Application Note

SPC11X8/SPD11X8 SSP 单元使用指南

2019 年 7 月 – 版本 1

©2019 Spintrol Limited

目录

©2019 Spintrol Limited

表格列表

图片列表

©2019 Spintrol Limited

1 SSP 概述

SSP 单元是一个同步串行通信单元，常常被用于与多种外部设备进行通信，比如 ADC、音频解码器、通信解码器，是一种应用广泛的通信接口。

SPC11X8/SPD11X8 的 SSP 单元有以下特点：

- 支持 TI 的 SSP 通信协议
- 支持 Motorola 的 SPI 通信接口
- 最大速度 50M bps
- 支持 4~32 位数据位宽，高位左对齐，MSB 位先发送
- 支持主/从配置
- 带有 16x32 bit 或 32x16bit 的接收/发送 FIFO
- 支持 back-to-back 传输模式

在 SSP 的驱动库中，已经有下列驱动函数可供调动，可大大方便用户使用和理解。表 1-1 和表 1-2 中 SSPx 表示 SSP 模块编号，取值为 SSP。

表 1-1: SSP 宏定义列表

宏名	功能及参数说明
SSP_Enable(SSPx)	使能 SSPx
SSP_Disable(SSPx)	禁用 SSPx
SSP_EnableFIFOPackMode(SSPx)	使能 SSPx FIFO Packed 模式
SSP_DisableFIFOPackMode(SSPx)	禁用 SSPx FIFO Packed 模式
SSP_EnableRxOverflowInt (SSPx)	使能 SSPx 接收 FIFO 超上限中断
SSP_DisableRxOverflowInt (SSPx)	禁用 SSPx 接收 FIFO 超上限中断
SSP_EnableTxUnderflowInt (SSPx)	使能 SSPx 发送 FIFO 超下限中断
SSP_DisableTxUnderflowInt (SSPx)	禁用 SSPx 发送 FIFO 超下限中断
SSP_EnableRxTimeoutInt(SSPx)	使能 SSPx 接收超时中断
SSP_DisableRxTimeoutInt(SSPx)	禁用 SSPx 接收超时中断
SSP_EnableTxDataRequestInt (SSPx)	使能 SSPx 发送 FIFO 为空中断
SSP_DisableTxDataRequestInt (SSPx)	禁用 SSPx 发送 FIFO 为空中断
SSP_EnableRxDataAvailableInt (SSPx)	使能 SSPx 接收 FIFO 满中断
SSP_DisableRxDataAvailableInt (SSPx)	禁用 SSPx 接收 FIFO 满中断
SSP_GetTxFIFOLevel(SSPx)	获取 SSPx 发送 FIFO 中数据个数
SSP_GetRxFIFOLevel(SSPx)	获取 SSPx 接收 FIFO 中数据个数
SSP_SetRxFIFOTriggerThreshold (SSPx, u8RxLevel)	设置 SSPx 接收 FIFO 的触发深度 SSPx：SSP 模组基址指针 u32RxLevel: RxFIFO 深度大于等于该值，触发中断。
SSP_SetTxFIFOTriggerThreshold (SSPx, u8TxLevel)	设置 SSPx 发送 FIFO 的触发深度 SSPx：SSP 模组基址指针 u32TxLevel: TxFIFO 深度小于等于该值，触发中断。
SSP_IsTxNotFull (SSPx)	返回 SSPx 发送 FIFO 是否非满状态
SSP_IsRxNotEmpty (SSPx)	返回 SSPx 接收 FIFO 是否非空状态

©2019 Spintrol Limited

宏名	功能及参数说明
SSP_IsBusy(SSPx)	返回 SSPx 是否正忙状态
SSP_IsTxServiceRequest (SSPx)	返回 SSPx 是否发生发送 FIFO 事件状态
SSP_IsRxServiceRequest (SSPx)	返回 SSPx 是否发生接收 FIFO 事件状态
SSP_IsRxOverflow (SSPx)	返回 SSPx 接收 FIFO 是否溢出状态
SSP_IsRxTimeoutPending(SSPx)	返回 SSPx 是否出现接收超时状态
SSP_IsTxUnderflow (SSPx)	返回 SSPx 发送 FIFO 是否下溢状态
SSP_IsBusySyncSlaveClock(SSPx)	返回 SSPx 是否正在同步时钟状态
SSP_IsTxHasOddSample (SSPx)	返回 SSPx 发送 FIFO 中是否有奇数个数据状态
SSP_IsRxHasOddSample (SSPx)	返回 SSPx 接收 FIFO 中是否有奇数个数据状态
SSP_ClearRxOverflowInt (SSPx)	清除 SSPx 接收 FIFO 溢出中断位
SSP_ClearTxUnderflowInt (SSPx)	清除 SSPx 发送 FIFO 下溢中断位
SSP_ClearRxTimeoutInt(SSPx)	清除 SSPx 接收 FIFO 超时中断位
SSP_SetRxTimeout(SSPx,u32Timeout)	设置 SSPx 接收超时时长为 u32Timeout
SSP_SendData(SSPx,u32Data)	SSPx 发送数据 u32Data
SSP_ReceiveData(SSPx)	返回 SSPx 接收数据

表 1-2: SSP 函数列表

函数名	功能及参数说明
void SSP_Init(SSP_REGS* SSPx, uint8_t u8MasterOrSlave, uint8_t u8DataSize, uint32_t u32Baudrate, SSP_FramePolarityEnum u1FramePolarity, SSP_ClockIdleLevelEnum u1ClockIdleLevel, SSP_SendDataEdgeEnum u1DataSendEdge)	初始化 SSP SSPx：SSP 模组基址指针 u8MasterOrSlave: 主机，1； 从机，0 u8DataSize: 数据位宽（1~32） u32Baudrate: 通信速率 u1FramePolarity: SFRM 信号低/高有效 u1ClockIdleLevel: SCLK 空闲时是低/高电平 u1DataSendEdge: SCLK 上升/下降沿发送数据
void SSP_InitEasy(SSP_REGS* SSPx, uint8_t u8MasterOrSlave, uint8_t u8DataSize, uint32_t u32Baudrate)	初始化 SSP，简化版 SSPx：SSP 模组基址指针 u8MasterOrSlave: 主机，1； 从机，0 u8DataSize: 数据位宽（1~32） u32Baudrate: 通信速率 其他假定条件为： • SCLK 上升沿发送数据 • SFRM 低有效 • SCLK 空闲状态为低
void SSP_Send(SSP_REGS* SSPx, uint32_t *pu32Buf, uint32_t u32Count)	发送一组数据 SSPx：SSP 模组基址指针 pu32Buf：发送数组指针 u32Count：发送数据数量
uint32_t SSP_Receive(SSP_REGS* SSPx, uint32_t *pu32Buf, uint32_t u32Count)	接收一组数据，返回接收数据计数 SSPx：SSP 模组基址指针 pu32Buf：接收数组指针 u32Count：接收数据数量

©2019 Spintrol Limited

函数名	功能及参数说明
uint32_t SSP_MasterTransceive(SSP_REGS* SSPx, uint32_t *pu32WriteBuffer, uint32_t *pu32ReadBuffer, uint32_t u32Count)	SSP 主机模式全双工收发收据，返回接收数据计数 SSPx：SSP 模组基址指针 pu32WriteBuffer：发送数组指针 pu32ReadBuferf：接收数组指针 u32Count：收发数据数量
uint32_t SSP_MasterB2BTransceive(SSP_REGS* SSPx, uint32_t *pu32WriteBuffer, uint32_t *pu32ReadBuffer, uint32_t u32Count)	SSP 主机 back-to-back 模式全双工收发收据，返回接收数据计数，最高速率 4Mbps（HCLK 200MHz，FIFO Packed 模式关闭） SSPx：SSP 模组基址指针 pu32WriteBuffer：发送数组指针 pu32ReadBuferf：接收数组指针 u32Count：收发数据数量
uint32_t SSP_SlaveTransceive(SSP_REGS* SSPx, uint32_t *pu32WriteBuffer, uint32_t *pu32ReadBuffer, uint32_t u32Count)	SSP 从机模式全双工收发收据，返回接收数据计数。 SSPx：SSP 模组基址指针 pu32WriteBuffe：发送数组指针 pu32ReadBuferf：接收数组指针 u32Count：收发数据数量

©2019 Spintrol Limited

2 SSP 各种模式实例

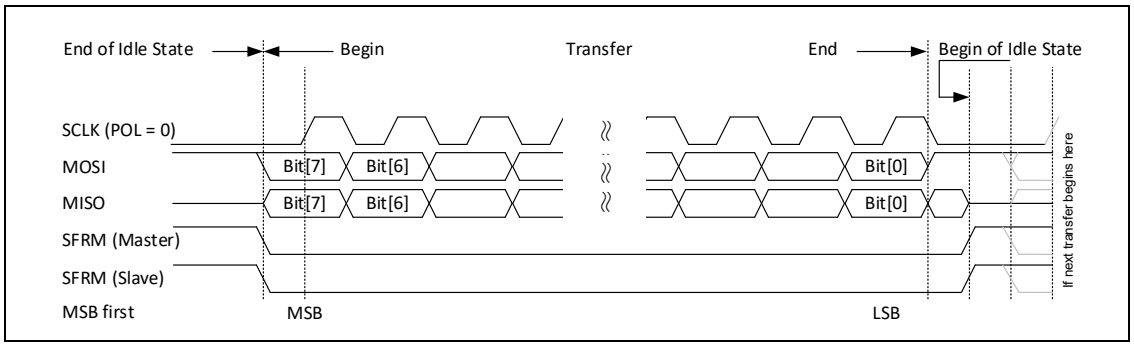
2.1 SSP 主模式和从模式收发实例

在这个例子里，会演示如何使用 SSP 的基本功能，收发数据。
该实例需要两颗 SPC11X8/SPD11X8 芯片，分别设置 SSP 为主机模式和从机模式，其余配置相同。
数据帧配置如下：

- 波特率 3Mbps
- 数据宽度 8 bits
- SFRM 低电平有效（CS 信号）
- 总线空闲时 SCLK 为低电平
- SCLK 上升沿采样数据
- FIFO Packed 模式关闭

如果涉及 SSP 的简单应用，用户可以依照这个例子进行设定，波形如下所示：

图 2–1: SSP 波形格式 (POL = 0, PHS = 0)



主机和从机全局变量定义，代码如下：

Example Code

```
#define T_BUFFER_SIZE 128
uint32_t gu32BuffSize = T_BUFFER_SIZE;
uint32_t gau32TxBuf[T_BUFFER_SIZE];
uint32_t gau32RxBuf[T_BUFFER_SIZE];
uint32_t gau32Buf[T_BUFFER_SIZE];
```


©2019 Spintrol Limited

主机和从机都需要配置系统时钟 HCLK 为 200MHz，并且打开串口，代码如下：

Example Code

```
FLASH_WALLOW();
FLASH_SetTiming(200000000);
/* Disable flash write access after flash operation had done */
FLASH_WDIS();
CLOCK_InitWithRCO(CLOCK_HCLK_200MHZ);
Delay_Init();

/*Init the UART*/
GPIO_SetPinChannel(GPIO_34,GPIO34_UART_TXD);
GPIO_SetPinChannel(GPIO_35,GPIO35_UART_RXD);
CLOCK_EnableModule(UART_MODULE);
UART_Init(UART,38400);
```

SSP 主机配置代码如下：

Example Code

```
// Init SSP Pinmux
GPIO_SetPinChannel(GPIO_36, GPIO36_BIT_MUXSEL_SPI_SCLK);
GPIO_SetPinChannel(GPIO_37, GPIO37_BIT_MUXSEL_SPI_SFRM);
GPIO_SetPinChannel(GPIO_38, GPIO38_BIT_MUXSEL_SPI_MOSI);
GPIO_SetPinChannel(GPIO_39, GPIO39_BIT_MUXSEL_SPI_MISO);

// Set Output Strength, for high speed
GPIO_SetOutStrength(GPIO_36, GPIO_OUT_STRENGTH_20MA);
GPIO_SetOutStrength(GPIO_37, GPIO_OUT_STRENGTH_20MA);
GPIO_SetOutStrength(GPIO_38, GPIO_OUT_STRENGTH_20MA);
GPIO_SetOutStrength(GPIO_39, GPIO_OUT_STRENGTH_20MA);

/*
 * Init SSP Baud Rate, Frame Format
 * 1) Set master mode
 * 2) Set frame data 8-bits width
 * 3) Set transfer speed 3Mbps
 * 4) Set pin SFRM active low
 * 5) Set SCLK Polarity: SCLK is low when bus idle
 * 6) Set SCLK Phase: sample data on SCLK rising edge
 */
SSP_Init(SSP, SSP_MASTER_MODE, DataWidth_8bit, SSPBaudrate_3M,
        SSP_FRM_ACTIVE_LOW,
        SSP_CLK_IDLE_LOW,SSP_SEND_ON_FALLING_EDGE);

/* Disable FIFO Packed Mode */
SSP_DisableFIFOPackMode(SSP);
```

©2019 Spintrol Limited

```
/*
 * We need delay some time to wait for GPIOs ready,
 * if not, there may be problem if the master and slave
 * is not the same Chip. eg. Master is SPC2168, Slave is SPC1068.
 */
Delay_Ms(500);

/* Enable SSP */
SSP_Enable(SSP);
```


©2019 Spintrol Limited

SSP 从机配置代码如下：

Example Code

```
// Init SSP Pinmux
GPIO_SetPinChannel(GPIO_36, GPIO36_BIT_MUXSEL_SPI_SCLK);
GPIO_SetPinChannel(GPIO_37, GPIO37_BIT_MUXSEL_SPI_SFRM);
GPIO_SetPinChannel(GPIO_38, GPIO38_BIT_MUXSEL_SPI_MOSI);
GPIO_SetPinChannel(GPIO_39, GPIO39_BIT_MUXSEL_SPI_MISO);

// Set Output Strength, for high speed
GPIO_SetOutStrength(GPIO_36, GPIO_OUT_STRENGTH_20MA);
GPIO_SetOutStrength(GPIO_37, GPIO_OUT_STRENGTH_20MA);
GPIO_SetOutStrength(GPIO_38, GPIO_OUT_STRENGTH_20MA);
GPIO_SetOutStrength(GPIO_39, GPIO_OUT_STRENGTH_20MA);

// Init SSP Baud Rate, Frame Format
// 1) Set slave mode
// 2) Set frame data 8-bits width
// 3) Set transfer speed 3Mbps, but don't care this under slave mode
// 4) Set pin SFRM active low
// 5) Set SCLK Polarity: SCLK is low when bus idle
// 6) Set SCLK Phase: sample data on SCLK rising edge
SSP_Init(SSP, SSP_SLAVE_MODE, 8, 3000000, SSP_FRM_ACTIVE_LOW,
         SSP_CLK_IDLE_LOW, SSP_SEND_ON_FALLING_EDGE);

// Disable FIFO Packed Mode
SSP_DisableFIFOPackMode(SSP);

// Enable SSP
SSP_Enable(SSP);
```

©2019 Spintrol Limited

SSP 主机收发实例代码如下：

Example Code

```
int i;

while(1)
{
    // Prepare data to be send
    for(i=0; i < (int)(gu32BuffSize); i++)
        gau32TxBuf[i] = rand() & 0xFF;

    printf("Master Tx data...\n");
    Delay_ms(4000); // Wait Slave Prepare data
    SSP_MasterTransceive(SSP, gau32TxBuf, gau32Buf, gu32BuffSize);

    printf("Master Rx data...\n");
    Delay_ms(4000); // Wait Slave Prepare data
    SSP_MasterTransceive(SSP, gau32Buf, gau32RxBuf, gu32BuffSize);

    for(i=0; i < (int)(gu32BuffSize); i++)
    {
        if( gau32TxBuf[i] != gau32RxBuf[i])
        {
            printf("[Error]@%4d: %3d: TX(%p,0x%02X) != RX(%p,0x%02X)\n",
                    __LINE__, i, &gau32TxBuf[i], gau32TxBuf[i],
                    &gau32RxBuf[i], gau32RxBuf[i] );
        }
        else
        {
            printf("%3d: TX(%p,0x%02X) == RX(%p,0x%02X)\n", i,
                     &gau32TxBuf[i], gau32TxBuf[i], &gau32RxBuf[i],
                     gau32RxBuf[i] );
        }
    }
}
```


©2019 Spintrol Limited

SSP 从机收发实例代码如下：

Example Code

```
int i;

while(1)
{
    // Prepare data to be send
    for(i=0; i < (int)(gu32BuffSize); i++)
        gau32TxBuf[i] = rand() & 0xFF;

    printf("Slave Rx data...\n");
    SSP_SlaveTransceive(SSP, gau32TxBuf, gau32Buf, gu32BuffSize);

    printf("Slave Tx data...\n");
    SSP_SlaveTransceive(SSP, gau32Buf, gau32RxBuf, gu32BuffSize);

    for(i=0; i < (int)(gu32BuffSize); i++)
    {
        if( gau32TxBuf[i] != gau32RxBuf[i])
        {
            printf("[Error]@%4d: %3d: TX(%p,0x%02X) != RX(%p,0x%02X)\n",
                    __LINE__, i, &gau32TxBuf[i], gau32TxBuf[i],
                    &gau32RxBuf[i], gau32RxBuf[i] );
        }
        else
        {
            printf("%3d: TX(%p,0x%02X) == RX(%p,0x%02X)\n", i,
                    &gau32TxBuf[i], gau32TxBuf[i], &gau32RxBuf[i],
                    gau32RxBuf[i] );
        }
    }
}
```

程序下载后，俩颗芯片同时运行:

1） 串口波特率设定为 38400bps，可以看到发送数据和接收数据一致，则代码运行正常。

2） 通过示波器观察波形与图一致。

©2019 Spintrol Limited

2.2 SSP back-to-back 主模式和从模式收发实例

SSP back-to-back 模式是连续发送多个数据帧，并且保持 SFRAM 始终有效的一种传输方式。运行主模式的 SSP，发送数据期间，TXFIFO 只要非空，那么 SFRM 就会保持有效状态；反之，TXFIFO 清空，则 SFRM 恢复空闲状态。因此，以下两个条件都会开启 SSP back-to-back 模式：

1） SFRAM 有效期间 TXFIFO 应当非空，这意味着要尽可能快将需要发送的数据写入 TXFIFO；

2） 当数据位宽小于等于 16bits，FIFO Packed 模式打开时候，写一次 TXFIFO 会发送两帧数据，触发 back-to-back 模式。

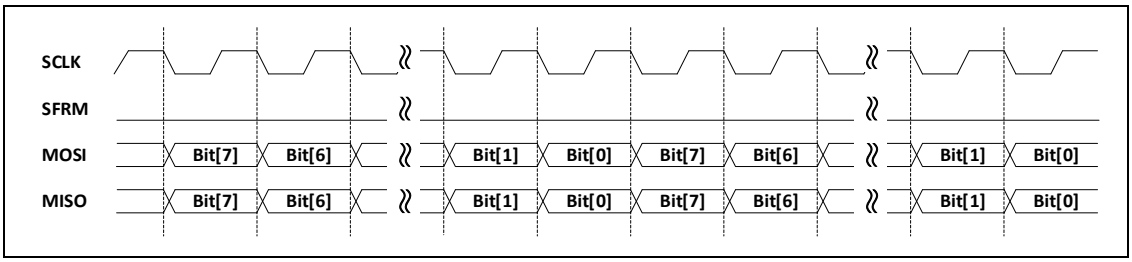
注意：驱动库中函数“SSP_MasterB2BTransceive()”支持 back-to-back 模式最高速率 4Mbps（HCLK 200MHz，FIFO Packed 模式关闭）。这是因为发送速率太快，MCU 写 TXFIFO 较慢，会导致 TXFIFO 清空，SFRM 恢复空闲。

本小节例子，将会演示 SSP 使用 back-to-back 模式收发数据。该实例需要两颗 SPC11X8/SPD11X8 芯片，分别设置 SSP 为主机模式和从机模式，其余配置相同。数据帧配置如下：

- 波特率 1Mbps
- 数据宽度 8 bits
- SFRM 低电平有效（CS 信号）
- 总线空闲时 SCLK 为低电平
- SCLK 上升沿采样数据
- FIFO Packed 模式关闭

波形如下所示：

图 2–2: SSP back-to-back 波形格式(POL = 0, PHS = 0)



主机和从机全局变量定义，代码如下：

Example Code

```
#define T_BUFFER_SIZE 128
uint32_t gu32BuffSize = T_BUFFER_SIZE;
uint32_t gau32TxBuf[T_BUFFER_SIZE];
uint32_t gau32RxBuf[T_BUFFER_SIZE];
uint32_t gau32Buf[T_BUFFER_SIZE];
```


©2019 Spintrol Limited

主机和从机都需要配置系统时钟 HCLK 为 200MHz，并且打开串口，代码如下：

Example Code

```
FLASH_WALLOW();
FLASH_SetTiming(200000000);
/* Disable flash write access after flash operation had done */
FLASH_WDIS();
CLOCK_InitWithRCO(CLOCK_HCLK_200MHZ);
Delay_Init();

/* Init the UART */
GPIO_SetPinChannel(GPIO_34,GPIO34_UART_TXD);
GPIO_SetPinChannel(GPIO_35,GPIO35_UART_RXD);
CLOCK_EnableModule(UART_MODULE);
UART_Init(UART,38400);
```

SSP 主机配置代码如下：

Example Code

```
// Init SSP Pinmux
GPIO_SetPinChannel(GPIO_36, GPIO36_BIT_MUXSEL_SPI_SCLK);
GPIO_SetPinChannel(GPIO_37, GPIO37_BIT_MUXSEL_SPI_SFRM);
GPIO_SetPinChannel(GPIO_38, GPIO38_BIT_MUXSEL_SPI_MOSI);
GPIO_SetPinChannel(GPIO_39, GPIO39_BIT_MUXSEL_SPI_MISO);

// Set Output Strength, for high speed
GPIO_SetOutStrength(GPIO_36, GPIO_OUT_STRENGTH_20MA);
GPIO_SetOutStrength(GPIO_37, GPIO_OUT_STRENGTH_20MA);
GPIO_SetOutStrength(GPIO_38, GPIO_OUT_STRENGTH_20MA);
GPIO_SetOutStrength(GPIO_39, GPIO_OUT_STRENGTH_20MA);

// Init SSP Baud Rate, Frame Format
// 1) Set master mode
// 2) Set frame data 8-bits width
// 3) Set transfer speed 1Mbps
// 4) Set pin SFRM active low
// 5) Set SCLK Polarity: SCLK is low when bus idle
// 6) Set SCLK Phase: sample data on SCLK rising edge
SSP_Init(SSP, SSP_MASTER_MODE, 8, 1000000, SSP_FRM_ACTIVE_LOW,
         SSP_CLK_IDLE_LOW, SSP_SEND_ON_FALLING_EDGE);

// Disable FIFO Packed Mode
SSP_DisableFIFOPackMode(SSP);

/*
 * We need delay some time to wait for GPIOs ready,
```

©2019 Spintrol Limited

```
 * if not, there may be problem if the master and slave
 * is not the same Chip. eg. Master is SPC2168, Slave is SPC1068.
 */
Delay_Ms(500);

// Enable SSP
SSP_Enable(SSP);
```


©2019 Spintrol Limited

SSP 从机配置代码如下：

Example Code

```
// Init SSP Pinmux
GPIO_SetPinChannel(GPIO_36, GPIO36_BIT_MUXSEL_SPI_SCLK);
GPIO_SetPinChannel(GPIO_37, GPIO37_BIT_MUXSEL_SPI_SFRM);
GPIO_SetPinChannel(GPIO_38, GPIO38_BIT_MUXSEL_SPI_MOSI);
GPIO_SetPinChannel(GPIO_39, GPIO39_BIT_MUXSEL_SPI_MISO);

// Set Output Strength, for high speed
GPIO_SetOutStrength(GPIO_36, GPIO_OUT_STRENGTH_20MA);
GPIO_SetOutStrength(GPIO_37, GPIO_OUT_STRENGTH_20MA);
GPIO_SetOutStrength(GPIO_38, GPIO_OUT_STRENGTH_20MA);
GPIO_SetOutStrength(GPIO_39, GPIO_OUT_STRENGTH_20MA);

// Init SSP Baud Rate, Frame Format
// 1) Set slave mode
// 2) Set frame data 8-bits width
// 3) Set transfer speed 1Mbps, but don't care this under slave mode
// 4) Set pin SFRM active low
// 5) Set SCLK Polarity: SCLK is low when bus idle
// 6) Set SCLK Phase: sample data on SCLK rising edge
SSP_Init(SSP, SSP_SLAVE_MODE, 8, 1000000, SSP_FRM_ACTIVE_LOW,
         SSP_CLK_IDLE_LOW, SSP_SEND_ON_FALLING_EDGE);

// Disable FIFO Packed Mode
SSP_DisableFIFOPackMode(SSP);

// Enable SSP
SSP_Enable(SSP);
```

©2019 Spintrol Limited

主机收发实例代码如下：

Example Code

```
int i;
uint32_t u32FrameSize;
uint32_t u32Tmp;

while(1)
{
    // Must full of buff size
    u32FrameSize = 1 << ( rand() % 5 );
    printf("Total %d bits druing SFRM active\n", u32FrameSize *
( (SSP->SSPCTL0.bit.SIZESEL + 1 ) * ( SSP->SSPCTL0.bit.ESIZESEL + 1 ) ) );
    u32Tmp = gu32BuffSize / u32FrameSize;

    // Prepare data to be send
    for(i=0; i < (int)(gu32BuffSize); i++)
        gau32TxBuf[i] = rand() & 0xFF;

    printf("Master Tx data...\n");
    Delay_ms(4000); // Wait Slave Prepare data
    for(i=0; i < (int)(u32Tmp); i++)
    {
        SSP_MasterB2BTransceive(SSP, gpu32TxBuf+u32FrameSize*i,
                        gpu32Buf+u32FrameSize*i, u32FrameSize);
    }

    printf("Master Rx data...\n");
    Delay_ms(4000); // Wait Slave Prepare data
    for(i=0; i < (int)(u32Tmp); i++)
    {
        SSP_MasterB2BTransceive(SSP, gpu32Buf+u32FrameSize*i,
                         gpu32RxBuf+u32FrameSize*i, u32FrameSize);
    }

    for(i=0; i < (int)(gu32BuffSize); i++)
    {
        if( gau32TxBuf[i] != gau32RxBuf[i])
        {
            printf("[Error]@%4d: %3d: TX(%p,0x%02X) != RX(%p,0x%02X)\n",
                    __LINE__, i, &gau32TxBuf[i], gau32TxBuf[i],
                    &gau32RxBuf[i], gau32RxBuf[i] );
        }
        else
        {
```


©2019 Spintrol Limited

```
            printf("%3d: TX(%p,0x%02X) == RX(%p,0x%02X)\n", i,
                    &gau32TxBuf[i], gau32TxBuf[i], &gau32RxBuf[i],
                    gau32RxBuf[i] );
        }
    }
}
```

SSP 从机收发实例代码如下：

Example Code

```
int i;

while(1)
{
    // Prepare data to be send
    for(i=0; i < (int)(gu32BuffSize); i++)
        gau32TxBuf[i] = rand() & 0xFF;

    printf("Slave Rx data...\n");
    SSP_SlaveTransceive(SSP, gau32TxBuf, gau32Buf, gu32BuffSize);

    printf("Slave Tx data...\n");
    SSP_SlaveTransceive(SSP, gau32Buf, gau32RxBuf, gu32BuffSize);

    for(i=0; i < (int)(gu32BuffSize); i++)
    {
        if( gau32TxBuf[i] != gau32RxBuf[i])
        {
            printf("[Error]@%4d: %3d: TX(%p,0x%02X) != RX(%p,0x%02X)\n",
                    __LINE__, i, &gau32TxBuf[i], gau32TxBuf[i],
                    &gau32RxBuf[i], gau32RxBuf[i] );
        }
        else
        {
            printf("%3d: TX(%p,0x%02X) == RX(%p,0x%02X)\n", i,
                    &gau32TxBuf[i], gau32TxBuf[i], &gau32RxBuf[i],
                    gau32RxBuf[i] );
        }
    }
}
```

程序下载后，俩颗芯片同时运行:

1） 串口波特率设定为 38400bps，可以看到发送数据和接收数据一致，则代码运行正常。

2） 通过示波器观察波形与图一致。

©2019 Spintrol Limited

2.3 SSP 发送中断实例

在这个例子里，会演示如何使用 SSP 的中断功能收发数据。
该实例需要两颗 SPC11X8/SPD11X8 芯片，分别设置 SSP 为主机模式和从机模式，其余配置相同。
数据帧配置如下：

- 波特率 3Mbps
- 数据宽度 8 bits
- SFRM 低电平有效（CS 信号）
- 总线空闲时 SCLK 为低电平
- SCLK 上升沿采样数据
- FIFO Packed 模式关闭

主机和从机全局变量定义，代码如下：

Example Code

```
#define T_BUFFER_SIZE 128
uint32_t gu32BuffSize = T_BUFFER_SIZE;
uint32_t gau32TxBuf[T_BUFFER_SIZE];
uint32_t gau32RxBuf[T_BUFFER_SIZE];
uint32_t gau32Buf[T_BUFFER_SIZE];

uint32_t gu32_cnt_ssp_tx_isr = 0;
uint32_t gu32_cnt_ssp_rx_isr = 0;
```

主机和从机都需要配置系统时钟 HCLK 为 200MHz，并且打开串口，代码如下：

Example Code

```
FLASH_WALLOW();
FLASH_SetTiming(200000000);
/* Disable flash write access after flash operation had done */
FLASH_WDIS();

CLOCK_InitWithRCO(CLOCK_HCLK_200MHZ);

Delay_Init();

/* Init the UART */
GPIO_SetPinChannel(GPIO_34,GPIO34_UART_TXD);
GPIO_SetPinChannel(GPIO_35,GPIO35_UART_RXD);
CLOCK_EnableModule(UART_MODULE);
UART_Init(UART,38400);
```

©2019 Spintrol Limited

SSP 主机配置代码如下：

Example Code

```
// Init SSP Pinmux
GPIO_SetPinChannel(GPIO_36, GPIO36_BIT_MUXSEL_SPI_SCLK);
GPIO_SetPinChannel(GPIO_37, GPIO37_BIT_MUXSEL_SPI_SFRM);
GPIO_SetPinChannel(GPIO_38, GPIO38_BIT_MUXSEL_SPI_MOSI);
GPIO_SetPinChannel(GPIO_39, GPIO39_BIT_MUXSEL_SPI_MISO);

// Set Output Strength, for high speed
GPIO_SetOutStrength(GPIO_36, GPIO_OUT_STRENGTH_20MA);
GPIO_SetOutStrength(GPIO_37, GPIO_OUT_STRENGTH_20MA);
GPIO_SetOutStrength(GPIO_38, GPIO_OUT_STRENGTH_20MA);
GPIO_SetOutStrength(GPIO_39, GPIO_OUT_STRENGTH_20MA);

/*
 * Init SSP Baud Rate, Frame Format
 * 1) Set master mode
 * 2) Set frame data 8-bits width
 * 3) Set transfer speed 3Mbps
 * 4) Set pin SFRM active low
 * 5) Set SCLK Polarity: SCLK is low when bus idle
 * 6) Set SCLK Phase: sample data on SCLK rising edge
 */
SSP_Init(SSP, SSP_MASTER_MODE, DataWidth_8bit, SSPBaudrate_3M,
         SSP_FRM_ACTIVE_LOW,
         SSP_CLK_IDLE_LOW,SSP_SEND_ON_FALLING_EDGE);

/* Disable FIFO Packed Mode */
SSP_DisableFIFOPackMode(SSP);

/* Set TxFIFO Threshold */
SSP_SetTxFIFOTriggerThreshold(SSP, 0);

/* Disable TxFIFO Empty Interrupt */
SSP_DisableTxDataRequestInt(SSP);

/* Enable SSP */
SSP_Enable(SSP);

/* Enable CM4 Interrupt Request */
NVIC_EnableIRQ(SSP_IRQn);
```


©2019 Spintrol Limited

SSP 从机配置代码如下：

Example Code

```
// Init SSP Pinmux
GPIO_SetPinChannel(GPIO_36, GPIO36_BIT_MUXSEL_SPI_SCLK);
GPIO_SetPinChannel(GPIO_37, GPIO37_BIT_MUXSEL_SPI_SFRM);
GPIO_SetPinChannel(GPIO_38, GPIO38_BIT_MUXSEL_SPI_MOSI);
GPIO_SetPinChannel(GPIO_39, GPIO39_BIT_MUXSEL_SPI_MISO);

// Set Output Strength, for high speed
GPIO_SetOutStrength(GPIO_36, GPIO_OUT_STRENGTH_20MA);
GPIO_SetOutStrength(GPIO_37, GPIO_OUT_STRENGTH_20MA);
GPIO_SetOutStrength(GPIO_38, GPIO_OUT_STRENGTH_20MA);
GPIO_SetOutStrength(GPIO_39, GPIO_OUT_STRENGTH_20MA);

/*
 * Init SSP Baud Rate, Frame Format
 * 1) Set master mode
 * 2) Set frame data 8-bits width
 * 3) Set transfer speed 3Mbps
 * 4) Set pin SFRM active low
 * 5) Set SCLK Polarity: SCLK is low when bus idle
 * 6) Set SCLK Phase: sample data on SCLK rising edge
 */
SSP_Init(SSP, SSP_SLAVE_MODE, DataWidth_8bit, SSPBaudrate_3M,
SSP_FRM_ACTIVE_LOW, SSP_CLK_IDLE_LOW,SSP_SEND_ON_FALLING_EDGE);

/* Disable FIFO Packed Mode */
SSP_DisableFIFOPackMode(SSP);

/* Enable SSP */
SSP_Enable(SSP);
```

©2019 Spintrol Limited

SSP 主机收发实例代码如下：

Example Code

```
int i;

while(1)
{
    gu32_cnt_ssp_tx_isr = 0;
    gu32_cnt_ssp_rx_isr = 0;

    // Prepare data to be send
    for(i=0; i < (int)(gu32BuffSize); i++)
        gau32TxBuf[i] = rand() & 0xFF;

    printf("Master Tx data...\n");
    Delay_ms(4000); // Wait Slave Prepare data
    // Enable TxFIFO Empty Interrupt
    SSP_EnableTxDataRequestInt(SSP);
    // Wait transmit data finish
    while(gu32_cnt_ssp_tx_isr < gu32BuffSize) {}
    // Wait receive data finish
    while(gu32_cnt_ssp_rx_isr < gu32BuffSize) {}

    printf("Master Rx data...\n");
    Delay_ms(4000); // Wait Slave Prepare data
    // Enable TxFIFO Empty Interrupt
    SSP_EnableTxDataRequestInt (SSP);
    // Wait transmit data finish
    while(gu32_cnt_ssp_tx_isr < ( gu32BuffSize * 2 ) ) {}
    // Wait receive data finish
    while(gu32_cnt_ssp_rx_isr < ( gu32BuffSize * 2 ) ) {}

    for(i=0; i < (int)(gu32BuffSize); i++)
    {
        if( gau32TxBuf[i] != gau32RxBuf[i])
        {
            printf("[Error]@%4d: %3d: TX(%p,0x%02X) != RX(%p,0x%02X)\n",
                    __LINE__, i, &gau32TxBuf[i], gau32TxBuf[i],
                    &gau32RxBuf[i], gau32RxBuf[i] );
        }
        else
        {
            printf("%3d: TX(%p,0x%02X) == RX(%p,0x%02X)\n", i,
                    &gau32TxBuf[i], gau32TxBuf[i], &gau32RxBuf[i],
                    gau32RxBuf[i] );
```


©2019 Spintrol Limited

```
            }
        }
}
```

©2019 Spintrol Limited

SSP 主机中断函数代码如下：

Example Code

```
// First send data to slave
if( gu32_cnt_ssp_tx_isr < u32BuffSize )
{
    SSP_MasterTransceive(SSP, gpu32TxBuf, gpu32Buf, u32BuffSize);

    gu32_cnt_ssp_tx_isr += u32BuffSize;
    gu32_cnt_ssp_rx_isr += u32BuffSize;

    // Disable TxFIFO Empty Interrupt
    SSP_DisableTxDataRequestInt (SSP);
}

// Second receive data from slave
else if( gu32_cnt_ssp_tx_isr < ( u32BuffSize * 2 ) )
{
    SSP_MasterTransceive(SSP, gpu32Buf, gpu32RxBuf, u32BuffSize);

    gu32_cnt_ssp_tx_isr += u32BuffSize;
    gu32_cnt_ssp_rx_isr += u32BuffSize;

    // Disable TxFIFO Empty Interrupt
    SSP_DisableTxDataRequestInt (SSP);
}
```


©2019 Spintrol Limited

SSP 从机收发实例代码如下：

Example Code

```
int i;

while(1)
{
    // Prepare data to be send
    for(i=0; i < (int)(gu32BuffSize); i++)
        gau32TxBuf[i] = rand() & 0xFF;

    printf("Slave Rx data...\n");
    SSP_SlaveTransceive(SSP, gau32TxBuf, gau32Buf, gu32BuffSize);

    printf("Slave Tx data...\n");
    SSP_SlaveTransceive(SSP, gau32Buf, gau32RxBuf, gu32BuffSize);

    for(i=0; i < (int)(gu32BuffSize); i++)
    {
        if( gau32TxBuf[i] != gau32RxBuf[i])
        {
            printf("[Error]@%4d: %3d: TX(%p,0x%02X) != RX(%p,0x%02X)\n",
                    __LINE__, i, &gau32TxBuf[i], gau32TxBuf[i],
                    &gau32RxBuf[i], gau32RxBuf[i] );
        }
        else
        {
            printf("%3d: TX(%p,0x%02X) == RX(%p,0x%02X)\n", i,
                    &gau32TxBuf[i], gau32TxBuf[i], &gau32RxBuf[i],
                    gau32RxBuf[i] );
        }
    }
}
```

程序下载后，俩颗芯片同时运行:

1） 串口波特率设定为 38400bps，可以看到发送数据和接收数据一致，则代码运行正常。

2） 通过示波器观察波形与图一致。

©2019 Spintrol Limited

3 修订记录

表 3-1: 文档修订记录

日期	版本	修改内容
2019-07-25	1	初始版本

©2019 Spintrol Limited