YC31xx SPI 应用说明

V1.0

Yichip Microelectronics

Revision History

Version	Date	Author	Description
V1.0	2020-2-20	Dengzhiqian	Initial version

Confidentiality Level:

confidential

目录

1. 文档说明

1.1 编写目的

为使用 SPI 相关 demo 及 demo 中相关 API 提供指南

1.2 适用范围

31xx 系列芯片

1.3 文件说明

SPI Demo 路径为 ModuleDemo\SPI
SPI 库文件为如下图 yc_spi.c 与 yc_spi.h,路径为 Librarier\sdk

2. 结构体说明

2.1. SPI 初始化结构体说明

结构体名称: SPI_InitTypeDef
说明: 目的保存 SPI 核心参数

元素名称	类型	说明	元素取值范围
Mode	uint8_t	Specifies the SPI operating mode. This parameter can be a value of @ref SPI_mode	SPI_Mode_Master SPI_Mode_Slave
CPOL	uint8_t	Specifies the serial clock steady state. This parameter can be a value of@ref SPI_Clock_Polarit	SPI_CPOL_Low SPI_CPOL_High
CPHA	uint8_t	Specifies the clock active edge for the bit capture. This parameter can be a value of @ref SPI_Clock_Phase	SPI_CPHA_First_Edge SPI_CPHA_Second_Edge
BaudRatePrescaler	uint8_t	Specifies the Baud Rate prescaler value which will be used to configure the transmit and receive SCK clock. This parameter can be a value of @ref SPI_BaudRate_Prescaler.@note The communication clock is derived from the master clock.	SPI_BaudRatePrescaler_1 SPI_BaudRatePrescaler_2 SPI_BaudRatePrescaler_4 SPI_BaudRatePrescaler_8 SPI_BaudRatePrescaler_16 SPI_BaudRatePrescaler_32

		The slave clock does not need to be set.	SPI_BaudRatePrescaler_64 SPI_BaudRatePrescaler_128
RW_Delay	uint8_t	Specifies the Delay time between send and receive data,the value must be 0 to 127	0~127

3. 函数说明

3.1. SPI_Init

函数原型：void SPI_Init(SPIx_TypeDef SPIx, SPI_InitTypeDef* SPI_InitStruct);
说明：　　SPI 初始化函数，目的是为 SPI 核心寄存器赋初值

参数	方向	说明
SPIx_TypeDef SPIx	IN	SPIx 的寄存器地址：3121 有两个 SPI 此参数的选择范围为：SPI0、SPI1
SPI_InitTypeDef* SPI_InitStruct	IN	参考 SPI 初始化结构体说明

返回值	说明
None	None

3.2. SPI_SendData

函数原型：void SPI_SendData(SPIx_TypeDef SPIx, uint8_t data)
说明：　　SPI 1 字节数据发送函数，目的是将数据写入 DMA。

参数	方向	说明
SPIx_TypeDef SPIx	IN	SPIx 的寄存器地址：3121 有两个 SPI 此参数的选择范围为：SPI0、SPI1
uint8_t data	IN	1 字节待发送数据

返回值	说明
None	None

3.3. SPI_SendBuff

函数原型：void SPI_SendBuff(SPIx_TypeDef SPIx, uint8_t *buff, int len)
说明：　　SPI 多字节数据发送函数，目的是将数据写入 DMA。

参数	方向	说明

SPIx_TypeDef SPIx	IN	SPIx 的寄存器地址：3121 有两个 SPI 此参数的选择范围为：SPI0、SPI1
uint8_t *buff	IN	待发送数据地址
int len	IN	待发送数据长度

返回值	说明
None	None

3.4. SPI_SendAndReceiveData

函数原型：void SPI_SendAndReceiveData(SPIx_TypeDef SPIx,
uint8_t *TxBuff,
uint16_t TxLen,
uint8_t *RxBuff,
uint16_t RxLen)

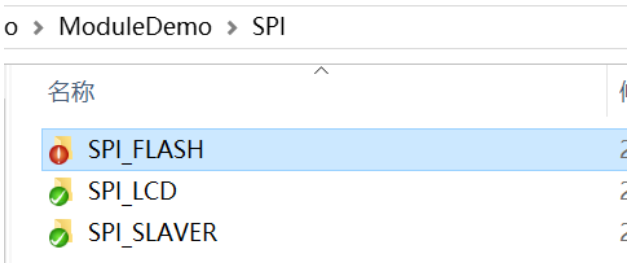
说明： SPI 多字节数据发送同时任意字节数据接收函数

参数	方向	说明
SPIx_TypeDef SPIx,	IN	SPIx 的寄存器地址：3121 有两个 SPI 此参数的选择范围为：SPI0、SPI1
uint8_t *TxBuff,	IN	待发送数据地址
uint16_t TxLen,	IN	待发送数据长度
uint8_t *RxBuff	OUT	待接收数据地址
uint16_t RxLen	OUT	待接收数据长度

返回值	说明
None	None

4. 示例代码及说明

示例代码存放在 ModuleDemo\SPI 目录下（如下图）



① SPI_FLASH: SPI 操作 W25Q16 flash 示例，包括读取 ID、擦除及读写。
② SPI_LCD ： SPI 操作 SHll06 lcd 示例，包括清屏、显示中文。

③ SPI_SLAVER : SPI0 做主机，SPI1 做从机进行数据传输示例。

4.1. SPI_FLASH 部分示例代码

```
int main(void)
{
    UART_Configuration(); // 串口初始化，配置说明参考 UART 应用说明文档
    SPI_Configuration(); // SPI 初始化

    /*以下操作的 GPIO，是对板子上其它 SPI_CS 进行操作，避免干扰*/
    GPIO_Config(GPIOC, GPIO_Pin_8, OUTPUT_HIGH);
    GPIO_Config(GPIOC, GPIO_Pin_6, OUTPUT_HIGH);
    GPIO_Config(GPIOB, GPIO_Pin_8, OUTPUT_HIGH);

    FlashID = Spi_W25Q16_ReadDeviceID(); // 获取设备 ID
    MyPrintf("FlashID = 0x%04x\n", FlashID);
    Spi_W25X_ChipErase(0x00); // 整片擦除
    Spi_W25Q16_PageWrite(0x00); // 写操作
    Spi_W25Q16_ReadData(0x00, 16, W25Q16_ReadBuff); // 读操作

    for (int i = 0; i < 15; i++)
        MyPrintf("%d\n", W25Q16_ReadBuff[i]);

    while (1)
    {
    }
}

void Spi_W25Q16_PageWrite(uint32_t Addr)
{

    uint8_t TxBuff[20]={W25X_PageProgram, Addr>>16, Addr>>8, Addr, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,
14, 15};
    W25Q16_WriteEnable(); // 写使能

    SPI_SendAndReceiveData(SPI0, TxBuff, 19, 0, 0);

    W25Q16_WaitBusyState(); // 等待忙信号
}
```

4.2. SPI_LCD 部分示例代码

```
int main(void)
```

```
{
    UART_Configuration(); //  串口初始化，配置说明参考 UART 应用说明文档
    LCD_Configuration(); // LCD 初始化

    MyPrintf("Yichip Yc3121 LCD Demo V1.0.\r\n");

    GPIO_Config(GPIOA, GPIO_Pin_2, OUTPUT_HIGH); //  开背光

    /*以下操作的 GPIO，是对板子上其它 SPI_CS 进行操作，避免干扰*/
    GPIO_Config(GPIOC, GPIO_Pin_8, OUTPUT_HIGH);
    GPIO_Config(GPIOC, GPIO_Pin_9, OUTPUT_HIGH);
    GPIO_Config(GPIOB, GPIO_Pin_8, OUTPUT_HIGH);

    SPI_LCD_Test();

    while (1)
    {

    }
}
```

4.3. SPI_SLAVE 部分示例代码

```
int main(void)
{
    UART_Configuration(); //  串口初始化，配置说明参考 UART 应用说明文档
    SPI_Configuration(); // SPI 初始化

    MyPrintf("Ychip 3121 SPI_SLAVER Demo !\n");

    /*以下操作的 GPIO，是对板子上其它 SPI_CS 进行操作，避免干扰*/
    GPIO_Config(GPIOC, GPIO_Pin_6, OUTPUT_HIGH);
    GPIO_Config(GPIOC, GPIO_Pin_8, OUTPUT_HIGH);
    GPIO_Config(GPIOC, GPIO_Pin_9, OUTPUT_HIGH);
    GPIO_Config(GPIOB, GPIO_Pin_8, OUTPUT_HIGH);

    M_txbuff[0] = 0xaa;
    S_txbuff[0] = 0x55;
    M_rxbuff[0] = 0;
    S_rxbuff[0] = 0;

    while (1)
    {
```

```
        if (UART_IsRXFIFONotEmpty(UART0)) //  判断 UART0 接收空间是否为空
        {
            Data = UART_ReceiveData(UART0);    // send 0xA1
        }
        SPI_Slaver_Test(); //  数据传输
    }
}
```