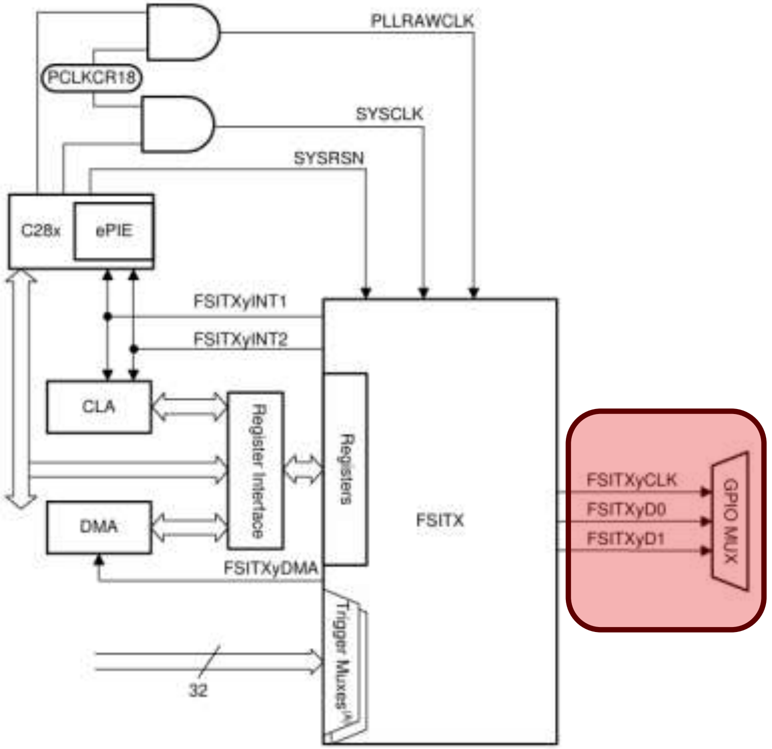
F28P55x编程实例Labs-FSI

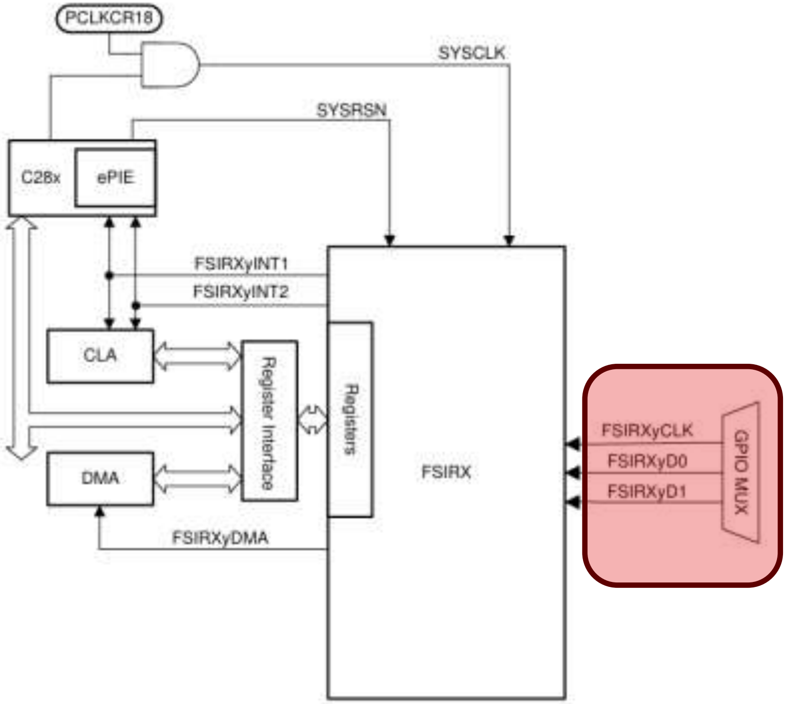
- Code Composer Studio
- C2000Ware
- LaunchXL-F28P55x

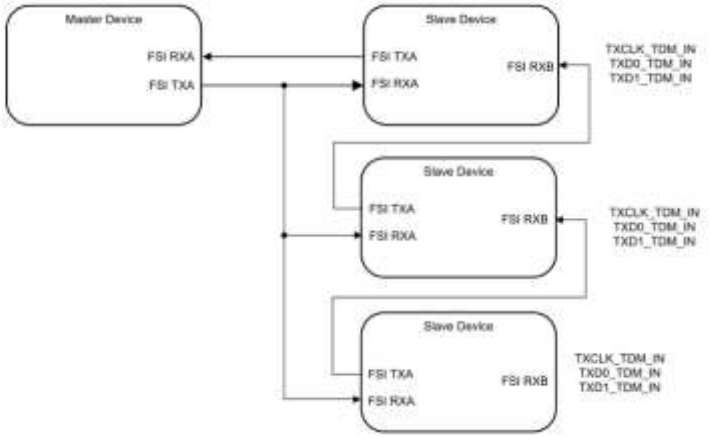
FSI

Fast Serial Interface

- ➢ 通信速率最高200Mbps
- ➢ 支持菊花链/星型多机连接拓扑
- ➢ 具有独立接收端/发送端







类星型多机通信拓扑

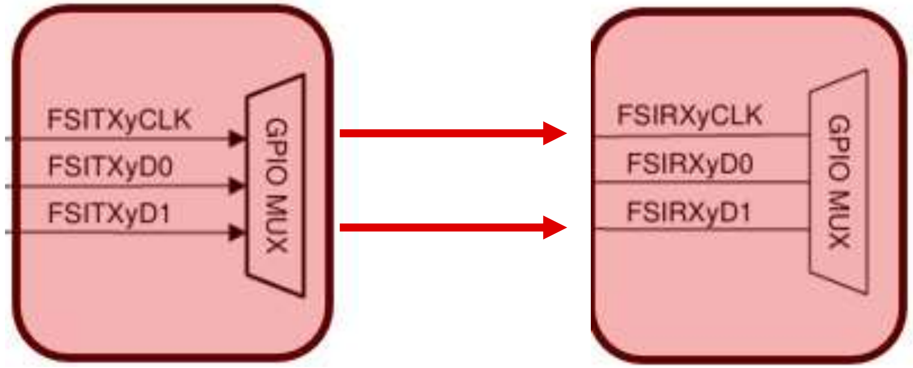
FSI

功能实现

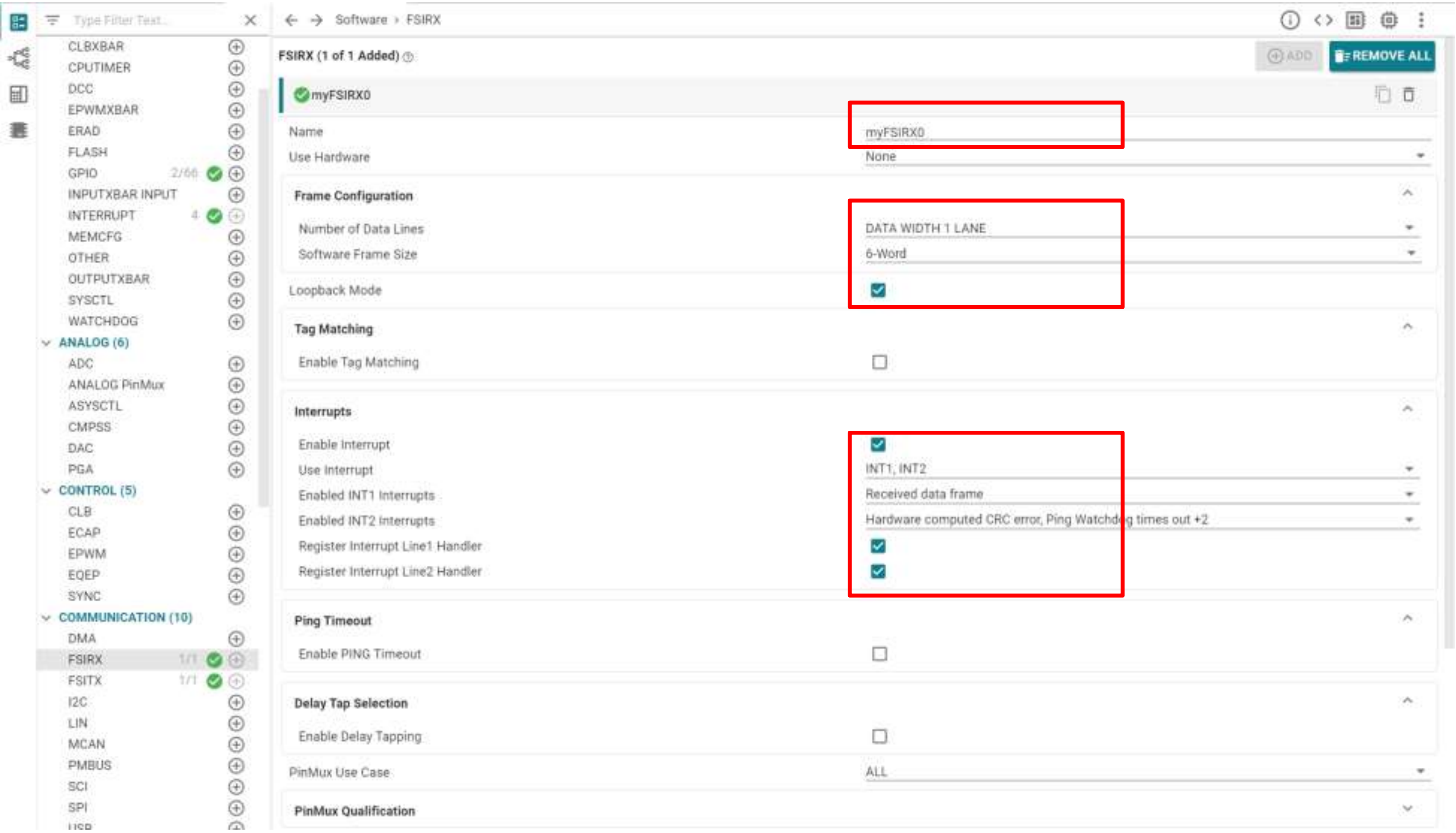
用Sysconfig配置FSI，实现接收模块FSIRX和发送模块FSITX之间的回环(loop-back)数据通信

实现步骤

- ➢ 复制空白工程
- ➢ Sysconfig配置FSIRX
- ➢ Sysconfig配置FSITX
- ➢ Sysconfig配置GPIO驱动LED4/LED5
- ➢ 编写应用代码



配置FSIRX



FSIRX (1 of 1 Added)

myFSIRX0

Name: myFSIRX0

Use Hardware: None

Frame Configuration

Number of Data Lines: DATA WIDTH 1 LANE

Software Frame Size: 6-Word

Loopback Mode: ☑

Tag Matching

Enable Tag Matching: □

Interrupts

Enable Interrupt: ☑

Use Interrupt: INT1, INT2

Enabled INT1 Interrupts: Received data frame

Enabled INT2 Interrupts: Hardware computed CRC error, Ping Watchdog times out +2

Register Interrupt Line1 Handler: ☑

Register Interrupt Line2 Handler: ☑

Ping Timeout

Enable PING Timeout: □

Delay Tap Selection

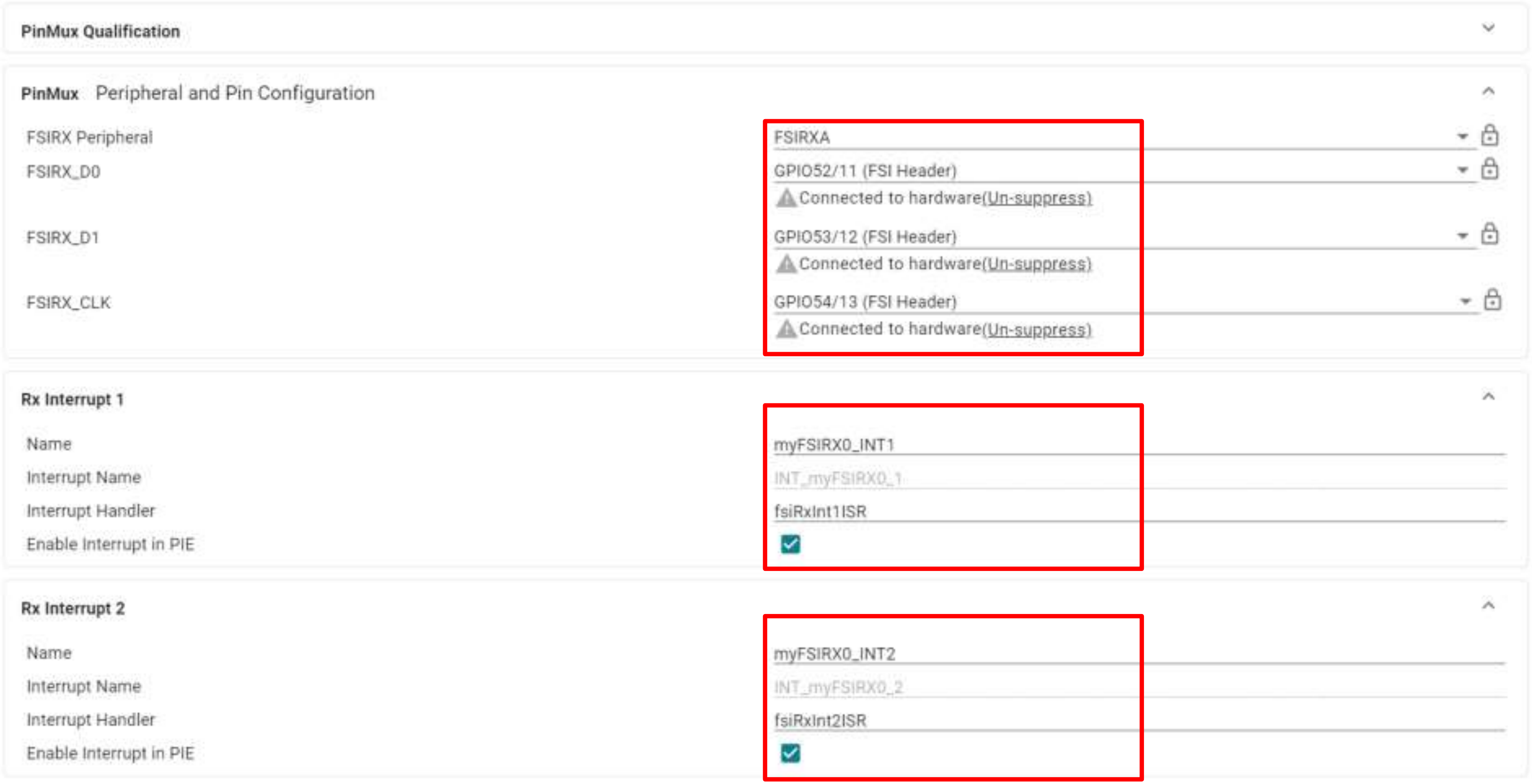
Enable Delay Tapping: □

PinMux Use Case: ALL

PinMux Qualification

- 单线传输
- 6字长度
- 使能回环模式
- 使能中断

配置FSIRX



PinMux Qualification

PinMux Peripheral and Pin Configuration

FSIRX Peripheral	FSIRXA
FSIRX_D0	GPIO52/11 (FSI Header) ⚠Connected to hardware(Un-suppress)
FSIRX_D1	GPIO53/12 (FSI Header) ⚠Connected to hardware(Un-suppress)
FSIRX_CLK	GPIO54/13 (FSI Header) ⚠Connected to hardware(Un-suppress)

Rx Interrupt 1

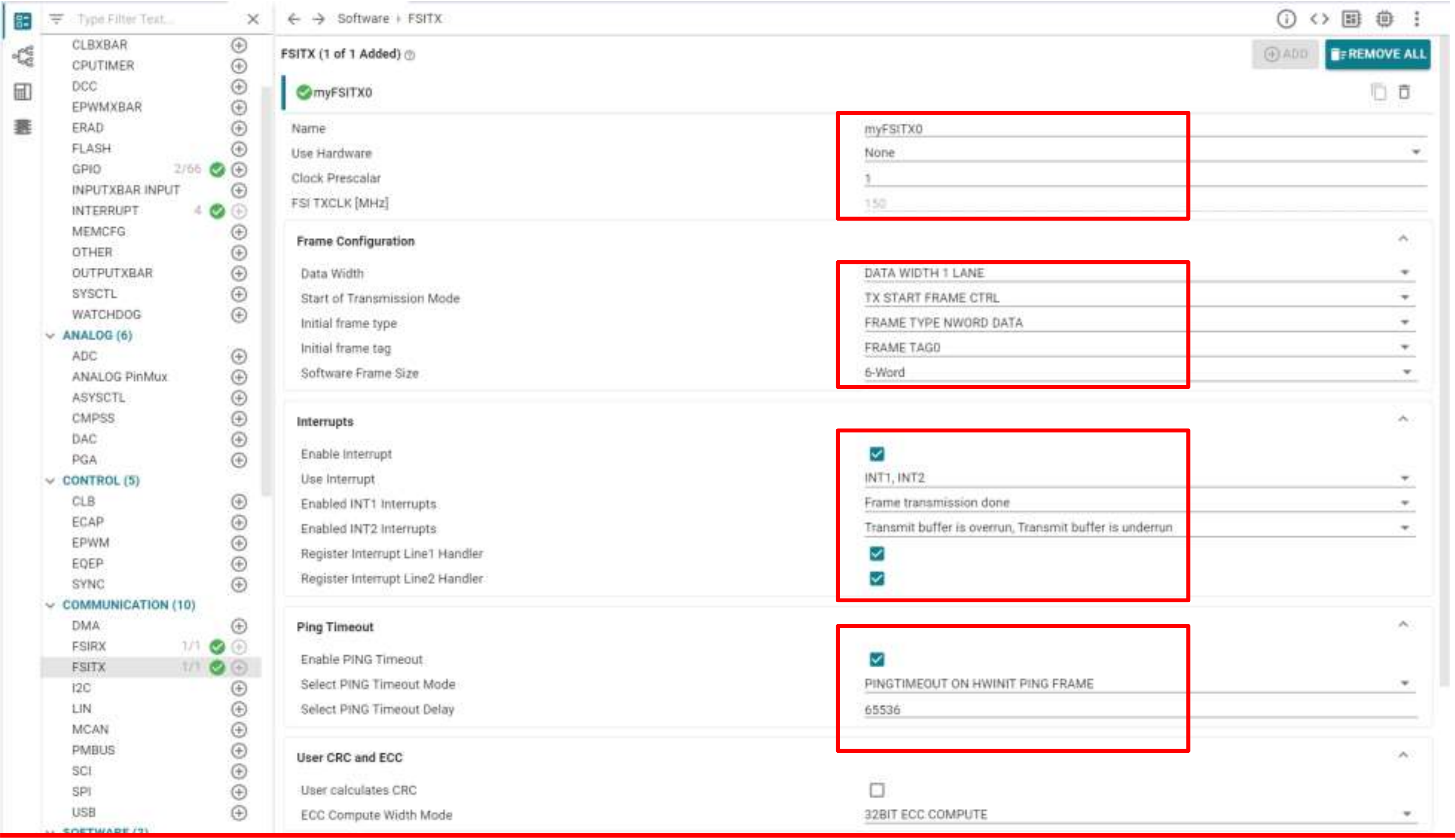
Name	myFSIRX0_INT1
Interrupt Name	INT_myFSIRX0_1
Interrupt Handler	fsiRxInt1ISR
Enable Interrupt in PIE	☑

Rx Interrupt 2

Name	myFSIRX0_INT2
Interrupt Name	INT_myFSIRX0_2
Interrupt Handler	fsiRxInt2ISR
Enable Interrupt in PIE	☑

- INT1接收中断
- INT2 超时、CRC

配置FSITX



FSITX (1 of 1 Added)

myFSITX0

Name	myFSITX0
Use Hardware	None
Clock Prescalar	1
FSI TXCLK [MHz]	150

Frame Configuration

Data Width	DATA WIDTH 1 LANE
Start of Transmission Mode	TX START FRAME CTRL
Initial frame type	FRAME TYPE NWORD DATA
Initial frame tag	FRAME TAG0
Software Frame Size	6-Word

Interrupts

Enable Interrupt	☑
Use Interrupt	INT1, INT2
Enabled INT1 Interrupts	Frame transmission done
Enabled INT2 Interrupts	Transmit buffer is overrun, Transmit buffer is underrun
Register Interrupt Line1 Handler	☑
Register Interrupt Line2 Handler	☑

Ping Timeout

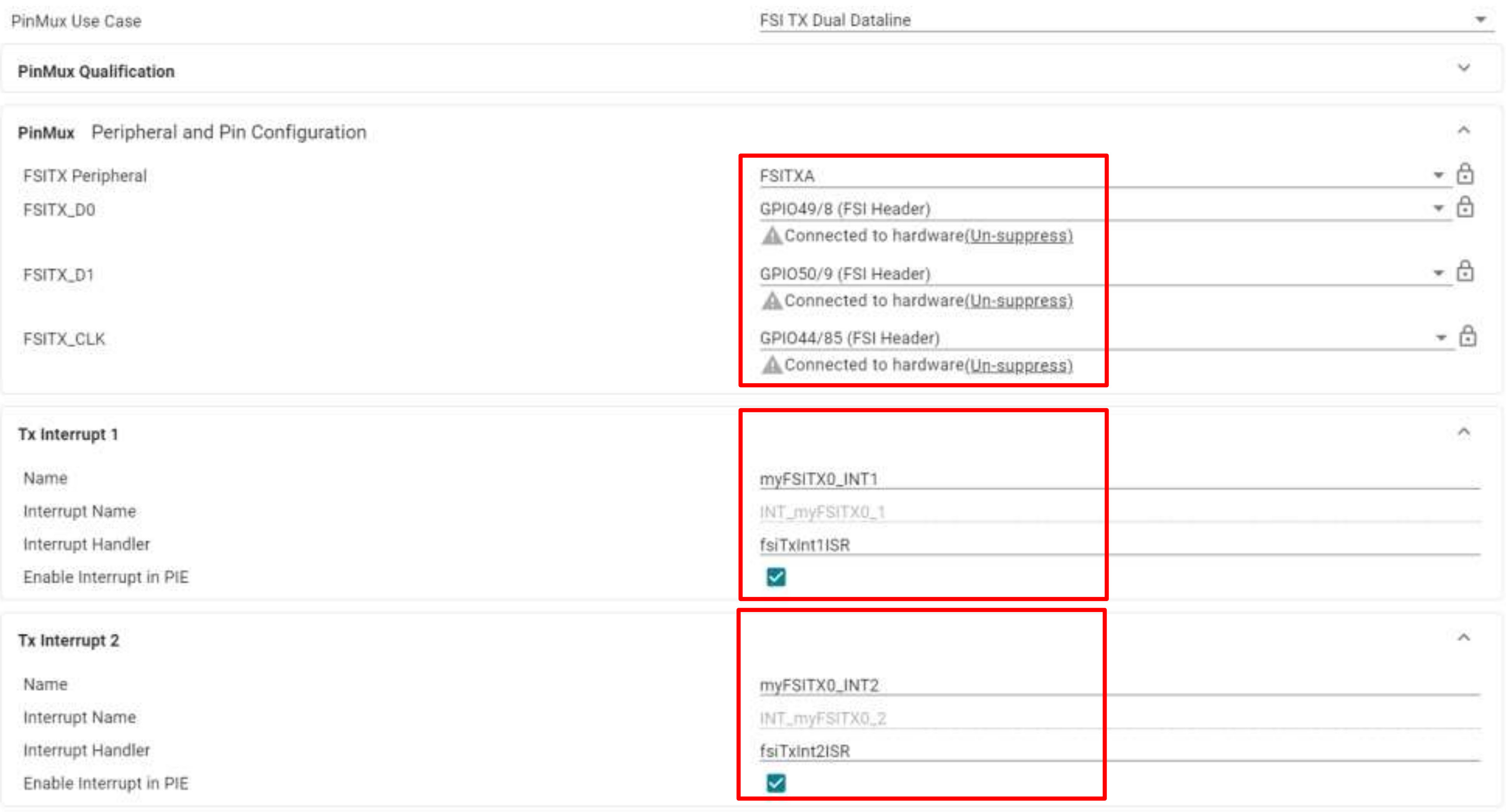
Enable PING Timeout	☑
Select PING Timeout Mode	PINGTIMEOUT ON HWINIT PING FRAME
Select PING Timeout Delay	65536

User CRC and ECC

User calculates CRC	☐
ECC Compute Width Mode	32BIT ECC COMPUTE

- 单线传输
- 6字长度
- 使能中断
- 使能Ping超时

配置FSITX



PinMux Use Case	FSI TX Dual Dataline

PinMux Qualification

PinMux Peripheral and Pin Configuration

FSITX Peripheral	FSITXA
FSITX_D0	GPIO49/8 (FSI Header) ⚠Connected to hardware(Un-suppress)
FSITX_D1	GPIO50/9 (FSI Header) ⚠Connected to hardware(Un-suppress)
FSITX_CLK	GPIO44/85 (FSI Header) ⚠Connected to hardware(Un-suppress)

Tx Interrupt 1

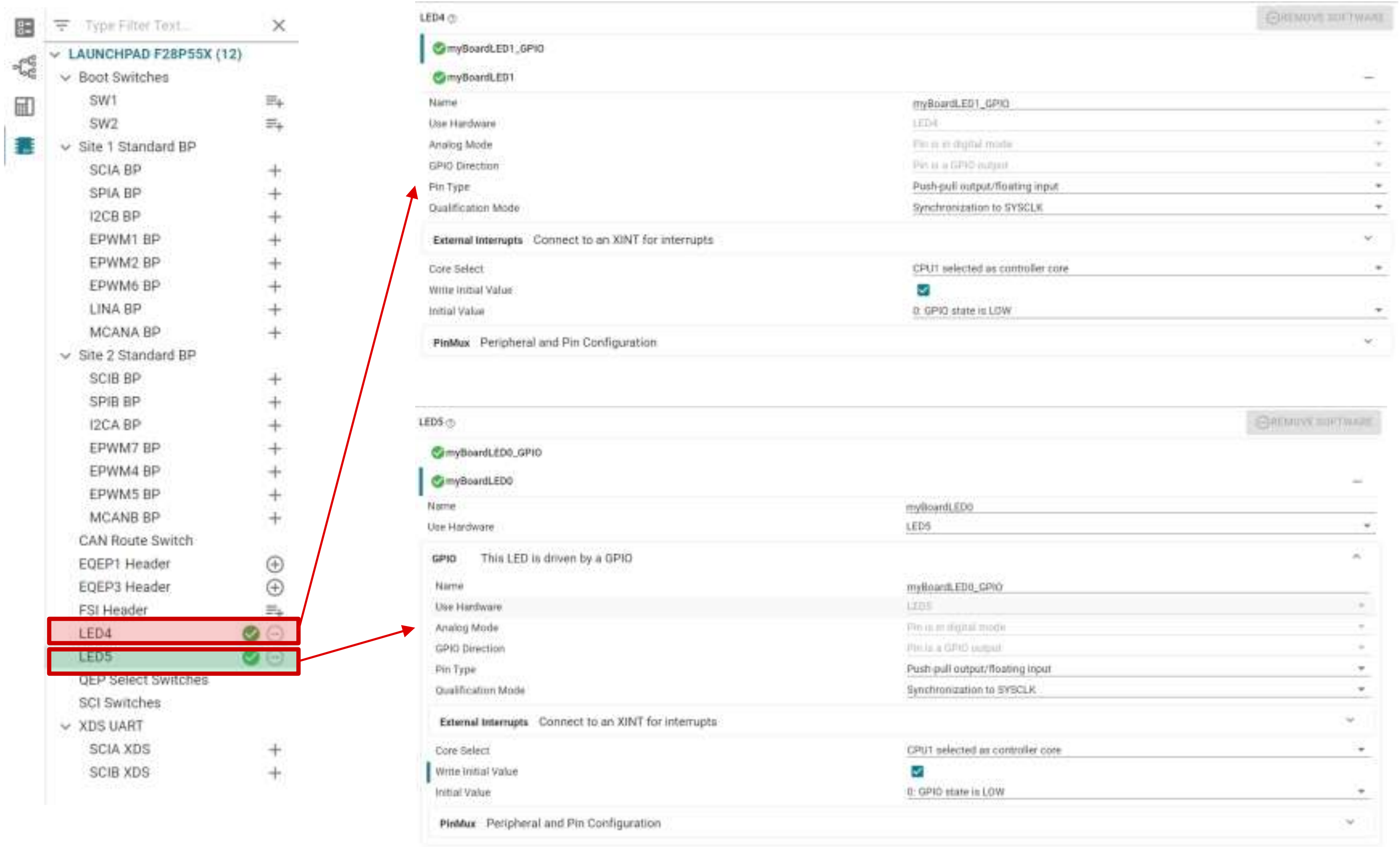
Name	myFSITX0_INT1
Interrupt Name	INT_myFSITX0_1
Interrupt Handler	fsiTxInt1ISR
Enable Interrupt in PIE	☑

Tx Interrupt 2

Name	myFSITX0_INT2
Interrupt Name	INT_myFSITX0_2
Interrupt Handler	fsiTxInt2ISR
Enable Interrupt in PIE	☑

- INT1 发送中断
- INT2 Ping超时中断

配置LED



LED4

myBoardLED1_GPIO

myBoardLED1

Name	myBoardLED1_GPIO
Use Hardware	LED4
Analog Mode	Pin is in digital mode
GPIO Direction	Pin is a GPIO output
Pin Type	Push-pull output/floating input
Qualification Mode	Synchronization to SYSCLK

External Interrupts Connect to an XINT for interrupts

Core Select	CPU1 selected as controller core
Write Initial Value	☑
Initial Value	0: GPIO state is LOW

PinMux Peripheral and Pin Configuration

LED5

myBoardLED0_GPIO

myBoardLED0

Name	myBoardLED0
Use Hardware	LED5

GPIO This LED is driven by a GPIO

Name	myBoardLED0_GPIO
Use Hardware	LED5
Analog Mode	Pin is in digital mode
GPIO Direction	Pin is a GPIO output
Pin Type	Push-pull output/floating input
Qualification Mode	Synchronization to SYSCLK

External Interrupts Connect to an XINT for interrupts

Core Select	CPU1 selected as controller core
Write Initial Value	☑
Initial Value	0: GPIO state is LOW

PinMux Peripheral and Pin Configuration

- LED4-发送数据闪烁

- LED5-接收数据闪烁

应用代码

```
//
// Included Files
//
#include "driverlib.h"
#include "device.h"
#include "board.h"
```

```
//
//Globals
//
uint16_t txBufData[16];                     //used to write to FSI transmit buffer
uint16_t rxDataArray[16];                   //buffer used to access FSI receive buffer
uint32_t dataFrameCntr = 0;                 //counts how many data frames have been received
uint16_t txEventSts = 0, rxEventSts = 0;    //captures FSITX/FSIRX event status in case of error
uint16_t txIntReceived = 0;                 //flag to signal TX interrupt has occurred
uint16_t rxIntReceived = 0;                 //flag to signal RX interrupt has occurred
uint32_t softwareTimeoutCntr = 0x100000;    //software watchdog counter in case interrupts fail

uint16_t txcnt = 0;
uint16_t rxcnt = 0;
```

应用代码

```
__interrupt void fsiRxInt1ISR(void)
{
    //
    //Set flag that interrupt has occurred, capture FSIRX event status, increment data frame count.
    //
    rxIntReceived = 1;
    rxEventSts = FSI_getRxEventStatus(FSIRXA_BASE);
    dataFrameCntr++;

    //
    //Transfer data from receive buffer to array
    //
    FSI_readRxBuffer(FSIRXA_BASE, rxDataArray, myFSIRX0_nWords, 0);

    //
    //Validate that data transmitted matches data received, otherwise terminate program.
    //
    uint16_t i;
    for(i = 0; i < myFSIRX0_nWords; i++)
    {
        if(rxDataArray[i] != txBufData[i])
        {
            ESTOP0;
        }
    }
    rxcnt++;
    if(rxcnt >= 35535)
    {

        rxcnt = 0;
        GPIO_togglePin(myBoardLED0_GPIO);
    }


    //
    // Clear the interrupt flag and issue ACK
    //
    FSI_clearRxEvents(FSIRXA_BASE,rxEventSts);
    Interrupt_clearACKGroup(INT_myFSIRX0_1_INTERRUPT_ACK_GROUP);
}
```