中断方式下获取SPI状态

10.3 基于HAL的SPI开发

stm32f1xx_hal_spi.h

```
/* Initialization/de-initialization functions  ********************************/
HAL_StatusTypeDef HAL_SPI_Init(SPI_HandleTypeDef *hspi);
HAL_StatusTypeDef HAL_SPI_DeInit(SPI_HandleTypeDef *hspi);
void HAL_SPI_MspInit(SPI_HandleTypeDef *hspi);
void HAL_SPI_MspDeInit(SPI_HandleTypeDef *hspi);

/* Callbacks Register/UnRegister functions  ***********************************/
#if (USE_HAL_SPI_REGISTER_CALLBACKS == 1U)
HAL_StatusTypeDef HAL_SPI_RegisterCallback(SPI_HandleTypeDef *hspi, HAL_SPI_CallbackIDTypeDef CallbackID,
                                           pSPI_CallbackTypeDef pCallback);
HAL_StatusTypeDef HAL_SPI_UnRegisterCallback(SPI_HandleTypeDef *hspi, HAL_SPI_CallbackIDTypeDef CallbackID);
#endif /* USE_HAL_SPI_REGISTER_CALLBACKS */
/**
  * @}
  */

/** @addtogroup SPI_Exported_Functions_Group2
  * @{
  */
/* I/O operation functions  ***************************************************/
HAL_StatusTypeDef HAL_SPI_Transmit(SPI_HandleTypeDef *hspi, const uint8_t *pData, uint16_t Size, uint32_t Timeou
HAL_StatusTypeDef HAL_SPI_Receive(SPI_HandleTypeDef *hspi, uint8_t *pData, uint16_t Size, uint32_t Timeout);
HAL_StatusTypeDef HAL_SPI_TransmitReceive(SPI_HandleTypeDef *hspi, const uint8_t *pTxData, uint8_t *pRxData,
                                          uint16_t Size, uint32_t Timeout);
HAL_StatusTypeDef HAL_SPI_Transmit_IT(SPI_HandleTypeDef *hspi, const uint8_t *pData, uint16_t Size);
HAL_StatusTypeDef HAL_SPI_Receive_IT(SPI_HandleTypeDef *hspi, uint8_t *pData, uint16_t Size);
HAL_StatusTypeDef HAL_SPI_TransmitReceive_IT(SPI_HandleTypeDef *hspi, const uint8_t *pTxData, uint8_t *pRxData,
                                             uint16_t Size);
HAL_StatusTypeDef HAL_SPI_Transmit_DMA(SPI_HandleTypeDef *hspi, const uint8_t *pData, uint16_t Size);
HAL_StatusTypeDef HAL_SPI_Receive_DMA(SPI_HandleTypeDef *hspi, uint8_t *pData, uint16_t Size);
HAL_StatusTypeDef HAL_SPI_TransmitReceive_DMA(SPI_HandleTypeDef *hspi, const uint8_t *pTxData, uint8_t *pRxData,
                                              uint16_t Size);
HAL_StatusTypeDef HAL_SPI_DMAPause(SPI_HandleTypeDef *hspi);
HAL_StatusTypeDef HAL_SPI_DMAResume(SPI_HandleTypeDef *hspi);
HAL_StatusTypeDef HAL_SPI_DMAStop(SPI_HandleTypeDef *hspi);
/* Transfer Abort functions */
HAL_StatusTypeDef HAL_SPI_Abort(SPI_HandleTypeDef *hspi);
HAL_StatusTypeDef HAL_SPI_Abort_IT(SPI_HandleTypeDef *hspi);
```

初始化相关

三种不同收发模式

- 定义在**stm32f1xx_hal_spi.c**源文件
- 在**stm32f1xx_hal_spi.h**头文件中可以查看HAL库中SPI库函数的声明以及相关的结构体定义和串口一样

10.3 基于HAL的SPI开发

中断及回调函数

```
694 void HAL_SPI_IRQHandler(SPI_HandleTypeDef *hspi);
695 void HAL_SPI_TxCpltCallback(SPI_HandleTypeDef *hspi);
696 void HAL_SPI_RxCpltCallback(SPI_HandleTypeDef *hspi);
697 void HAL_SPI_TxRxCpltCallback(SPI_HandleTypeDef *hspi);
698 void HAL_SPI_TxHalfCpltCallback(SPI_HandleTypeDef *hspi);
699 void HAL_SPI_RxHalfCpltCallback(SPI_HandleTypeDef *hspi);
700 void HAL_SPI_TxRxHalfCpltCallback(SPI_HandleTypeDef *hspi);
701 void HAL_SPI_ErrorCallback(SPI_HandleTypeDef *hspi);
702 void HAL_SPI_AbortCpltCallback(SPI_HandleTypeDef *hspi);
703/**
704   * @}
705   */
706
707/** @addtogroup SPI_Exported_Functions_Group3
708   * @{
709   */
710 /* Peripheral State and Error functions ************************************
711 HAL_SPI_StateTypeDef HAL_SPI_GetState(const SPI_HandleTypeDef *hspi);
712 uint32_t             HAL_SPI_GetError(const SPI_HandleTypeDef *hspi);
713/**
714   * @}
```

状态及错误查询

- 定义在stm32f1xx_hal_spi.c源文件
- 在stm32f1xx_hal_spi.h头文件中可以查看HAL库中SPI库函数的声明以及相关的结构体定义和串口一样

10.3 基于HAL的SPI开发

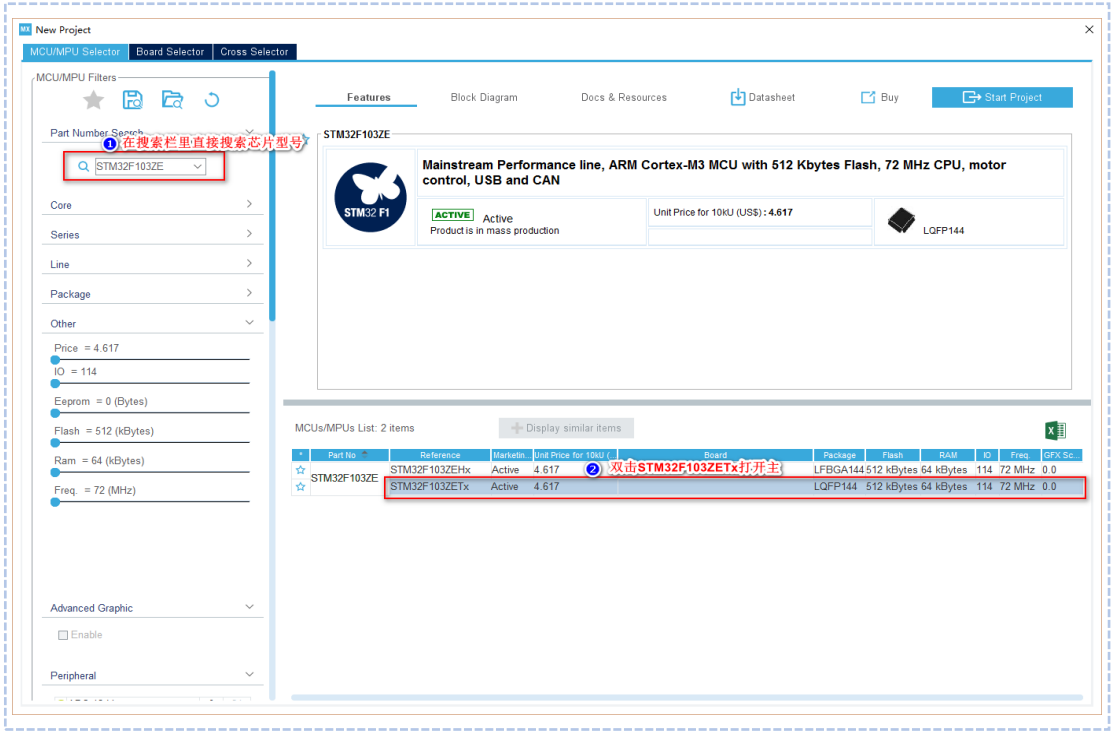
- **HAL_SPI_TransmitReceive函数**

```
1166
1167 /**
1168   * @brief  Transmit and Receive an amount of data in blocking mode.
1169   * @param  hspi pointer to a SPI_HandleTypeDef structure that contains
1170   *               the configuration information for SPI module.
1171   * @param  pTxData pointer to transmission data buffer
1172   * @param  pRxData pointer to reception data buffer
1173   * @param  Size amount of data to be sent and received
1174   * @param  Timeout Timeout duration
1175   * @retval HAL status
1176   */
1177 HAL_StatusTypeDef HAL_SPI_TransmitReceive(SPI_HandleTypeDef *hspi, const uint8_t *pTxData, uint8_t *pRxData,
1178                                           uint16_t Size, uint32_t Timeout)
```

10.3 基于HAL的SPI开发

软件设计——新建STM32CubeMX工程，选择MCU

新建STM32CubeMx工程，选择MCU，这里选择STM32F103ZETx芯片，读者可根据自己的目标板选择相应的芯片

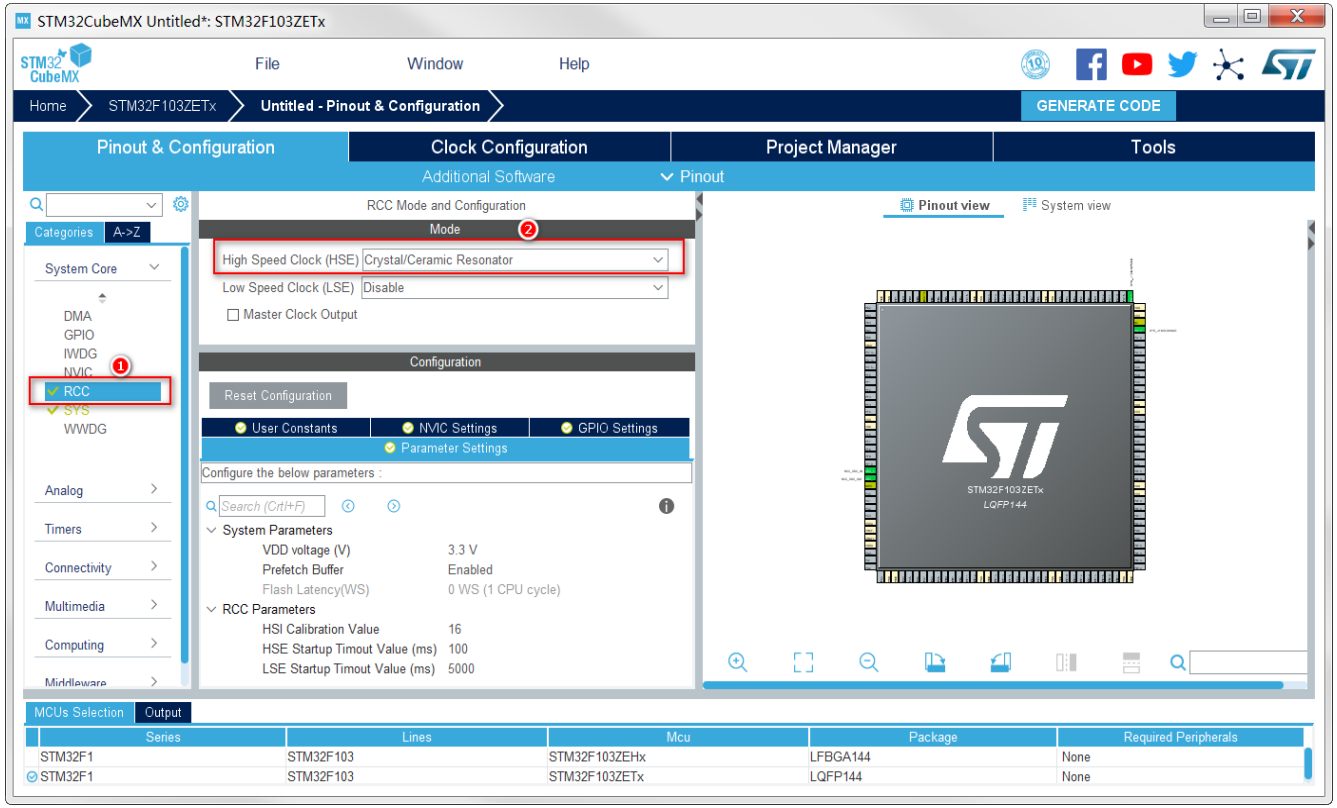


10.3 基于HAL的SPI开发

软件设计——STM32CubeMX功能参数配置

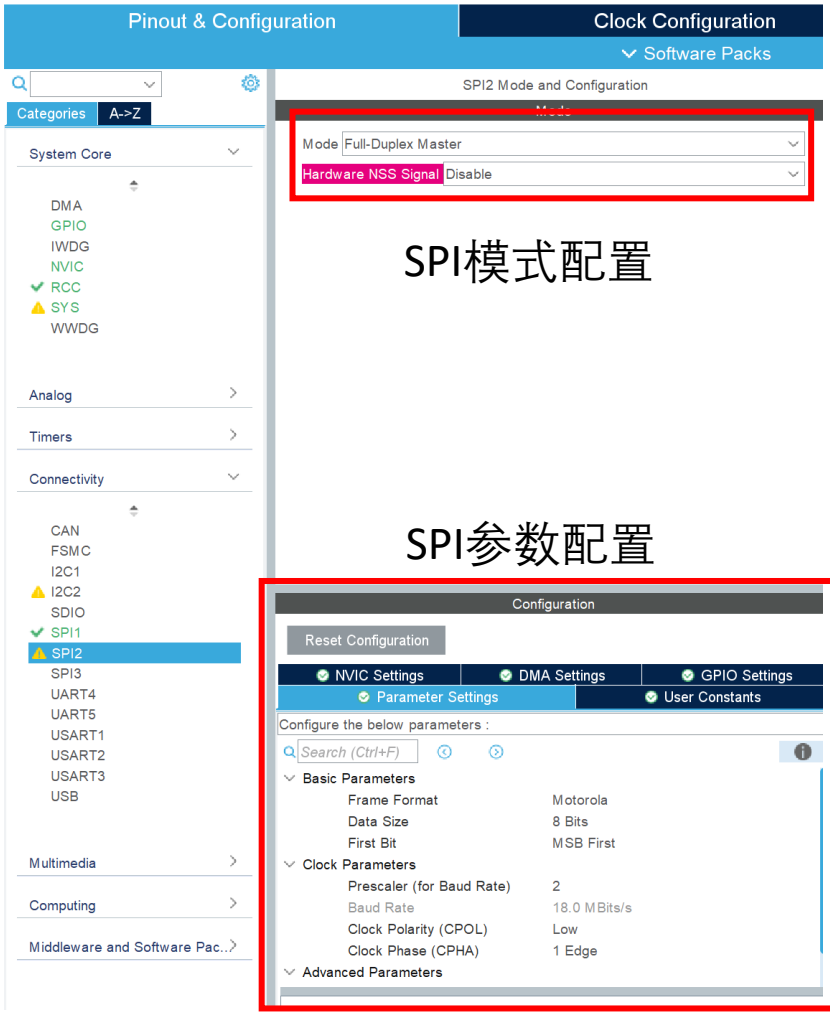
RCC配置

HSE选择“Crystal/Ceramic Resonator”（晶振/陶瓷谐振器），LSE选择“Disable”



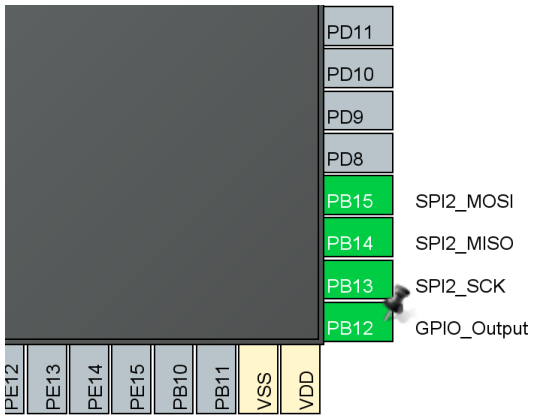
10.3 基于HAL的SPI开发

软件设计——STM32CubeMX功能参数配置



SPI模式配置

SPI参数配置

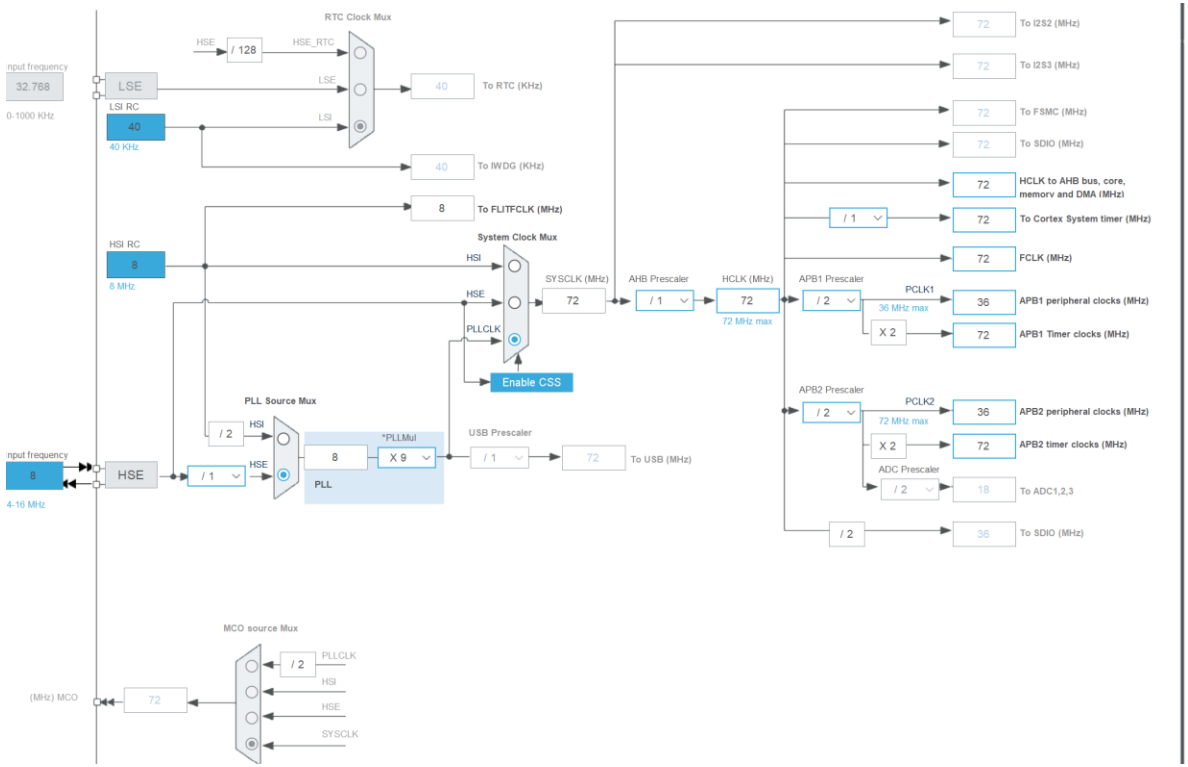


需配置软件控制的使能引脚，设置为GPIO_Output模式

10.3 基于HAL的SPI开发

软件设计——STM32CubeMX功能参数配置

设置STM32时钟系统和ADC的时钟

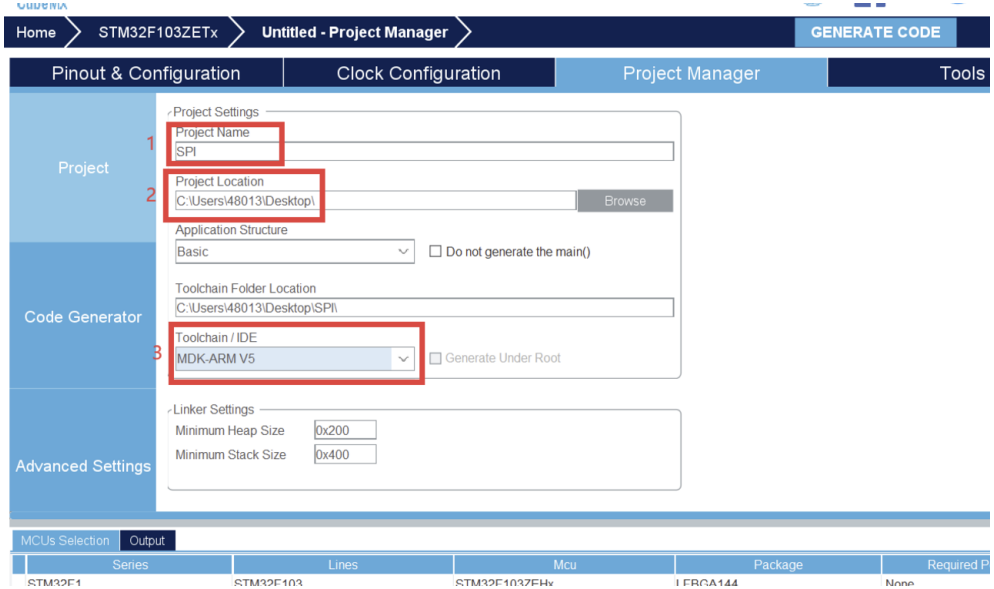


- 选择外部时钟HSE 8MHz
- PLL锁相环倍频9倍
- 系统时钟来源选择为PLL
- 设置APB1分频器为 /2
- 使能CSS监视时钟

10.3 基于HAL的SPI开发

软件设计——生成工程代码

配置keil工程名称和存放位置



配置工程名称和存放位置，输入“Project Name”为“SPI”，“Toolchain/IDE”选择“MDK-ARM V5”

10.3 基于HAL的SPI开发

stm32f1xx_hal_spi.h中找到SPI收发数据相关函数，并编写相关逻辑

```
/** @addtogroup SPI_Exported_Functions_Group2
  * @{
  */
/* I/O operation functions  ***************************************************/
HAL_StatusTypeDef HAL_SPI_Transmit(SPI_HandleTypeDef *hspi, const uint8_t *pData, uint16_t Size, uint32_t Timeout);
HAL_StatusTypeDef HAL_SPI_Receive(SPI_HandleTypeDef *hspi, uint8_t *pData, uint16_t Size, uint32_t Timeout);
HAL_StatusTypeDef HAL_SPI_TransmitReceive(SPI_HandleTypeDef *hspi, const uint8_t *pTxData, uint8_t *pRxData,
                                          uint16_t Size, uint32_t Timeout);
HAL_StatusTypeDef HAL_SPI_Transmit_IT(SPI_HandleTypeDef *hspi, const uint8_t *pData, uint16_t Size);
HAL_StatusTypeDef HAL_SPI_Receive_IT(SPI_HandleTypeDef *hspi, uint8_t *pData, uint16_t Size);
HAL_StatusTypeDef HAL_SPI_TransmitReceive_IT(SPI_HandleTypeDef *hspi, const uint8_t *pTxData, uint8_t *pRxData,
                                             uint16_t Size);
HAL_StatusTypeDef HAL_SPI_Transmit_DMA(SPI_HandleTypeDef *hspi, const uint8_t *pData, uint16_t Size);
HAL_StatusTypeDef HAL_SPI_Receive_DMA(SPI_HandleTypeDef *hspi, uint8_t *pData, uint16_t Size);
HAL_StatusTypeDef HAL_SPI_TransmitReceive_DMA(SPI_HandleTypeDef *hspi, const uint8_t *pTxData, uint8_t *pRxData,
                                              uint16_t Size);
```

10.3 基于HAL的SPI开发

软件模拟SPI通信方法（参考）

MySPI_SwapByte实现SPI读/写1byte

MySPI_W_SCK时钟引脚

MySPI_R_MISO输入数据引脚

MySPI_R_MOSI输出数据引脚

```
uint8_t MySPI_SwapByte(uint8_t ByteSend)
{
        uint8_t i, ByteReceive = 0x00;

        for(i = 0; i < 8; i ++)
        {
                MySPI_W_MOSI(ByteSend & (0x80 >> i));
                MySPI_W_SCK(1);
                if(MySPI_R_MISO() == 1)
                {
                        ByteReceive |= (0x80 >> i);
                }
                MySPI_W_SCK(0);
        }

        return ByteReceive;
}
```

本章小结

10.1 SPI协议基本概念

10.1.1 SPI主从模式

10.1.2 SPI信号线

10.1.3 SPI设备选择

10.1.4 SPI数据发送接收

10.2 SPI通信时序的四种配置

10.3 基于HAL的SPI开发

作业

选择一个采用SPI作为通信接口的传感器芯片(课上案例除外)，简单描述其功能，分析其采用的是哪种SPI通信模式，并说出使用STM32F103C8T6对该芯片采用硬件SPI进行通信可使用哪些引脚 (300字以内)。

作业提交：

学习平台s.ecust.edu.cn，提交截止时间：2025.11.23