

### PY32F030\_003\_002A 系列 SPI 的 应用注意事项

#### 前言

PY32F030\_003\_002A 的串行外设接口(SPI)允许芯片与外部设备以半双工、全双工、单工同步的串行方式通信。此接口可以被配置成主模式，并为外部从设备提供通信时钟(SCK)。接口还能以从模式的方式工作。它可用于多种用途，包括使用一条双向数据线的双线单工同步传输。

本应用笔记将帮助用户了解 PY32F030\_003\_002A 的 SPI 模块的应用注意事项。

在本文档中，PY32 仅指表 1 中列出的产品系列。

表 1. 适用产品

类型	产品系列
微型控制器系列	PY32F030、PY32F003、PY32F002A

## 目录

<b>1</b>	<b>SPI 的最大频率</b>	<b>3</b>
1.1	注意事项	3
1.2	操作流程	3
1.3	代码示例	3
<b>2</b>	<b>SPI TX FIFO 操作方法</b>	<b>4</b>
2.1	注意事项	4
2.2	操作流程	4
2.3	代码示例	4
<b>3</b>	<b>SPI 作主机时, 如何使用半双工模式接收数据</b>	<b>5</b>
3.1	注意事项	5
3.2	操作流程	5
3.3	代码示例	5
<b>4</b>	<b>SPI 作从机且 SCK 为 PCLK/4 时, 如何配置从机</b>	<b>7</b>
4.1	注意事项	7
4.2	操作流程	7
4.3	代码示例	7
<b>5</b>	<b>SPI 作从机时, 对 SCK 的要求</b>	<b>8</b>
5.1	注意事项	8
<b>6</b>	<b>使用 PY32F030/003 SPI 分别作主、从进行互联通信</b>	<b>9</b>
6.1	注意事项	9
6.2	解决方案	9
<b>7</b>	<b>使用 SPI DMA 发送和接收</b>	<b>10</b>
7.1	注意事项	10
<b>8</b>	<b>版本历史</b>	<b>11</b>

# 1 SPI 的最大频率

## 1.1 注意事项

- 在使用 PY32 设备的 SPI 模块进行通信时，为了保证数据传输的可靠性，需要考虑 PCLK 和 SPI 时钟频率 SCK 的比例关系，表 1-1 给出了 SPI 在不同模式下时钟配置的建议。

表 1-1 SPI 不同模式下频率限制

模式	描述
Master	主模式下 SCK 频率最大为 PCLK/4
Slave	从模式下 SCK 频率最大为 PCLK/4

## 1.2 操作流程

- 定义 SPI\_HandleTypeDef 类型变量 SpiHandle，并初始化 SpiHandle 各个成员变量；
- 对于 SpiHandle 中的 BaudRatePrescaler 成员变量，无论 SPI 作主机还是从机，最大设置为 SPI\_BAUDRATEPRESCALER\_4 (4 分频)，具体根据实际应用场景来配置；
- 初始化 SPI 模块；
- 使用 SPI 进行通信；

## 1.3 代码示例

```

SPI_HandleTypeDef SpiHandle;
...
/* Set the SPI parameters */
SpiHandle.Instance = SPIx;
//设置 SPI SCK 的大小
SpiHandle.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_4;
SpiHandle.Init.Direction = SPI_DIRECTION_2LINES;
SpiHandle.Init.CLKPhase = SPI_PHASE_1EDGE;
SpiHandle.Init.CLKPolarity = SPI_POLARITY_LOW;
SpiHandle.Init.DataSize = SPI_DATASIZE_8BIT;
SpiHandle.Init.FirstBit = SPI_FIRSTBIT_MSB;
SpiHandle.Init.TIMode = SPI_TIMODE_DISABLE;
SpiHandle.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
SpiHandle.Init.CRCPolynomial = 7;
SpiHandle.Init.CRCLength = SPI_CRC_LENGTH_8BIT;
SpiHandle.Init.NSS = SPI_NSS_SOFT;
SpiHandle.Init.NSSPMode = SPI_NSS_PULSE_DISABLE;
SpiHandle.Init.Mode = SPI_MODE_MASTER;
if(HAL_SPI_Init(&SpiHandle) != HAL_OK)
{
    /* Initialization Error */
    Error_Handler();
}
    
```

## 2 SPI TX FIFO 操作方法

### 2.1 注意事项

- 在 SPI 模块的设计中，SPI TX FIFO 的内容一旦写入且还未发送出去，将不能清除；
  - 如果 TX FIFO 已满，继续写 SPI\_DR 会被视为无效操作，不会覆盖 TX FIFO 中的数据；
- 在实际应用中，如果用户想通覆盖 TX FIFO 中的数据，可通过以下 2.2 操作流程来实现。

### 2.2 操作流程

- 需要先复位 SPI 模块。以 SPI1 举例，先使能 RCC->RCC\_APB1RST2 中的 SPI1\_RST 位，然后置 0 来完成 SPI1 模块复位；
- 初始化 SPI 模块；
- 使用 SPI 进行通信

### 2.3 代码示例

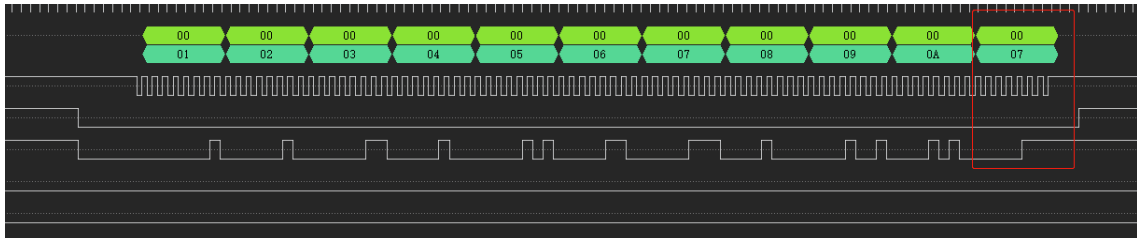
```
/*复位 SPI1*/
__RCC_SPI1_FORCE_RESET();
__RCC_SPI1_RELEASE_RESET();
/* Set the SPI parameters */
...
if(HAL_SPI_Init(&SpiHandle) != HAL_OK)
{
    /* Initialization Error */
    Error_Handler();
}
```

### 3 SPI 作主机时，如何使用半双工模式接收数据

#### 3.1 注意事项

- 半双工主机接收模式下，主机接收数据完成后，会多发一个帧长的 SCK 时钟。(对于 8 位数据格式多发 8 个 SCK 时钟，对于 16 位数据格式多发 16 个 SCK 时钟)，具体如下图 3-1:

图 3-1 半双工主机接收数据



- 上图是半双工主机接收模式下，接收 10 个字节的的数据波形，会发现数据接收完成后，主机继续发了 1 个 bytes 的 SCK；在实际应用中建议用户按照 3.2 操作流程来解决此问题。

#### 3.2 操作流程

方案一：使用全双工模式替代半双工模式

- 定义 SPI\_HandleTypeDef 类型变量 SpiHandle，并初始化 SpiHandle 各个成员变量；
- 将 SpiHandle 中的 Direction 成员变量设置为 SPI\_DIRECTION\_2LINES，使用全双工模式；
- 对于中断模式通信，使用 HAL\_SPI\_TransmitReceive 接口来代替 HAL\_SPI\_Receive 接口；DMA 模式和中断模式采用类似的方法；具体可参考 3.3 中的代码示例。

方案二：

- 用户接收数据时，多接收一个帧长的数据，然后将该数据丢弃  
如果用户预期接收 datalen 长度的数据，在此种方案下，需要接收 datalen+1 长度的数据，如下：  
HAL\_SPI\_Receive(&SpiHandle, (uint8\_t \*)RxBuff, datalen+1, 0xFFFFFFFF);
- 然后将接收到的 RxBuff 中的最后一帧数据丢弃，即可解决此问题。

#### 3.3 代码示例

以下是方案一的示例

```

SPI_HandleTypeDef SpiHandle;
...
/* Set the SPI parameters */
SpiHandle.Instance           = SPIx;
SpiHandle.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_4;
//全双工配置
SpiHandle.Init.Direction      = SPI_DIRECTION_2LINES;
SpiHandle.Init.CLKPhase       = SPI_PHASE_1EDGE;
SpiHandle.Init.CLKPolarity   = SPI_POLARITY_LOW;
SpiHandle.Init.DataSize      = SPI_DATASIZE_8BIT;
SpiHandle.Init.FirstBit      = SPI_FIRSTBIT_MSB;
SpiHandle.Init.TIMode        = SPI_TIMODE_DISABLE;
SpiHandle.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
SpiHandle.Init.CRCPolynomial = 7;
SpiHandle.Init.CRCLength     = SPI_CRC_LENGTH_8BIT;
    
```

```
SpiHandle.Init.NSS                = SPI_NSS_SOFT;
SpiHandle.Init.NSSPMODE          = SPI_NSS_PULSE_DISABLE;
SpiHandle.Init.Mode              = SPI_MODE_MASTER;
if(HAL_SPI_Init(&SpiHandle) != HAL_OK)
{
    /* Initialization Error */
    Error_Handler();
}
while(1)
{
    //全双工通信
    HAL_SPI_TransmitReceive(&SpiHandle,(uint8_t *)aTxBuffer,(uint8_t *)aRxBuffer,len,TIMEOUT );
    while(HAL_SPI_GetState(&SpiHandle) != HAL_SPI_STATE_READY);
}
```

## 4 SPI 作从机且 SCK 为 PCLK/4 时，如何配置从机

### 4.1 注意事项

- 需要将 SPI\_CR2 寄存器中的 SLVFM 位置起。该位使能 SPI 从机 fast speed mode，而当 SCK 的速度小于 PCLK/4 时，一定不能设定该寄存器位；

### 4.2 操作流程

- 定义 SPI\_HandleTypeDef 类型变量 SpiHandle，并初始化 SpiHandle 各个成员变量；
- 初始化 SPI 模块；
- 将 SPI\_CR2 寄存器中的 SLVFM 位置起；
- 使用 SPI 进行通信；

### 4.3 代码示例

```

SPI_HandleTypeDef SpiHandle;
...
/* Set the SPI parameters */
SpiHandle.Instance           = SPIx;
//设置 SPI SCK 的大小
SpiHandle.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_4;
SpiHandle.Init.Direction        = SPI_DIRECTION_2LINES;
SpiHandle.Init.CLKPhase         = SPI_PHASE_1EDGE;
SpiHandle.Init.CLKPolarity      = SPI_POLARITY_LOW;
SpiHandle.Init.DataSize         = SPI_DATASIZE_8BIT;
SpiHandle.Init.FirstBit         = SPI_FIRSTBIT_MSB;
SpiHandle.Init.TIMode           = SPI_TIMODE_DISABLE;
SpiHandle.Init.CRCCalculation   = SPI_CRCCALCULATION_DISABLE;
SpiHandle.Init.CRCPolynomial    = 7;
SpiHandle.Init.CRCLength        = SPI_CRC_LENGTH_8BIT;
SpiHandle.Init.NSS              = SPI_NSS_SOFT;
SpiHandle.Init.NSSPMode         = SPI_NSS_PULSE_DISABLE;
SpiHandle.Init.Mode             = SPI_MODE_SLAVE;
if(HAL_SPI_Init(&SpiHandle) != HAL_OK)
{
    /* Initialization Error */
    Error_Handler();
}

SPIx->SPI_CR2 |= SPI_CR2_SLVFM;// 使能从机 fast speed mode

```

## 5 SPI 作从机时，对 SCK 的要求

### 5.1 注意事项

- SPI 模块作为 Slave 模式时，对 SCK 的速度有一定限制，这是由于 SPI 内部设计导致。关于 SCK 的选择与配置方式，请参考下表 4-1:

表 4-1 SPI SLAVE 模式下 SCK 配置表

SCK 频率	配置要求
SCK ≤ 6M	波特率无需配置，如果 SCK 和 PCLK 配置满足 SCK=PCLK /4 时,SLVFM 需设置,否则不能设置 SLVFM 位
SCK = 8M	<ol style="list-style-type: none"> <li>1. HSI 选择 16M, PCLK 配置为 32M</li> <li>2. 波特率配置为 PCLK/4</li> <li>3. SLVFM 需置位</li> </ol>
SCK = 11.6M	<ol style="list-style-type: none"> <li>1. HSI 选择 22.12M, PCLK 配置为 44.24M</li> <li>2. 波特率配置为 PCLK/4</li> <li>3. SLVFM 需置位</li> </ol>
SCK = 12M	<ol style="list-style-type: none"> <li>1. HSI 选择 24M, PCLK 配置为 48M</li> <li>2. 波特率配置为 PCLK/4</li> <li>3. SLVFM 需置位</li> </ol>
其他频率	暂不支持

- SPI 模块作为 Slave 模式时，对 SCK 的占空比要求为 45%-55%。对于 Master 用 GPIO 模拟的 SPI 需要特别注意，不推荐用户使用 GPIO 模拟的 SPI 作为 Master;



## 6 使用 PY32F030/003 SPI 分别作主、从进行互联通信

### 6.1 注意事项

- 在 SCK 为 PCLK/4 条件下，同时使用 PY32F030\_003\_002A SPI 分别作主机和从机进行互联通信会失败。

### 6.2 解决方案

- 方案一：使用其他分频系数，将 SCK 调整为 PCLK/8 或者以下来进行通信
- 方案二：将主从机中的任意一个换成其他厂家的 SPI

PUYA CONFIDENTIAL

## 7 使用 SPI DMA 发送和接收

### 7.1 注意事项

- SPI 使用 DMA 方式实现数据的发送和接收，建议配置 DMA 接收的优先级高于 DMA 发送的优先级。默认 DMA 通道优先级按照通道顺序由高到低，既 CH1>CH2>CH3；

PUYA CONFIDENTIAL

## 8 版本历史

版本	日期	更新记录
V0.1	2021.10.15	初版
V1.0	2022.06.30	修改格式
V1.1	2022.08.16	增加 002A 内容
V1.2	2023.08.24	更新声明



Puya Semiconductor Co., Ltd.

### 声 明

普冉半导体(上海)股份有限公司 (以下简称: "Puya" ) 保留更改、纠正、增强、修改 Puya 产品和/或本文档的权利, 恕不另行通知。用户可在下单前获取产品的最新相关信息。

Puya 产品是依据订单时的销售条款和条件进行销售的。

用户对 Puya 产品的选择和使用承担全责, 同时若用于其自己或指定第三方产品上的, Puya 不提供服务支持且不对此类产品承担任何责任。

Puya 在此不授予任何知识产权的明示或暗示方式许可。

Puya 产品的转售, 若其条款与此处规定不一致, Puya 对此类产品的任何保修承诺无效。

任何带有 Puya 或 Puya 标识的图形或字样是普冉的商标。所有其他产品或服务名称均为其各自所有者的财产。

本文档中的信息取代并替换先前版本中的信息。

普冉半导体(上海)股份有限公司 - 保留所有权利