第6章
总线通信接口I2C及SPI工作原理及应用

广西大学电气工程学院

资料整理
火龙果软件
荐

内容提要：

本章主要介绍I2C协议的软硬件构成，STM32的I2C工作原理、主要特点、内部结构、功能描述、I2C从模式、主模式等。分析了STM32平台上I2C接口的工作过程，使用固件函数库实现了I2C接口通信实验。了解了STM32的I2C接口在于软件程序设计层面及应用，I2C接口组网和灵活运用。依次讲述STM32的SPI通信原理、半双工/全双工、同步和串行方式通信。配置为主模式、从模式及通信时钟SCK，以多种配置方式工作。还介绍STM32的SPI的组网和相关应用。

内容安排

- I2C概论
- I2C通信原理
- SPI概述
- SPI通信原理

资料整理 火龙果软件 荐

6.1.1 I2C概述

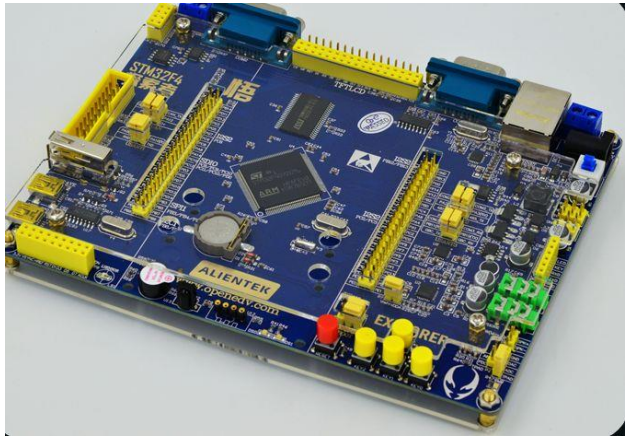
I2C（Inter Integrated Circuit）又被称为IIC、I^2C，最早由Philips（现被NXP收购）公司开发用于电视中CPU与外设通信两线串行短距离通信总线，由于其引脚少、硬件简单、通信速率较高、易于建立、可扩展性强，目前已经成为工业标准，被广泛地应用于微控制器、存储器和外设模块中。例如，STM32系列微控制器、EEPROM存储器模块24Cxx系列、温度传感器模块TMP102、光照传感器模块BH1750FVI、电子罗盘模块HMC5883L等都集成了I2C接口。

founded by Philips

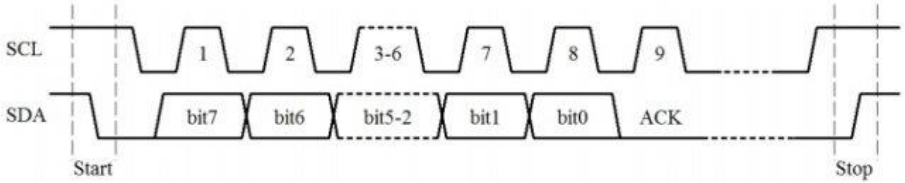
资料整理 火龙果软件 荐

6.1.2 I2C工作原理

STM32微控制器的I2C模块连接微控制器和I2C总线，提供多主机功能，支持标准和快速两种传输速率，控制所有I2C总线特定的时序、协议、仲裁和定时。STM32的I2C有多种用途，包括CRC码的生成和校验、SMBus （系统管理总线）和PMBus（电源管理总线）。根据特定设备的需要还可以使用DMA以减轻CPU的负担。



STM32微控制器



I²C 时序流程图

I2C时序流程图

资料整理 火龙果软件 荐

6.1.3 I2C主要特点

STM32的小容量产品有1个I2C，中等容量和大容量产品有2个I2C。I2C主要具有以下特性。

- 并行总线I2C总线协议转换器
- 多主机功能：该模块既可做主设备又可做从设备。
- I2C主设备功能：产生时钟、起始和停止信号。
- PC从设备功能：可编程的PC地址检测、可响应两个从地址的双地址能力、停止位检测；产生和检测7位/10位地址和广播呼叫。
- 支持不同的通信速度：标准速度（高达100kHz）。
- 状态标志：发送器/接收器模式标志、字节发送结束标志、I2C总线忙标志。
- 错误标志：主模式时的仲裁丢失、地址/数据传输后的应答（ACK）错误、检测到错位的起始或停止条件、禁止拉长时钟功能时的上溢或下溢。
- 两个中断向量：一个中断用于地址/数据通信成功；一个中断用于错误。
- 可选的拉长时钟功能。
- 具单字节缓冲器的DMA。
- 可配置PEC（信息包错误检测）的产生或校验：发送模式中PEC值可作为最后一个字节传输用于最后一个接收字节的PEC错误校验。
- 兼容SMBus2.0：25ms时钟低超时延时、10ms主设备累积时钟低扩展时间、25ms从设备累积时钟低扩展时间、带ACK控制的硬件PEC产生/效验、支持地址分辨协议（ARP）
- 兼容SMBus。

资料整理 火龙果软件 荐

6.1.4 I2C内部结构

STM32系列微控制器的I2C结构，由SDA线和SCL线展开（其中SMBALERT线用于SMBus），主要分为时钟控制、数据控制和控制逻辑等部分，负责实现I2C的时钟产生、数据收发、总线仲裁和中断、DMA等功能，如图6.1所示。

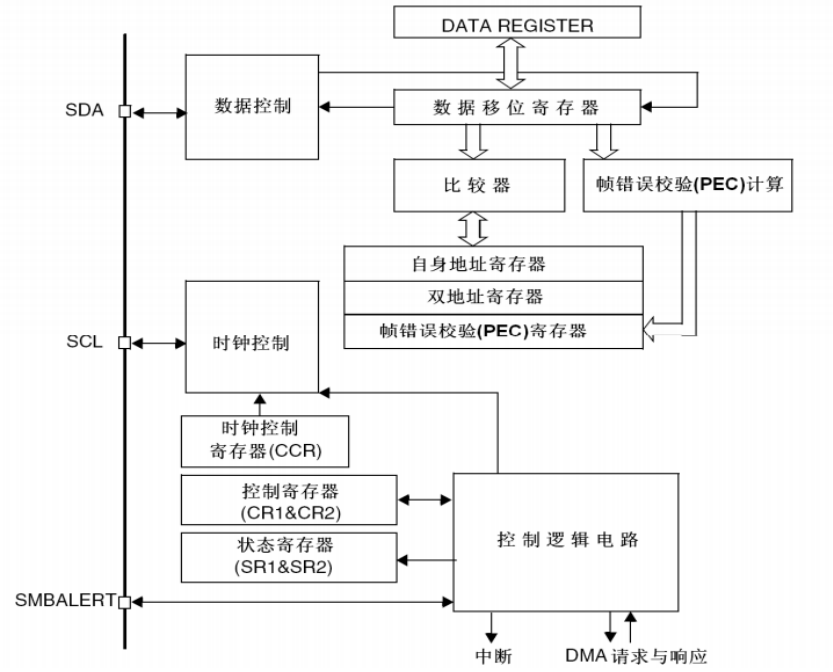


图6.1 STM32系列微控制器的I2C结构

资料整理
火龙果软件
荐

（1）时钟控制

时钟控制模块根据控制寄存器CCR、CR1和CR2中的配置产生I2C协议的时钟信号，即SCL线上的信号。为时序正确，在I2C_CR2寄存器中设定I2C的输入时钟。当I2C工作在标准传输速率时，输入时钟的频率≥ 2MHz；当I2C工作在快速传输速率时，输入时钟的频率≥ 4MHz。

（2）数据控制

包括起始、地址、应答和停止信号，将数据逐位从SDA线上发送出去。读取数据时，则从SDA线上的信号中提取接收到的数据。发送和接收的数据都被保存在数据寄存器（Data Register，DR）中。

（3）控制逻辑

控制逻辑用于产生I2C中断和DMA请求，观察分析表6.1

中断事件	事件标志	开启控制位
起始位已发送（主）	SB	ITEVFEN
地址已发送（主）或地址匹配（从）	ADDR	
10位头段已发送（主）	ADD10	
已收到停止（从）	STOPF	
数据字节传输完成	BTF	
接收缓冲区非空	RxNE	ITEVFEN 和 ITBUFEN
发送缓冲区空	TxE	
总线错误	BERR	ITERREN
仲裁丢失（主）	ARLO	
响应失败	AF	
过载/欠载	OVR	
PEC错误	PECERR	
超时/Tlow错误	TIMEOUT	
SMBus提醒	SMBALERT	

表6.1 I2C中断相关信息

资料整理 火龙果软件 荐

6.1.5 I2C功能描述

I2C模块接收和发送数据，并将数据从串行→并行，或从并行→串行，可以开启或禁止中断。接口通过数据引脚（SDA）和时钟引脚（SCL）连接到I2C总线，允许连接到标准（高达100kHz）或快速（高达400kHz）的I2C总线。

1. 模式选择，接口可按下述4种模式中的一种运行：

- 从发送器模式。
- 从接收器模式。
- 主发送器模式。
- 主接收器模式。

模块默认工作于从模式：接口在生成起始条件后自动将从模式切换到主模式；当仲裁丢失或产生停止信号时，则从主模式切换到从模式。允许多主机功能。

2. 通信流

主模式时，I2C接口启动数据传输并产生时钟信号。串行数据传输总是以起始条件开始并以停止条件结束。起始和停止条件都是在主模式下由软件控制产生。

从模式时，I2C 接口能识别它自己的地址（7位或10位）和广播呼叫地址。软件能控制开启或禁止广播呼叫地址的识别。

资料整理 火龙果软件 荐

数据和地址按8位/字节进行传输，高位在前。在起始条件后的1个或2个字节是地址（7位模式为1个字节，10位模式为2个字节）。地址只在主模式发送。在一个字节传输的8个时钟后第9个时钟期间，接收器必须回送一个应答位（ACK）给发送器，软件开启或禁止应答（ACK），并设置I2C接口的地址（7位、10位地址或广播呼叫地址）。

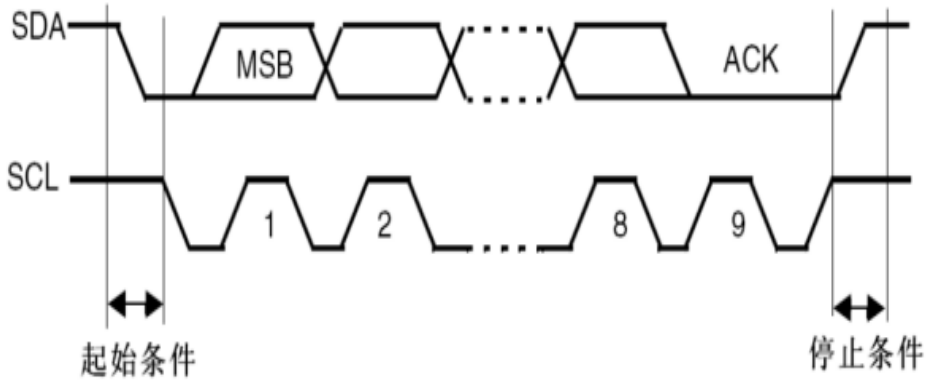


图6.2 I2C结构总线协议

起始条件：SCL线是“1”时，SDA线从“1”→“0”。
停止条件：SCL线是“1”时，SDA线从“0”→“1”。

资料整理 火龙果软件 荐

例题6.1：启动信号代码

```
void I2C_Start(void)
{   SDA_OUT();  //SDA 线输出
    SCL_OUT();  //SCL 线输出
    I2C_DELAY();  //IO输出
    SDA_SET();    //SDA置1
    SCL_SET();    //IO置高
    I2C_DELAY();    //延时
    SDA_CLR(); //SDA置0
    I2C_DELAY();          //延时
    I2C_DELAY();
    SCL_CLR();    //SCL为低   }
```

例题6.2：结束信号代码

```
void I2C_Stop(void)
{ SDA_OUT();  //SDA 线输出
  SCL_OUT();    //IO输出
  SDA_CLR();   //SDA置0
  SCL_CLR();   //SCL置0
  I2C_DELAY();  //延时
  SCL_SET();    //IO置0
  I2C_DELAY();
  I2C_DELAY();
  I2C_DELAY();    //延时
  SDA_SET();
  I2C_DELAY();
  I2C_DELAY();   //SDA置1}
```

资料整理
火龙果软件
荐

例题6.3：8位字节逐写入代码

```
//发送一个字节，返回从机有无应答，0为无应答，1为有应答
**************************************************/
uint8_t I2C_Send_byte(uint8_t data)
{ uint8_t k;
  //发送8bit数据
  for(k=0;k<8;k++)
{  I2C_DELAY();
   if(data&0x80)
    {
     SDA_SET(); //SDA置1
   }
   else{
     SDA_CLR(); //SDA置0
   }
   data=data<<1; //往前移动1次
   I2C_DELAY(); //调用延时
   SCL_SET();  //SCL置1
   I2C_DELAY();
   I2C_DELAY();
   SCL_CLR();  //SCL置0
  }
  //延时读取ACK响应
**************************************************/
I2C_DELAY();
 SDA_SET(); //SDA置1
 //置为输入线
 SDA_IN();   //SDA设置为输入
 I2C_DELAY();
 SCL_SET();  //SCL置1
 I2C_DELAY();
 //读数据
k=SDA_READ();
 if(k){ ////NACK响应
   return 0;
 }
 I2C_DELAY();
 SCL_CLR();  //SCL置0
 I2C_DELAY();
 SDA_OUT();  //SDA 线输出
 if(k){     //NACK响应
   return 0;
 }
 return 1;
}
//读 1 个字节，ack=1 时，发送 ACK，ack=0，发送 nACK
**************************************************/
```

资料整理 火龙果软件 荐

（接上页）

```
uint8_t I2C_Receive_byte(uint8_t flg)
{
 uint8_t k,data;
 //接收8bit数据
 //置为输入线
 SDA_IN();
 data=0;
 for(k=0;k<8;k++){
  I2C_DELAY();
  SCL_SET();   //SCL置1
  I2C_DELAY();
  //读数据
  data=data |SDA_READ();
  data=data<<1;
  I2C_DELAY();
  SCL_CLR(); //SCL置0
  I2C_DELAY();
 }
 data=data>>1; //往回移动1次
 //返回ACK响应
 //置为输出线
```

（接左栏）

```
SDA_OUT();  //SDA 线输出
 if(flg){
  SDA_SET();    //输出1-NACK
 }else{
  SDA_CLR();   //输出0-ACK
 }
 I2C_DELAY();
 SCL_SET(); / /SCL置1
 I2C_DELAY();
 I2C_DELAY();
 SCL_CLR();
 I2C_DELAY();
 SDA_OUT();  //SDA 线输出
 //返回读取的数据
 return (uint8_t)data;
}
```

资料整理
火龙果软件
荐

6.1.6 I2C从模式：

从模式是STM32微控制器I2C默认的工作模式。在从模式下，STM32可作为发送器或接收器，状态寄存器I2C_SR2中的TRA位标识了当前是发送器还是接收器。

1.从发生器：在接收到地址和清除ADDR位后，从发送器将字节从DR寄存器经由内部移位寄存器发送到SDA线上。从设备保持SCL为0，直到ADDR位被清除并且待发送数据已写入DR寄存器（图6.3中的EV1和EV3）。当收到应答脉冲，TxE位被硬件置位，如果设置了ITEVFEN和ITBUFEN位，则产生一个中断。如果TxE位被置位，在下一个数据发送结束之前没有新数据写入到I2C_ DR寄存器，则BTF位被置位，在清除BTF之前I2C接口将保持SCL为0；读出I2C_SR1后，再写入I2C_ DR寄存器清除BTF位。

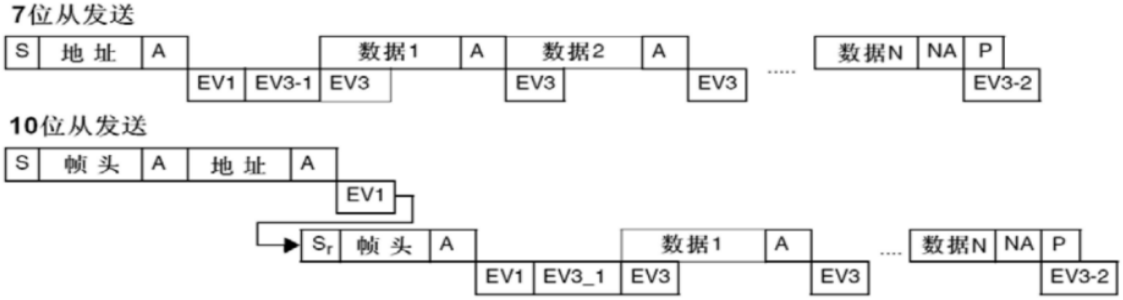


图6.3 从发送器的传输序列图

从发送器的传输序列图说明：S=Start(起始条件）；Sr=重复的起始条件；P=Stop（停止条件）；A=响应；NA=非响应；EVx=事件（ITEVFEN=1时产生中断）。

EV1：ADDR=1，读SR1再读SR2将清除该事件。

EV3_1：TxE=1，移位寄存器空，数据寄存器空，写DR。

EV3：TxE=1，移位寄存器非空，数据寄存器空，写DR将清除该事件。

EV3_2：AF=1，在SR1寄存器的AF位写0可清除AF位。

注意：

（1）EV1和EV3_1事件拉长SCL低的时间，直到对应的软件序列结束。

（2）EV3的软件序列应在当前字节传输结束之前完成。

资料整理 火龙果软件 荐

2.从接收器：在接收到地址并清除ADDR后，从接收器将通过内部移位寄存器，从SDA线接收到的字节存进DR寄存器。I2C接口在接收到每个字节后都执行下列操作：

如果设置了ACK位，则产生一个应答脉冲。

硬件设置RxNE=1。如果设置了ITEVFEN和ITBUFEN位，则产生一个中断。如果RxNE被置位，并且在接收新的数据结束之前DR寄存器未被读出，BTF位被置位，在清除BTF之前PC接口SCL将保持为0；读出I2C_SR1之后再写入I2C_DR寄存器将清除BTF位。

从接收器的接收序列如图6.4所示。

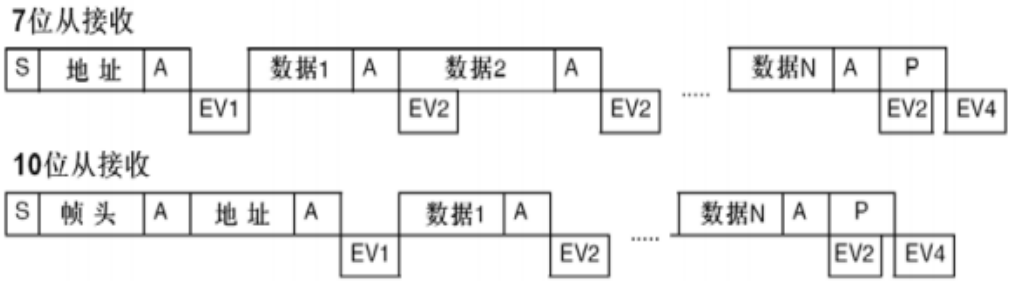


图6.4 从接收器的接收序列

图

从从接收器的传输序列图说明。S=Start(起始条件)；Sr=重复的起始条件；P-Stop（停止条件）；A=响应；NA=非响应；EVx=事件（ITEVFEN=1时产生中断）。

EV1：ADDR=1，读SR1再读SR2将清除该事件。

EV2：RxNE=1，读DR将清除该事件。

EV4：STOPF=1，读SR1然后写CR1寄存器将清除该事件。

注意：

（1） EV1事件拉长SCL低的时间，直到对应的软件序列结束。

（2） EV2的软件序列必须在当前字节传输结束之前完成。

资料整理 火龙果软件 荐

例题6.4：默认情况下，I2C接口总是工作在从模式，从模式切换到主模式，需要什么操作？

需要产生一个起始条件。为了产生正确的时序，必须在I2C_CR2寄存器中设定该模块的输入时钟。

输入时钟的频率必须至少是：

- 标准模式下为：2MHz
- 快速模式下为：4MHz

一旦检测到起始条件，在SDA线上接收到的地址被送到移位寄存器。然后与芯片自己的地址OAR1和OAR2（当ENDUAL=1）或者广播呼叫地址（如果ENGC=1）相比较。

注：

在10位地址模式时，比较包括头段序列（11110xx0），其中的xx是地址的两个最高有效位。

头段或地址不匹配：I2C接口将其忽略并等待另一个起始条件。

头段匹配（仅10位模式）：如果ACK位被置1，I2C接口产生一个应答脉冲并等待8位从地址。

地址匹配：I2C接口产生以下时序：

- 如果ACK被置1，则产生一个应答脉冲
- 硬件设置ADDR位；如果设置了ITEVFEN位，则产生一个中断
- 如果ENDUAL=1，软件必须读DUALF位，以确认响应了哪个从地址。

在10位模式，接收到地址序列后，从设备总是处于接收器模式。在收到与地址匹配的头序列并 且最低位为“1”（即11110xx1）后，当接收到重复的起始条件时，将进入发送器模式。将在从模式下TRA位指示当前是处于接收器模式还是发送器模式。

资料整理 火龙果软件 荐

6.1.7 I2C主模式

在主模式状态下，I2C接口启动数据传输并产生时钟信号。串行数据传输总是以起始条件开始并以停止条件结束。当通过START位在总线上产生了起始条件，设备就进入了主模式。在主模式下，STM32微控制器作为发送器，也可以作为接收器。状态寄存器I2C_AR2中的TRA位标识了当前是发送器还是接收器。

1.主接收器

（2）在发送地址和清除ADDR之后，I2C 接口进入主接收器模式。在此模式下，I2C 接口从SDA线接收数据字节，并通过内部移位寄存器送至DR寄存器。在每个字节后，I2C接口依次执行以下操作：

如果ACK位被置位，发出一个应答脉冲。

硬件设置RxNE=1，如果设置了INEVFEN和ITBUFEN位，则会产生一个中断（图6.5中的EV7）。

如果RNE位被置位，且在接收新数据结束前，DR寄存器中的数据没有被读走，硬件将设置BTF=1，在清除BTF之前I2C接口将保持SCL为低电平：读出I2C-SR1之后再读出I2C_DR寄存器将清除BTF位。

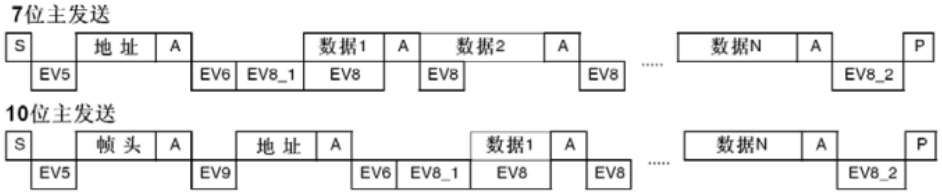


图6.5 主接收器传送序列图

图6.5说明S=Start（起始条件）；Sr=重复的起始条件；P-Stop（停止条件）；A=响应；NA=非响应；EVx=事件（ITEVFEN=1时产生中断）

资料整理 火龙果软件 荐

6.1.7 I2C主模式

2. 主发送器

在发送了地址和清除了ADDR位后，主设备通过内部移位寄存器将字节从DR寄存器发送到SDA线上。主设备等待，直到TxE被清除（图6.6 中的EV8）。

当收到应答脉冲时：TxE位被硬件置位，如果设置了INEVFEN和ITBUFEN位，则产生一个中断。

如果TxE被置位且在上一次数据发送结束之前没有写新的数据字节到DR寄存器，则BTF被硬件置位，在清除BTF之前I2C接口将保持SCL为低电平；读出I2C_SR1后再写入I2C_DR寄存器将清除BTF位。

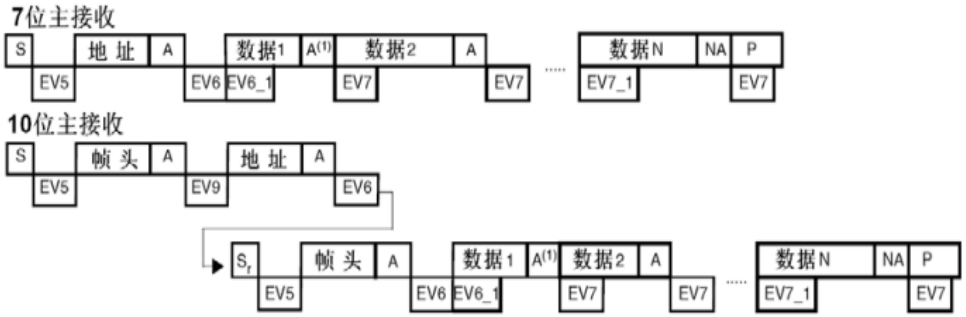


图6.6 主发送器的传输序列图

图6.6说明：S=Start（起始条件）；Sr=重复的起始条件；P=Stop（停止条件）；A=响应；NA=非响应；EVx=事件（ITEVFEN=1时产生中断）。

资料整理 火龙果软件 荐

例题6.5：在主模式时，I2C接口启动数据传输并产生时钟信号。串行数据传输总是以起始条件开始并以停止条件结束。当通过START位在总线上产生了起始条件，设备就进入了主模式，主模式操作分析？

主模式要求的操作顺序：

- 在I2C_CR2寄存器中设定该模块的输入时钟以产生正确的时序
- 配置时钟控制寄存器
- 配置上升时间寄存器
- 编程I2C_CR1寄存器启动外设
- 置I2C_CR1寄存器中的START位为1，产生起始条件

I2C模块的输入时钟频率至少是：

- 标准模式下为：2MHz
- 快速模式下为：4MHz

起始条件

当BUSY=0时，设置START=1，I2C接口将产生一个开始条件并切换至主模式（M/SL位置位）。

I2C_Mode //设置I2C的工作模式

I2C_Mode_I2C //设置I2C为I2C模式。

I2C_ Mode_ SMBusDevice //设置I2C为SMBus设备模式。

I2C Mode_ SMBusHost //设置I2C为SMBus主控模式。

注：

在主模式下，设置START位将在当前字节传输完后由硬件产生一个重开始条件。

一旦发出开始条件：

- SB位被硬件置位，如果设置了ITEVFEN位，则会产生一个中断。

然后主设备等待读SR1寄存器，接着将从地址写入DR寄存器（EV5）。

资料整理 火龙果软件 荐

从地址的发送

从地址通过内部移位寄存器被送到SDA线上。

- 在10位地址模式时，发送一个头段序列产生以下事件：

— ADD10位被硬件置位，如果设置了ITEVFEN位，则产生一个中断。

然后主设备等待读SR1寄存器，再将第二个地址字节写入DR寄存器。

— ADDR位被硬件置位，如果设置了ITEVFEN位，则产生一个中断。随后主设备等待一次读SR1寄存器，跟着读SR2寄存器。

- 在7位地址模式时，只需送出一个地址字节。一旦该地址字节被送出，

— ADDR位被硬件置位，如果设置了ITEVFEN位，则产生一个中断。随后主设备等待一次读SR1寄存器，跟着读SR2寄存器。根据送出从地址的最低位，主设备决定进入发送器模式还是进入接收器模式。

- 在7位地址模式时，

— 要进入发送器模式，主设备发送从地址时置最低位为0。

— 要进入接收器模式，主设备发送从地址时置最低位为1。

- 在10位地址模式时

— 要进入发送器模式，主设备先送头字节（11110xx0），再送最低位为0的从地址。

— 要进入接收器模式，主设备先送头字节（11110xx0），再送最低位为1的从地址。然后重新发送一个开始条件，后面跟着头字节（11110xx1）

TRA位指示主设备是在接收器模式还是发送器模式。

资料整理 火龙果软件 荐

6.2 I2C通信原理

作为半双工同步串行通信接口，仅使用两根线就完成了数据传输，方便地构成多机系统和外围器件扩展系统。I2C采用器件地址的硬件设置方法，通过软件寻址避免了像SPI的器件片选线寻址，简化了STM32和外围器件之间的连接。只要满足I2C标准的器件，都可以连接在同条总线上，便于建立复杂的网络及随时增加和删除节点。I2C通信要求被寻址的设备发回应答信息，以满足容错要求。

I2C的不足：易受干扰、不检查错误、传输速率有限。

I2C的常用术语：

主机：初始化发送、产生时钟和终止发送的器件，通常是微控制器。

从机：被主机寻址的器件。

发送器：本次传输中发送数据到I2C总线的器件，可以是主机或从机，由通信过程具体确定。

接收器：本次传输中从I2C总线上接收数据的器件，可以是主机或从机，由通信过程具体确定。

连接在I2C总线上的器件既是主机（或从机），又是发送器（或接收器），取决于要完成的具体功能。

资料整理 火龙果软件 荐

6.2.1 I2C的物理层

1. I2C接口

I2C是半双工同步串行通信，相比USART和SPL，它所需的信号少，只需SCL和SCK两根线。

（1）SCL（串行时钟线）：I2C通信中用于传输时钟的信号线，由主机发出。SCL采用集电极开路或漏极开路的输出方式。I2C 器件只能使SCL下拉到逻辑“0”，而不能强制SCL上拉到逻辑“1”。

（2）SDA （串行数据线）： I2C通信中用于传输数据的信号线。与SCL类似，SDA采用集电极开路或漏极开路的输出方式。I2C器件只能使SDA下拉到逻辑“0”，而不能强制SDA上拉到逻辑“1”。

2. I2C互连

I2C总线（即SCL和SDA）上可以连接多个I2C器件，如图6.6所示。I2C互连主要有以下特点。STM32属于ARM内核的一个版本，远强于传统的51单片机，有很多资源是51不具备的，如USB控制器。

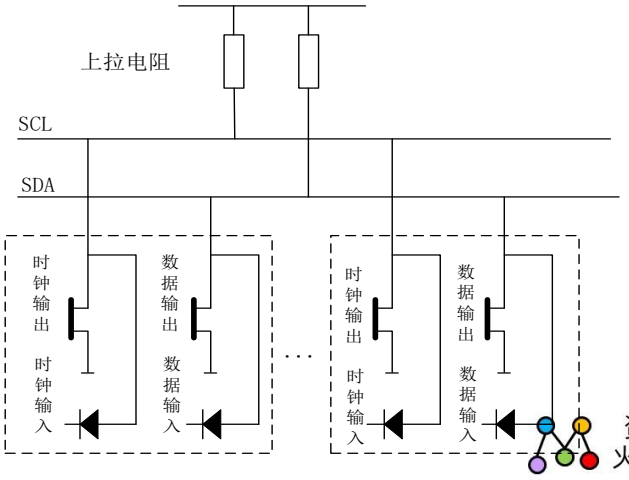


图6.6 I2C总线上的互连

资料整理
火龙果软件
荐

（1）必须在I2C总线上外接上拉电阻。由于I2C总线（SCL和SDA）采用集电极开路或漏极开路的输出方式，连接到I2C总线上的器件只能使SCL或SDA置0。应在SCL和SDA上外加上拉电阻，使两根信号线进行置1，才能正确进行数据通信。

当一个I2C器件将信号线下拉到逻辑“0”并信号消失，上拉电阻将该线重置为逻辑“1”。I2C标准规定这段时间（SCL或SDA的上升时间）应小于1000ns。由于分布电容的存在，节点越多该电容越高（最大400pF），根据RC时间常数算出上拉电阻值，约为1~5.1kΩ，选用5.1kΩ（5V）或4.7kΩ（3.3V）。

（2）多个I2C器件可以并联在I2C总线上，I2C使用地址来识别总线上的器件，易于器件的扩展。在I2C互连系统中，每个I2C器件都有一个独立的身份标识（ID）——器件地址（address），如图6.7 所示。

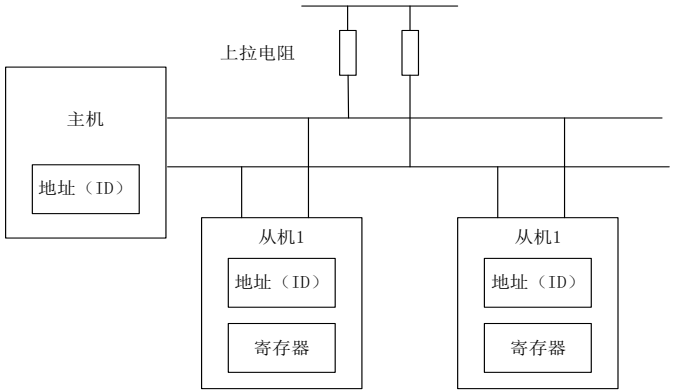


图6.7 I2C总线上的地址

（3）支持多主机互连。I2C带有竞争检测和仲裁电路，实现了真正的多主机互连。当多主机同时使用总线发送数据时，根据仲裁方式决定由哪个设备占用总线，以防止数据冲突和数据丢失。尽管I2C支持多主机互连，但同时只能有一个主机运行。

资料整理
火龙果软件
荐

6.2.2 I2C的协议层

I2C是同步串行半双工通信，按照SCL的时钟节拍在SDA上逐位进行数据传输。SCL每产生一个时钟脉冲，SDA上传输一位数据。

I2C的协议层包括：位传输、字节传输、数据流传输、传输模式和传输速率5个方面。

1.位传输

I2C的时序包括：起始条件、数据有效性、停止条件等。

（1）起始条件（起始信号）。当SCL为高电平时，SDA上由高到低地跳变。

（2）数据有效性。SDA的数据线应在SCL为高电平时保持稳定，只能在SCL为低电平时改变。否则，会被误判为起始位或停止位。

（3）停止条件（停止信号）。当SCL为高电平时，SDA上由低到高地跳变。

2. 字节传输

I2C上所有数据都是以1B（8b）为最小单位，按高位（MSB）在前、低位（LSB）在后的顺序在SDA上传输。

每当发送器发送完一个字节，接收器应发送一个应答位（ACK）以确认接收器是否成功收到数据。

3.数据流传输

一次标准的I2C通信由起始信号、从机地址传输、数据传输和停止信号组成。

4.传输模式

I2C主要的两种数据传输模式：

- 主机向从机写数据。
- 主机向从机读数据。

5.传输速率

标准传输速率为100Kbps，快速传输可达到400Kbps。

资料整理 火龙果软件 荐

例题6.6：分析SDA/SCL线控制有几种模式？

- 如果允许时钟延长：

— 发送器模式：如果TxE=1且BTF=1：I2C接口在传输前保持时钟线为低，以等待软件读SR1，再把数据写进数据寄存器（缓冲器和移位寄存器都是空的）。

— 接收器模式：如果RxNE=1且BTF=1：I2C接口在接收到数据字节后保持时钟线为低，以等待软件读SR1，再读数据寄存器DR（缓冲器和移位寄存器都是满的）。

- 如果在从模式中禁止时钟延长：

— 如果RxNE=1，在接收到下个字节前DR还没有被读出，则发生过载错。接收到的最后一个字节丢失。

— 如果TxE=1，在必须发送下个字节之前却没有新数据写进DR，则发生欠载错。相同的字节将被重复发出。

— 不控制重复写冲突。

例题6.7：当接口检测到一个无应答时，发生应答错应答错误。此时会发生什么？

- AF位被置位，如果设置了ITERREN位，则产生一个中断；
- 当发送器接收到一个NACK时，必须复位通讯。

— 如果是处于从模式，硬件释放总线。

— 如果是处于主模式，软件必须生成一个停止条件。

资料整理 火龙果软件 荐

例题6.8：从模式时禁止时钟延长，怎么判断过载/欠载错误。

在从模式下，如果禁止时钟延长，I2C接口正在接收数据时，当它已经接收到一个字节（RxNE=1），但在DR寄存器中前一个字节数据还没有被读出，则发生过载错误。此时：

- 最后接收的数据被丢弃；
- 在过载错误时，软件应清除RxNE位，发送器应该重新发送最后一次发送的字节。

在从模式下，如果禁止时钟延长，I2C接口正在发送数据时，在下一个字节的时钟到达之前，新的数据还未写入DR寄存器（TxE=1），则发生欠载错误。此时：

- 在DR寄存器中的前一个字节将被重复发出；
- 应该确定在发生欠载错时，接收端应丢弃重复接收到的数据。发送端应按I2C总线标准在规定的时间更新DR寄存器。在发送第一个字节时，清除ADDR之后在第一个SCL上升沿之前写入DR寄存器；否则接收方应该丢弃第一个数据。

例题6.9：如何从I2C模式切换到SMBus模式。

应该执行下列步骤：

- 设置I2C_CR1寄存器中的SMBus位；
- 按要求配置I2C_CR1寄存器中的SMBTYPE和ENARP位。

如果要把设备配置成主设备，产生起始条件的步骤见I2C主模式。否则参见I2C从模式。 软件程序必须处理多种SMBus协议。

- 如果ENARP=1且SMBTYPE=0，使用SMB设备默认地址。
- 如果ENARP=1且SMBTYPE=1，使用SMB主设备头字段。
- 如果SMBALERT=1，使用SMB提醒响应地址。

资料整理 火龙果软件 荐

6.3 SPI概述

SPI（Scrial Pepheal Interface）是串行外围设备接口的缩写，是一种高速、全双工、同步的通信总线，SPI在芯片的引脚上仅用4根线。由于其简单方便、成本低廉、传输速度快，成为事实上的标准。

目前，许多微控制器、存储器和外设模块都集成了SPI接口，如STM32系列微控制器、EEPROM、FLASH存储器W25X16、双路16位ADC数据采集模块AD7705、MicroSD卡模块、TFT LCD模块等。

在大容量产品和互联型产品上，SPI支持SPI协议或者I2S音频协议，通过软件把功能从SPI模式切换到I2S模式。在中小容量产品上，不支持I2S音频协议。

I2S是一种3引脚的同步串行接口通信协议。在半双工通信中，可以工作在主和从两种模式下。作为主设备时，通过接口向外部的从设备提供时钟信号。

SPI广泛应用于高速通信的场合，STM32通过SPI组成一个小型同步网络，进行高速数据交换，完成复杂的工作。

资料整理
火龙果软件
荐

6.3.1 SPI工作原理及主要特征

1.SPI工作原理：STM32的SPI模块允许与外部设备以半双工/全双工、同步和串行方式通信。常被配置为主模式，为多个从设备提供通信时钟SCK，以多种配置方式工作。STM32的SPI模块有多种用途，如使用一条双向数据线的双线单工同步传输，使用CRC（循环冗余校验）检验的可靠通信等。

2.SPI主要特征：STM32的小容量产品有1个SPI、中等容量产品有2个SPI、大容量产品则有3个SPI。

STM32的SPI主要具有以下特性：

- 3线全双工同步传输。
- 带或不带第三根双向数据线的双线单工同步传输。
- 8位或16位传输帧格式选择。
- 主或从操作。
- 支持多主模式。
- 8个主模式波特率预分频系数（最大为$f_{PCLK}/2$）。
- 从模式频率（最大为$f_{PCLK}/2$ ）。
- 主模式和从模式的快速通信。
- 主模式和从模式下可以由软硬件进行NSS管理；主/从操作模式的动态改变。
- 可编程的时钟极性和相位。

资料整理 火龙果软件 荐

6.3.2 SPI功能介绍

STM32的SPI功能如图6.8所示，主要由波特率控制、收发控制和数据储存转移三部分构成。

1.波特率控制

特率发生器产生的SPI的SCK时钟信号。波特率预分频系数为2、4、8、16、32、64、128或256。通过设置波特率控制位（BR）可以控制SCK的输出频率，从而控制SPI的传输速率。

2.收发控制

收发控制由若干个控制寄存器组成，如SPI控制寄存器SPI_CR1、SPI_CR2和SPI状态寄存器SPI_SR等。

SPI_CR1寄存器主控收发电路，用于设置SPI的协议，如时钟极性、相位和数据格式等。

SPI_CR2寄存器用于设置各种SPI中断使能，如使能TXE的TXEIE和使能RXNE的RXNEIE等。

通过SPL_SR寄存器中的各个标志位可以查询SPI当前的状态。

SPI的控制和状态查询可以通过库函数实现，只需使用SPI相关的库函数。

3.数据存储转移

数据存储转移如图6.8所示的左上部分，主要由移位寄存器、接收缓冲区和发送缓冲区等构成。

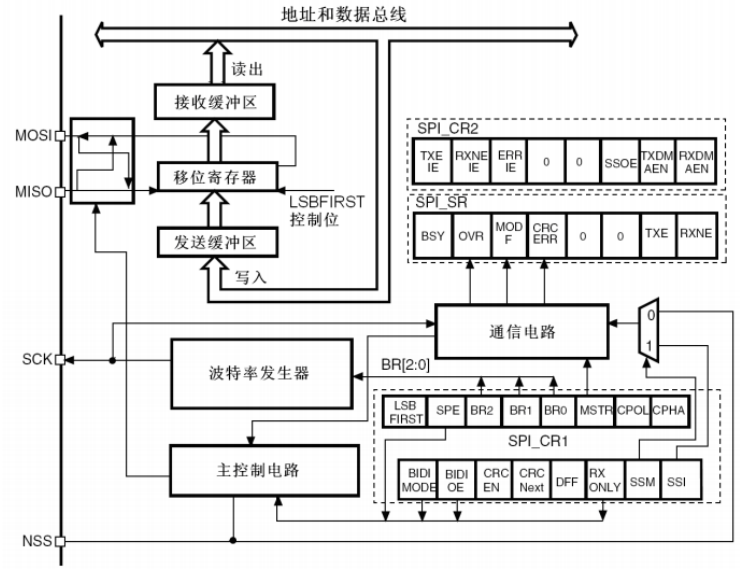


图6.8 STM32系列微控制器SPI的功能框图

资料整理
火龙果软件
荐

6.3.3 SPI工作模式

1.配置SPI为从模式:

在从模式下，SCK引脚用于接收从主设备来的串行时钟。SPI_CR1寄存器中BR[2:0]的设置不影响数据传输速率。

注意：建议在主设备发送时钟之前使能SPI从设备，否则可能发生意外的数据传输。在通信时钟的第一个边沿到来之前或正在进行的通信结束之前，从设备的数据寄存器必须就绪。在使能从设备和主设备之前，通信时钟的极性应稳定。

（1）配置步骤。

- 设置DFF位以定义数据帧格式为8位或16位。
- 选择CPOL和CPHA位定义数据传输和串行时钟之间的相位关系。为保证数据传输的正确性，从设备和主设备的CPOL和CPHA位应配置成相同的方式。
- 帧格式（SPI_CR1寄存器中的LSBFIRST位定义的“MSB在前”还是“LSB在前”）应与主设备相同。
- 硬件NSS模式下，在完整的数据帧（8位或16位）传输过程中，NSS引脚为低电平。软件NSS模式下，设置SPI_CR1寄存器中的SSM位并清除SSI位。
- 清除MSTR位、设置SPE位（SPI_CR1寄存器），使相应引脚工作于SPI模式下。

在这个配置中MOSI引脚是数据输入，MISO引脚是数据输出。

（2）数据发送过程

在写操作中，数据字同时写入发送缓冲器。

当从设备收到时钟信号，且在MOSI引脚上出现第一个数据位时，发送过程开始（注：此时第一位被发送出去）。余下的位（对于8位数据帧格式，还有7位，其他依次类推）被装进移位寄存器。当发送缓冲器中的数据传输到移位寄存器时SPI_SPR寄存器的TXE标志被设置，如果设置了SPI_CR2寄存器的TXEIE位，将会产生中断。

资料整理 火龙果软件 荐

6.3.3 SPI工作模式

（3）数据接收过程

对于接收器来说，当数据传输完成时：

· 移位寄存器里的数据传送到接收缓冲器，SPI_SR寄存器中的RXNE标志被设置。

· 如果设置了SPI_CR2寄存器中的RXNEIE位，则产生中断。

· 在最后一个采样时钟边沿后，RXNE位被设置，移位寄存器中接收到的数据被传送到接收缓冲器。读SPI_DR寄存器时，SPI设备返回。

2.配置SPI为主模式：

在主配置时，SCK引脚产生串行时钟。

（1）配置步骤。

- 通过SPI_CR1寄存器的BR[2:0]位定义串行时钟波特率。
- 选择CPOL和CPHA位定义数据传输和串行时钟间的相位关系。
- 设置DFF位定义8位或16位数据帧格式。
- 配置SPI_CR1寄存器的LSBFIRST位定义帧格式。
- 硬件NSS模式下，在整个数据帧传输期间把NSS引脚连接到高电平；在软件NSS模式下，需设置SPL_CR1寄存器的SSM位和SSI位。设置SSOE位，NSS引脚工作在输出模式。
- 必须设置MSTR位和ISPE位（只有当NSS引脚被连接到“1”，这些位一直置位）。在这个配置中，MOSI引脚是数据输出，而MISO引脚是数据输入。

（2）数据发送过程。

当写入数据至发送缓冲器时，发送过程开始的。

在发送第一个数据位时，数据字由并/串移动到MOSI引脚上；MSB与LSB的顺序取决于SP_CR1寄在器中的LSBFIRST位的设置。数据从发送缓冲器传输到移位寄存器时TXE标志将被置位，如果设置了SPI_CR2寄存器中的TXEIE位，将产生中断。

资料整理 火龙果软件 荐

6.3.3 SPI工作模式

（3）数据接收过程。

对于接收器来说，当数据传输完成时：

- 移位寄存器里的数据传送到接收寄存器，并且RXNE标志被置位。
- 如果设置了SPI_CR2寄存器中的RXNEIE位，则产生中断。

在最后一个采样时钟边沿后，RXNE位被设置，移位寄存器中接收到的数据字被传送到接收缓冲器，读SPI_DR寄存器将清除RXNE位。

当传输开始，将发送的数据放入发送缓冲器，维持连续的传输流。在写发送缓冲器之前，确认TXE标志为1。

注意：在NSS硬件模式下，从设备的NSS输入由NSS引脚控制，或通过软件由GPIO引脚控制。

6.3.4 SPI发送数据和接收数据

在STM32使用SPI发送数据前，已经完成SPI物理层（如引脚）和协议层（时钟极性、时钟相位、数据格式和传输速率）的相关配置，并将数据写入发送缓冲区，进行SPI数据的收发。

1. SPI发送数据

STM32的SPI发送一个字节数据的具体流程：先检测SPI1的发送缓冲区是否为空（TXE标志位置位），若为空，则将要发送的字节数据写入发送缓冲区；再继续检测SPI1的接收缓冲区是否为空（RXNE标志位置位），若不为空，则从接收缓冲区读取收到的一个字节数据并返回，发送结束。

资料整理 火龙果软件 荐

6.3.4 SPI发送数据和接收数据

2. SPI接收数据

SPI在发送二个字节数据的同时会收到一个字节的数据。因此，STM32的SPI接收一个字节数据和发送一个字节数据的过程基本相同。不同在于要使用SPI接收数据，其发送的数据是一个空字节而不是有意义的指令字节。在编程实现时，SPI数据的发送和接收通常使用同一个函数实现，调用时对参数不同的赋值进行区别。

例题6.8：从选择（NSS）脚管理

有2种NSS模式：

- **软件NSS模式：**通过设置SPI_CR1寄存器的SSM位来使能这种模式。在这种模式下NSS引脚可以用作它用，而内部NSS信号电平通过写SPI_CR1的SSI位来驱动。
- **硬件NSS模式，分两种情况：**

— **NSS输出被使能：**当STM32工作为主SPI，并且NSS输出已经通过SPI_CR2寄存器的SSOE位使能，这时NSS引脚被拉低，所有NSS引脚与这个主SPI的NSS引脚相连并配置为硬件NSS的SPI设备，将自动变成从SPI设备。

当一个SPI设备需要发送广播数据，它必须拉低NSS信号，通知其它的设备它是主设备；如果它不能拉低NSS，这意味着总线上有另外一个主设备在通信，这时将产生一个硬件失败错误（Hard Fault）。

— **NSS输出被关闭：**允许操作于多主环境。

资料整理 火龙果软件 荐

6.4 SPI通信原理

SPI为全双工同步串行通信接口，它采用主/从模式支持一个或多个从设备，实现主设备和从设备之间的高速数据通信。SPI 的缺点：无法检错纠错、不具备寻址能力和接收方没有应答信号等，不适合复杂或可靠性要求较高的场合。

6.4.1 SPI的物理层

1.SPI接口

SPI是全双工同步串行通信。SPI有一条公共的时钟线实现同步；全双工时，SPI至少有两根数据线实现数据的双向同时传输；串行时，SPI收发数据只能逐位地在数据线上传输，一根发送数据线和一根接收数据线。

SPI在物理层体现为4根信号线，分别是SCK、MOSI、MISO和SS。

SCK （Serial Clock），即时钟线。由主设备产生，作为主设备、从设备的时钟信号。

MOSI （Master Input Slave Output）即主设备数据输出/从设备数据输入线。在主模式下发送数据，在从模式下接收数据。

MISO （Master Output Slave Input），即主设备数据输入/从设备数据输出线。在主模式下接收数据，在从模式下发送数据。

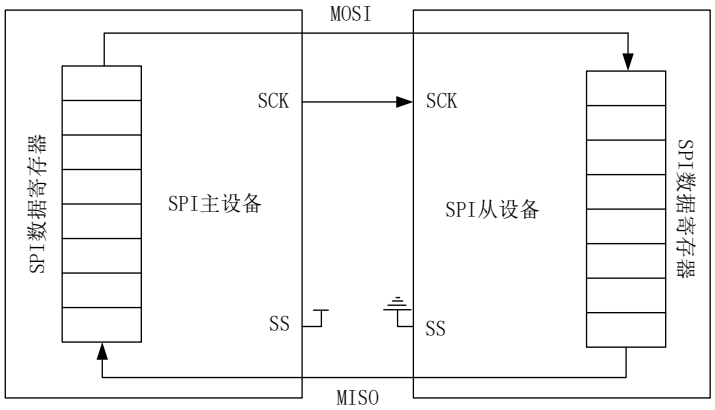


图6.9 SPI接口组成

资料整理
火龙果软件
荐

6.4.1 SPI的物理层

2.SPI互联

SPI互连主要有“一主一从”和“一主多从”两种方式。

（1）“一主一从”。在该方式的SPI互连下，有SPI的主/从设备各一个。

SPI从设备进行通信。分别将主/从设备的SCK、MOSI、MISO直接相连，将主设备的SS置1，从设备的SS置0，如图6.10所示。

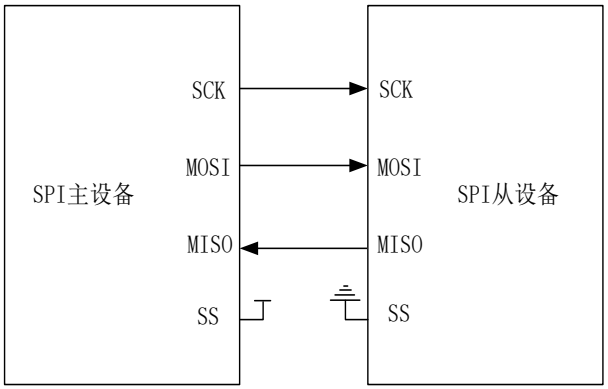


图6.10 “一主一从”的SPI互连

当SPI互连时，主设备和从设备的两根数据线直接相连，即主/从设备的MISO及MOSI相连。

资料整理 火龙果软件 荐

6.4.1 SPI的物理层

2.SPI互联

SPI互连主要有“一主一从”和“一主多从”两种方式。

（2）“一主多从”。在“一主多从”的SPI互连方式下，一个SPI主设备可以和多个从设备相互通信。所有的SPI设备（包括主设备和从设备）共享时钟线和数据线，即SCK、MOSI、MISO 三根线，并在主设备端使用多个GPIO引脚模拟多个SS引脚，以实现多个从设备的选择，如图6.11所示。在多个从设备的SPI互连方式下，片选信号SS对每个从设备分别选通，增加了连接的数量和难度。

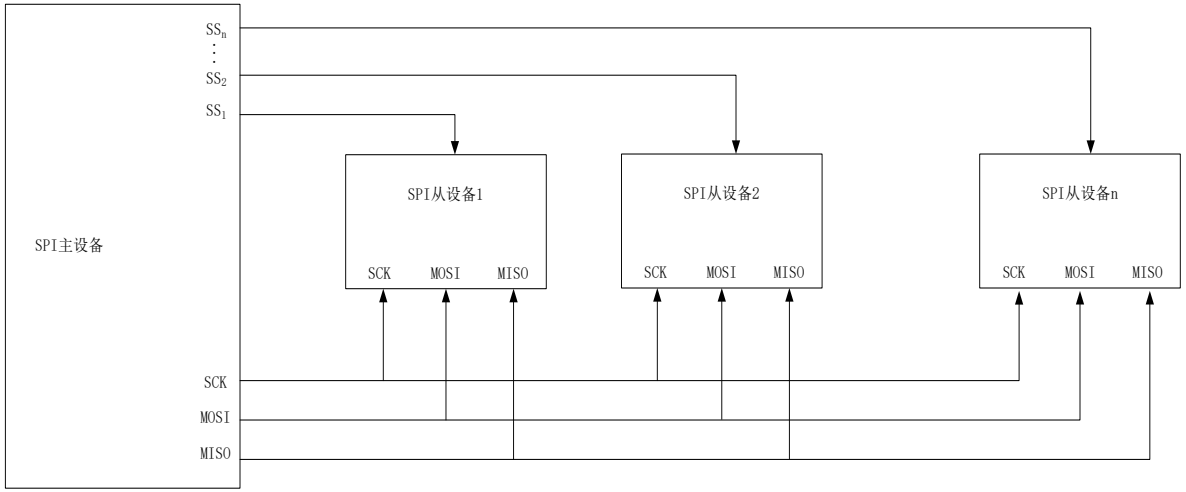


图6.10 “一主一从”的SPI互连

在多个从设备的SPI系统中，所有的SPI设备共享时钟线和数据线，同一时刻只能有一个从设备参与通信。当主设备与其中一个从设备进行通信时，其他从设备的时钟线和数据线保持高阻态，避免影响当前数据的传输。

资料整理 火龙果软件 荐

6.4.2 SPI协议层

SPI是全双工同步串行通信，它按照时钟线SCK的节拍在数据线MOSI和MISO上逐位数据传输。SCK每产生一个时钟脉冲，MOSI 和MISO就各自传输一位数据。经过若干个（由SPI数据格式指定）SCK脉冲，完成一个SPI数据帧的传输，SPI主/从设备的串行寄存器完成一次数据交换。

SPI的协议层包括SPI时序、SPI数据格式和SPI传输速率三个方面。

1.SPI时序

SPI时序与其时钟极性和时钟相位有关。

（1）时钟极性。时钟极性（CPOL）是指SPI通信设备空闲时SPI时钟线SCK的电平，认为SPI开始通信，即SPI片选线SS为0时SCK的电平。

- CPOL=0。当CPOL=0时，SCK 在空闲状态时为0。
- CPOL=1。当CPOL=1时，SCK在空闲状态时为1。

（2）时钟相位。时钟相位（CPHA）是指SPI数据接收方在SCK一个时钟周期中的那个跳边沿，从准备就绪的数据线上采集数据。

- CPHA=0。当CPHA=0时，数据线MOSI/MISO上的数据会在SCK时钟线的奇数（即第1、3...）跳边沿存取。

资料整理 火龙果软件 荐

6.4.2 SPI协议层

2.SPI数据格式

SPI数据传输以帧为单位，选择8位或16位数据帧格式。SPI数据由高位到低位（即MSB在前，LSB在后），也可以由低位到高位（即LSB在前，MSB在后）依次传输。

3.SPI传输速率

相比于USART及I2C方式，SPI具有较高的传输速率，时钟SCK高达几十兆赫。

例题6.11：利用SPI2进行初始化等操作。

#配置相关引脚的复用功能，使能SPIx时钟。调用函数：void GPIO_Init();

#初始化SPIx，设置SPIx工作模式。调用函数：void SPI_Init();

#使能SPIx。　　调用函数：void SPI_Cmd();

#SPI传输数据。调用函数：void SPI_I2S_SendData()；uint16_t SPI_I2S_ReceiveData();

#查看SPI传输状态。调用函数：SPI_I2S_GetFlagStatus(SPI2, SPI_I2S_FLAG_RXNE)。

下面按照这个一般步骤来进行一个简单的SPI程序：（接下页）

资料整理
火龙果软件
荐

下面按照这个一般步骤来进行一个简单的SPI程序：

```
void SPI2_Init(void)
{
   GPIO_InitTypeDef GPIO_InitStructure;
   SPI_InitTypeDef  SPI_InitStructure;

        RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE );   //PORTB时钟使能
        RCC_APB1PeriphClockCmd(RCC_APB1Periph_SPI2,  ENABLE );    //SPI2时钟使能

        GPIO_InitStructure.GPIO_Pin = GPIO_Pin_13 | GPIO_Pin_14 | GPIO_Pin_15;
        GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;        //PB13/14/15复用推挽输出
        GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
        GPIO_Init(GPIOB, &GPIO_InitStructure);                  //初始化GPIOB
        GPIO_SetBits(GPIOB,GPIO_Pin_13|GPIO_Pin_14|GPIO_Pin_15);   //PB13/14/15上拉
        SPI_InitStructure.SPI_Direction = SPI_Direction_2Lines_FullDuplex;   //设置SPI单向或者
双向的数据模式:SPI设置为双线双向全双工
        SPI_InitStructure.SPI_Mode = SPI_Mode_Master;       //设置SPI工作模式:设置为主SPI
        SPI_InitStructure.SPI_DataSize = SPI_DataSize_8b;    //设置SPI的数据大小:SPI发送接收
8位帧结构
        SPI_InitStructure.SPI_CPOL = SPI_CPOL_High;         //串行同步时钟的空闲状态为高电
平
```

资料整理 火龙果软件 荐

（接上页）

```
SPI_InitStructure.SPI_CPHA = SPI_CPHA_2Edge;      //串行同步时钟的第二个跳变沿（上升或下降）数据被采样
SPI_InitStructure.SPI_NSS = SPI_NSS_Soft;          //NSS信号由硬件（NSS管脚）还是软件（使用SSI位）管理:内部NSS信号有SSI位控制
SPI_InitStructure.SPI_BaudRatePrescaler = SPI_BaudRatePrescaler_256;  //定义波特率预分频的值:波特率预分频值为256
SPI_InitStructure.SPI_FirstBit = SPI_FirstBit_MSB;     //指定数据传输从MSB位还是LSB位开始:数据传输从MSB位开始
SPI_InitStructure.SPI_CRCPolynomial = 7;               //CRC值计算的多项式
SPI_Init(SPI2, &SPI_InitStructure);                    //根据SPI_InitStruct中指定的参数初始化外设SPIx寄存器
SPI_Cmd(SPI2, ENABLE);                                 //使能SPI外设
SPI2_ReadWriteByte(0xff);                              //启动传输
}

//SPI 速度设置函数
//SpeedSet:
//SPI_BaudRatePrescaler_2   2分频
//SPI_BaudRatePrescaler_8   8分频
//SPI_BaudRatePrescaler_16  16分频
//SPI_BaudRatePrescaler_256 256分频
```

资料整理
火龙果软件
荐

（接上页）

```
void SPI2_SetSpeed(u8 SPI_BaudRatePrescaler)
{
  assert_param(IS_SPI_BAUDRATE_PRESCALER(SPI_BaudRatePrescaler));
        SPI2->CR1&=0XFFC7;
        SPI2->CR1|=SPI_BaudRatePrescaler;        //设置SPI2速度
        SPI_Cmd(SPI2,ENABLE);
 }
 //SPIx 读写一个字节
//TxData:要写入的字节
//返回值:读取到的字节
u8 SPI2_ReadWriteByte(u8 TxData)
{
        u8 retry=0;
        while (SPI_I2S_GetFlagStatus(SPI2, SPI_I2S_FLAG_TXE) == RESET) //检查指定的SPI标志位设置与否:发送缓存空标志位
 {
                retry++;
                if(retry>200)return 0;
                }

        return SPI_I2S_ReceiveData(SPI2); //返回通过SPIx最近接收的数据

}
```

资料整理 火龙果软件 荐

例题6.12：SPI为单工通信有几种方式，请分别描述。

SPI模块能够以两种配置工作于单工方式：

- 1条时钟线和1条双向数据线；
- 1条时钟线和1条数据线（只接收或只发送）；

1条时钟线和1条双向数据线（BIDIMODE=1）

设置SPI_CR1寄存器中的BIDIMODE位而启用此模式。该模式下，SCK引脚作为时钟，主设备使用MOSI引脚而从设备使用MISO引脚作为数据通信。传输的方向由SPI_CR1寄存器里的BIDIOE控制，当这个位是1的时候，数据线是输出，否则是输入。

1条时钟和1条单向数据线（BIDIMODE=0）

该模式下，SPI模块可以作为只发送，或者作为只接收。

- 只发送模式类似于全双工模式（BIDIMODE=0，RXONLY=0）：数据在发送引脚（主模式时是MOSI、从模式时是MISO）上传输，而接收引脚（主模式时是MISO、从模式时是MOSI）可以作为通用的I/O使用。此时，软件不必理会接收缓冲器中的数据（如果读出数据寄存器，它不包含任何接收数据）。
- 配置并使能SPI模块为只接收模式的方式是：在只接收模式，设置SPI_CR2寄存器的RXONLY位关闭SPI的输出功能；此时，发送引脚（主模式时是MOSI、从模式时是MISO）被激活，可以作为其它功能使用。
- 在主模式时，一旦使能SPI，通信立即启动，当清除SPE位时立即停止当前的接收。在此模式下，不必读取BSY标志，在SPI通信期间这个标志始终为1。
- 在从模式时，拉低NSS（或在NSS软件模式时，SSI位为0）同时SCK有时钟脉冲，SPI就一直在接收。

资料整理 火龙果软件 荐

例题6.13：描述主模式下开始各模式传输流程。

- 全双工模式（BIDIMODE=0并且RXONLY=0）
  - 当写入数据到SPI_DR寄存器（发送缓冲器）后，传输开始；
  - 在传送第一位数据的同时，数据由并到串送到MOSI引脚上；
  - 同时，在MISO引脚上接收到的数据，按顺序由串到并传送到SPI_DR寄存器中。
- 单向的只接收模式（BIDIMODE=0并且RXONLY=1）
  - SPE=1时，传输开始；
  - 只有接收器被激活，在MISO引脚上接收到的数据，按顺序由串到并传送到SPI_DR寄存器（接收缓冲器）中。
- 双向模式，发送时（BIDIMODE=1并且BIDIOE=1）
  - 当写入数据到SPI_DR寄存器（发送缓冲器）后，传输开始；
  - 在传送第一位数据的同时，数据由并到串传送到MOSI引脚上；
  - 不接收数据。
- 双向模式，接收时（BIDIMODE=1并且BIDIOE=0）
  - SPE=1并且BIDIOE=0时，传输开始；
  - 在MOSI引脚接收到的数据，按顺序由串到并传送到SPI_DR寄存器（接收）中。
  - 不激活发送器，没有数据被串行地送到MOSI引脚上。

资料整理 火龙果软件 荐

本 章 小 结

本章对STM32单片机通信接口I2C 和SPI的特点及其应用方法做了详细阐述。主要包括以下几个方面：

1. I2C总线是一种简单、双向二线制同步串行总线。它只需要两根线即可在连接于总线上的器件之间传送信息。主器件用于启动总线传送数据，并产生时钟以开放传送的器件，此时任何被寻址的器件均被认为是从器件。在总线上主和从、发和收的关系不是恒定的，而取决于此时数据传送方向。

(1) 在硬件上，I2C总线只需要一根数据线和一根时钟线两根线。

(2) I2C总线是一个真正的多主机总线。

(3) I2C总线可以通过外部连线进行在线检测。

(4) 连接到相同总线上的IC数量只受总线最大电容的限制。

(5) 总线具有小的电流消耗，传输距离达到15m。

资料整理 火龙果软件 荐

2. STM32 I2C 的特点：多向控制总线、半双工，两个双向的数据线SDA和时钟线SCL、8位逐位传输数据和地址、主机或从机操作、主从模式接收与发送、传输结束中断标志等。

3. SPI 全双工同步串行总线，是微处理控制单元(MCU)和外围设备之间进行通信的同步串行端口。主要应用在EEPROM、Flash、实时时钟(RTC)、数模转换器(ADC)、网络控制器、MCU、数字信号处理器(DSP)以及数字信号解码器之间。SPI 系统可直接与各个厂家生产的多种标准外围器件直接对接，一般使用4 条线:串行时钟线SCK、主机输入/从机输出数据线MISO、主机输出/从机输入数据线MOSI 和低电平有效的从机片选SS。

资料整理
火龙果软件
荐

本章习题

1. I2C基本操作有哪些？I2C的特点是什么？
2. SPI基本操作有哪些？SPI的特点是什么？
3. I2C对I2C接口存储器件进行读/写有哪些相同和不同之处？
4. SPI对SPI接口存储器件进行读/写有哪些相同和不同之处？
5. I2C与SPI的使用相同和不同之处？
6. I2C与SPI分为哪几个主要部分？各起什么作用

资料整理
火龙果软件
荐

本章结束

资料整理
火龙果软件
荐