Spi双方通信测试。

Spi配置为片选低电平有效，在时钟第一个上升沿获取数据，在第一个下降沿改变数据。SOUT空闲时候呈现高电平，SIN呈现低电平。

轮询法：

主机轮询：

1、首先配置SIN，SOUT，SCK，PCS以及使能对应时钟

CLOCK\_EnableClock(kCLOCK\_PortD);

PORT\_SetPinMux(PORTD, PIN0\_IDX, kPORT\_MuxAlt2);/\* PORTD0 (pin 93) SPI0\_PCS0 \*/

PORT\_SetPinMux(PORTD, PIN1\_IDX, kPORT\_MuxAlt2); /\* PORTD1 (pin 94) SPI0\_SCK \*/

PORT\_SetPinMux(PORTD, PIN2\_IDX, kPORT\_MuxAlt2); /\* PORTD2 (pin 95) SPI0\_SOUT \*/

PORT\_SetPinMux(PORTD, PIN3\_IDX, kPORT\_MuxAlt2); /\* PORTD3 (pin 96) SPI0\_SIN \*/

2、配置主机模式属性

使用CTAR0，波特率为500kHz，每帧有8位，时钟极性为高电平有效，时钟相位为第一个边沿。传输高位在前，片选到时钟延迟时间2000ns，最后的时钟到片选延迟时间2000ns，传输间延迟时间2000ns，pcs0作为片选引脚，片选有效信号为低电平，禁用连续时钟，禁用接收fifo覆写，禁用改变时间格式，采样点为第一个边沿采样。

masterConfig.whichCtar= kDSPI\_Ctar0;

masterConfig.ctarConfig.baudRate= TRANSFER\_BAUDRATE;

masterConfig.ctarConfig.bitsPerFrame= 8U;

masterConfig.ctarConfig.cpol= kDSPI\_ClockPolarityActiveHigh;

masterConfig.ctarConfig.cpha= kDSPI\_ClockPhaseFirstEdge;

masterConfig.ctarConfig.direction= kDSPI\_MsbFirst;

masterConfig.ctarConfig.pcsToSckDelayInNanoSec=2000

masterConfig.ctarConfig.lastSckToPcsDelayInNanoSec=2000

masterConfig.ctarConfig.betweenTransferDelayInNanoSec=2000

masterConfig.whichPcs = EXAMPLE\_DSPI\_MASTER\_PCS\_FOR\_INIT;

masterConfig.pcsActiveHighOrLow = kDSPI\_PcsActiveLow;

masterConfig.enableContinuousSCK = false;

masterConfig.enableRxFifoOverWrite = false;

masterConfig.enableModifiedTimingFormat = false;

masterConfig.samplePoint = kDSPI\_SckToSin0Clock;

3、主机初始化，

DSPI\_MasterInit(EXAMPLE\_DSPI\_MASTER\_BASEADDR, &masterConfig, srcClock\_Hz);

4发送数据配置：

将发送数据指针传递，设置传输字节大小。

masterXfer.txData = masterTxData;

masterXfer.rxData = NULL;

masterXfer.dataSize = TRANSFER\_SIZE;

masterXfer.configFlags = kDSPI\_MasterCtar0 | EXAMPLE\_DSPI\_MASTER\_PCS\_FOR\_TRANSFER | kDSPI\_MasterPcsContinuous;

5、发送数据

DSPI\_MasterTransferBlocking(EXAMPLE\_DSPI\_MASTER\_BASEADDR, &masterXfer);

从机中断：

1. 设置从机传输属性，从机只能使用CTAR0，每帧8位，时钟极性有效电平位高电平，时钟相位是第一个边沿。禁用连续时钟，禁用接收fifo覆写，禁用可变时间格式，采样点位第一个边沿。

slaveConfig.whichCtar = kDSPI\_Ctar0;

slaveConfig.ctarConfig.bitsPerFrame = 8U;

slaveConfig.ctarConfig.cpol = kDSPI\_ClockPolarityActiveHigh;

slaveConfig.ctarConfig.cpha = kDSPI\_ClockPhaseFirstEdge;

slaveConfig.enableContinuousSCK = false;

slaveConfig.enableRxFifoOverWrite = false;

slaveConfig.enableModifiedTimingFormat = false;

slaveConfig.samplePoint = kDSPI\_SckToSin0Clock;

1. 从机初始化DSPI\_SlaveInit(EXAMPLE\_DSPI\_SLAVE\_BASEADDR, &slaveConfig);
2. 设置从机传输句柄，定义从机回调函数

DSPI\_SlaveTransferCreateHandle(EXAMPLE\_DSPI\_SLAVE\_BASEADDR, &g\_s\_handle, DSPI\_SlaveUserCallback, NULL);

1. 设置传输配置（接收），传递接收地址，数据大小

slaveXfer.txData = NULL;

slaveXfer.rxData = slaveRxData;

slaveXfer.dataSize = TRANSFER\_SIZE;

slaveXfer.configFlags = kDSPI\_SlaveCtar0;

DSPI\_SlaveTransferNonBlocking(EXAMPLE\_DSPI\_SLAVE\_BASEADDR,&g\_s\_handle,&slaveXfer);接收数据。

EDMA：

主机EDMA:

1、初始化DMAMUX，使能SPI0接收和发送通道

DMAMUX\_Init(EXAMPLE\_DSPI\_MASTER\_DMA\_MUX\_BASEADDR);

DMAMUX\_SetSource(EXAMPLE\_DSPI\_MASTER\_DMA\_MUX\_BASEADDR, masterRxChannel,

(uint8\_t)EXAMPLE\_DSPI\_MASTER\_DMA\_RX\_REQUEST\_SOURCE);

DMAMUX\_EnableChannel(EXAMPLE\_DSPI\_MASTER\_DMA\_MUX\_BASEADDR, masterRxChannel);

DMAMUX\_SetSource(EXAMPLE\_DSPI\_MASTER\_DMA\_MUX\_BASEADDR,

(uint8\_t)EXAMPLE\_DSPI\_MASTER\_DMA\_TX\_REQUEST\_SOURCE);

DMAMUX\_EnableChannel(EXAMPLE\_DSPI\_MASTER\_DMA\_MUX\_BASEADDR, masterTxChannel);

2、EDMA初始化

EDMA\_GetDefaultConfig(&userConfig);

EDMA\_Init(EXAMPLE\_DSPI\_MASTER\_DMA\_BASEADDR, &userConfig);

3、配置SPI，初始化。使用CTAR0，波特率为500kHz，每帧有8位，时钟极性为高电平有效，时钟相位为第一个边沿。传输高位在前，片选到时钟延迟时间2000ns，最后的时钟到片选延迟时间2000ns，传输间延迟时间2000ns，pcs0作为片选引脚，片选有效信号为低电平，禁用连续时钟，禁用接收fifo覆写，禁用改变时间格式，采样点为第一个边沿采样。

masterConfig.whichCtar = kDSPI\_Ctar0;

masterConfig.ctarConfig.baudRate = TRANSFER\_BAUDRATE;

masterConfig.ctarConfig.bitsPerFrame = 8;

masterConfig.ctarConfig.cpol = kDSPI\_ClockPolarityActiveHigh;

masterConfig.ctarConfig.cpha = kDSPI\_ClockPhaseFirstEdge;

masterConfig.ctarConfig.direction = kDSPI\_MsbFirst;

masterConfig.ctarConfig.pcsToSckDelayInNanoSec= 1000000000U / TRANSFER\_BAUDRATE;

masterConfig.ctarConfig.lastSckToPcsDelayInNanoSec=1000000000U/TRANSFER\_BAUDRATE;

masterConfig.ctarConfig.betweenTransferDelayInNanoSec=1000000000U/TRANSFER\_BAUDRATE;

masterConfig.whichPcs = EXAMPLE\_DSPI\_MASTER\_PCS\_FOR\_INIT;

masterConfig.pcsActiveHighOrLow = kDSPI\_PcsActiveLow;

masterConfig.enableContinuousSCK = false;

masterConfig.enableRxFifoOverWrite = false;

masterConfig.enableModifiedTimingFormat = false;

masterConfig.samplePoint = kDSPI\_SckToSin0Clock;

srcClock\_Hz = DSPI\_MASTER\_CLK\_FREQ;

DSPI\_MasterInit(EXAMPLE\_DSPI\_MASTER\_BASEADDR, &masterConfig, srcClock\_Hz);

1. EDMA创建接收，发送，中间媒介句柄，初始化句柄，传入回调函数。

EDMA\_CreateHandle(&(dspiEdmaMasterRxRegToRxDataHandle), EXAMPLE\_DSPI\_MASTER\_DMA\_BASEADDR, masterRxChannel);

EDMA\_CreateHandle(&(dspiEdmaMasterTxDataToIntermediaryHandle),

EXAMPLE\_DSPI\_MASTER\_DMA\_BASEADDR,masterIntermediaryChannel);

EDMA\_CreateHandle(&(dspiEdmaMasterIntermediaryToTxRegHandle), EXAMPLE\_DSPI\_MASTER\_DMA\_BASEADDR, masterTxChannel);

DSPI\_MasterTransferCreateHandleEDMA(EXAMPLE\_DSPI\_MASTER\_BASEADDR, &g\_dspi\_edma\_m\_handle,DSPI\_MasterUserCallback,NULL, &dspiEdmaMasterRxRegToRxDataHandle,

&dspiEdmaMasterTxDataToIntermediaryHandle,

&dspiEdmaMasterIntermediaryToTxRegHandle);

6、发送前，配置发送属性

masterXfer.txData = masterTxData;

masterXfer.rxData = NULL;

masterXfer.dataSize = TRANSFER\_SIZE;

masterXfer.configFlags = kDSPI\_MasterCtar0 | EXAMPLE\_DSPI\_MASTER\_PCS\_FOR\_TRANSFER | kDSPI\_MasterPcsContinuous;

DSPI\_MasterTransferEDMA(EXAMPLE\_DSPI\_MASTER\_BASEADDR, &g\_dspi\_edma\_m\_handle, &masterXfer)

1. 接收前，配置接收属性

isTransferCompleted = false;

masterXfer.txData = NULL;

masterXfer.rxData = masterRxData;

masterXfer.dataSize = TRANSFER\_SIZE;

masterXfer.configFlags = kDSPI\_MasterCtar0 | EXAMPLE\_DSPI\_MASTER\_PCS\_FOR\_TRANSFER | kDSPI\_MasterPcsContinuous;

DSPI\_MasterTransferEDMA(EXAMPLE\_DSPI\_MASTER\_BASEADDR, &g\_dspi\_edma\_m\_handle, &masterXfer)

从机EDMA：

1. 初始化DMAMUX，使能SPI0接收和发送通道

DMAMUX\_Init(EXAMPLE\_DSPI\_SLAVE\_DMA\_MUX\_BASEADDR);

DMAMUX\_SetSource(EXAMPLE\_DSPI\_SLAVE\_DMA\_MUX\_BASEADDR, slaveRxChannel,

(uint8\_t)EXAMPLE\_DSPI\_SLAVE\_DMA\_RX\_REQUEST\_SOURCE);

DMAMUX\_EnableChannel(EXAMPLE\_DSPI\_SLAVE\_DMA\_MUX\_BASEADDR, slaveRxChannel);

DMAMUX\_SetSource(EXAMPLE\_DSPI\_SLAVE\_DMA\_MUX\_BASEADDR, slaveTxChannel,

(uint8\_t)EXAMPLE\_DSPI\_SLAVE\_DMA\_TX\_REQUEST\_SOURCE);

DMAMUX\_EnableChannel(EXAMPLE\_DSPI\_SLAVE\_DMA\_MUX\_BASEADDR, slaveTxChannel);

2、初始化EDMA

EDMA\_GetDefaultConfig(&userConfig);

EDMA\_Init(EXAMPLE\_DSPI\_SLAVE\_DMA\_BASEADDR, &userConfig);

1. 配置从机SPI属性，初始化从机。设置从机传输属性，从机只能使用CTAR0，每帧8位时钟极性有效电平位高电平，时钟相位是第一个边沿。禁用连续时钟，禁用接收fifo覆写，禁用可变时间格式，采样点位第一个边沿。

slaveConfig.whichCtar = kDSPI\_Ctar0;

slaveConfig.ctarConfig.bitsPerFrame = 8U;

slaveConfig.ctarConfig.cpol = kDSPI\_ClockPolarityActiveHigh;

slaveConfig.ctarConfig.cpha = kDSPI\_ClockPhaseFirstEdge;

slaveConfig.enableContinuousSCK = false;

slaveConfig.enableRxFifoOverWrite = false;

slaveConfig.enableModifiedTimingFormat = false;

slaveConfig.samplePoint = kDSPI\_SckToSin0Clock;

DSPI\_SlaveInit(EXAMPLE\_DSPI\_SLAVE\_BASEADDR, &slaveConfig);

1. 创建EDMA收发句柄，初始化句柄

EDMA\_CreateHandle(&(dspiEdmaSlaveRxRegToRxDataHandle), EXAMPLE\_DSPI\_SLAVE\_DMA\_BASEADDR, slaveRxChannel);

EDMA\_CreateHandle(&(dspiEdmaSlaveTxDataToTxRegHandle), EXAMPLE\_DSPI\_SLAVE\_DMA\_BASEADDR, slaveTxChannel);

DSPI\_SlaveTransferCreateHandleEDMA(EXAMPLE\_DSPI\_SLAVE\_BASEADDR, &g\_dspi\_edma\_s\_handle, DSPI\_SlaveUserCallback, NULL,

&dspiEdmaSlaveRxRegToRxDataHandle, &dspiEdmaSlaveTxDataToTxRegHandle);

1. 配置接收属性，调用接收函数

slaveXfer.txData = NULL;

slaveXfer.rxData = slaveRxData;

slaveXfer.dataSize = TRANSFER\_SIZE;

slaveXfer.configFlags = kDSPI\_SlaveCtar0;

DSPI\_SlaveTransferEDMA(EXAMPLE\_DSPI\_SLAVE\_BASEADDR, &g\_dspi\_edma\_s\_handle, &slaveXfer))

1. 配置发送属性，调用发送函数

slaveXfer.txData = slaveRxData;

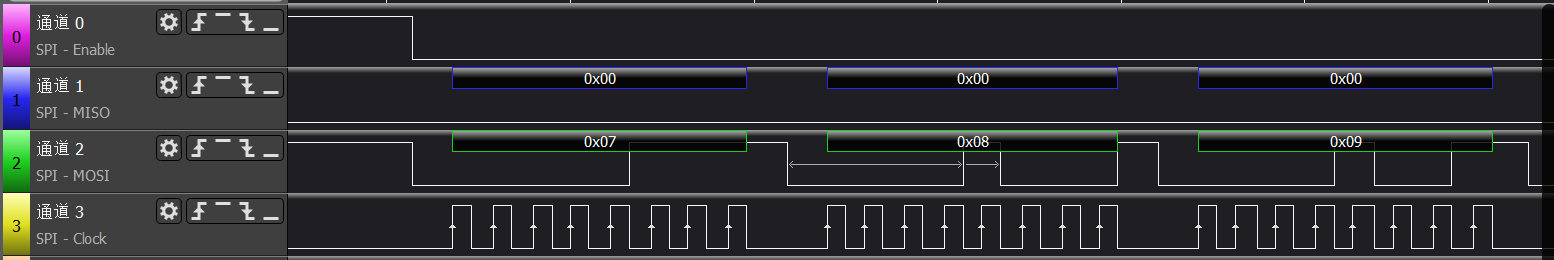
slaveXfer.rxData = NULL;

slaveXfer.dataSize = TRANSFER\_SIZE;

slaveXfer.configFlags = kDSPI\_SlaveCtar0;

DSPI\_SlaveTransferEDMA(EXAMPLE\_DSPI\_SLAVE\_BASEADDR, &g\_dspi\_edma\_s\_handle, &slaveXfer))

主机发送数据：



从机将收到数据发送

