//----------------------------------

// Lab 3 - Part 3: SPI - Lab03\_spi.c

//----------------------------------

//

#include "init.h"

#include <stdio.h>

#include <stdlib.h>

//Declase Variables&Structs

SPI\_HandleTypeDef SPIHandle;//Declare struct

unsigned char input;

unsigned char Data;

unsigned char V\_MAJ;//Major version number

unsigned char V\_MIN;//Minor version number

unsigned char TMP\_LO;//LOW byte of temp read

unsigned char TMP\_HI;//HIGH byte of temp read

// See 769 Description of HAL drivers.pdf, Ch. 58.1 or stm32f7xx\_hal\_spi.c

void configureSPI()

{

    /\*In this function, only construct a SPI\_HandleTypeDef Struct and pass it

     \* in to HAL\_SPI\_Init function, GPIO Setups should be in HAL\_SPI\_MspInit(SPI\_HandleTypeDef \*hspi)

     \* function

     \*/

    \_\_SPI2\_CLK\_ENABLE();

    SPIHandle.Instance = SPI2; //Handler Instance set to SPI2

    SPIHandle.Init.Mode = SPI\_MODE\_MASTER; // Set master mode

    SPIHandle.Init.NSS = SPI\_NSS\_HARD\_OUTPUT;//Dr wilt says add this line

    SPIHandle.Init.TIMode = SPI\_TIMODE\_DISABLE; // Use Motorola mode, not TI mode

    SPIHandle.Init.CRCCalculation = SPI\_CRCCALCULATION\_DISABLE;//disable CRCcalc

    SPIHandle.Init.DataSize = SPI\_DATASIZE\_8BIT;// DataSize set to 8 Bit

    SPIHandle.Init.Direction = SPI\_DIRECTION\_2LINES;//two way communications

    SPIHandle.Init.BaudRatePrescaler = SPI\_BAUDRATEPRESCALER\_256;//Prescalr set to 216  to have BaudRate close to 1MHz

    SPIHandle.Init.CLKPolarity = SPI\_POLARITY\_HIGH;//Because the SCLK line is HIGH when idle, Clock Polarioty should be 1

    SPIHandle.Init.CLKPhase = SPI\_PHASE\_2EDGE;//Data is captured on the 2st edge of the clock,(the rising edge for this polarity)

    SPIHandle.Init.FirstBit = SPI\_FIRSTBIT\_LSB;//First bit is a master bit

    HAL\_SPI\_Init(&SPIHandle);//Invoke HAL SPI Init function

    // HAL\_SPI\_Init() will call HAL\_SPI\_MspInit() after reading the properities of

    // the passed SPI\_HandleTypeDef. After finishing the MspInit call, it will set

    // the SPI property bits. This is how all HAL\_[peripheral]\_Init() functions work.

}

/\*

 \* This is called upon SPI initialization. It handles the configuration

 \* of the GPIO pins for SPI.

 \*/

 // Do NOT change the name of an MspInit function; it needs to override a

 // \_\_weak function of the same name. It does not need a prototype in the header.

void HAL\_SPI\_MspInit(SPI\_HandleTypeDef \*hspi)

{

    // SPI GPIO initialization structure here

    \_\_HAL\_RCC\_GPIOB\_CLK\_ENABLE();

    \_\_HAL\_RCC\_GPIOA\_CLK\_ENABLE();

    GPIO\_InitTypeDef GPIOInit;

    GPIOInit.Mode = GPIO\_MODE\_AF\_PP;//GPIO set to alternate function with Push-Pull

    GPIOInit.Pin = (GPIO\_PIN\_15 | GPIO\_PIN\_14);//Pin 14 and 15

    GPIOInit.Pull = GPIO\_PULLUP;//Pull Up

    GPIOInit.Alternate =  GPIO\_AF5\_SPI2;

    HAL\_GPIO\_Init(GPIOB,&GPIOInit);//Initialize GPIOB

    GPIOInit.Pin = (GPIO\_PIN\_12);//and 12 Enable SCLK

    HAL\_GPIO\_Init(GPIOA,&GPIOInit);//Initialize GPIOA

    GPIOInit.Mode = GPIO\_MODE\_OUTPUT\_PP;

    GPIOInit.Pin = GPIO\_PIN\_11;

    HAL\_GPIO\_Init(GPIOA,&GPIOInit);//Initialize GPIOA - Pin11

    if (hspi->Instance == SPI2)

    {

        // Enable SPI GPIO port clocks, set HAL GPIO init structure's values for each

        // SPI-related port pin (SPI port pin configuration), enable SPI IRQs (if applicable), etc.

        // No GPIO is needed for Task 3 therefore no GPIO setup here

    }

}

//  This function is used to print a character on the top half of the terminal

void printTopHalf(char c){

    printf("\033[H");

    fflush(stdout);

    putchar(c);

    fflush(stdout);

}

void printBotHalf(char c){

    printf("\033[12;0H");

    fflush(stdout);

    putchar(c);

    fflush(stdout);

}

void readFWVerion(void){

    int i;

    unsigned char dummyChar = 0;

    unsigned char addr\_V\_MAJ = 0;//address V\_MAJ

    unsigned char addr\_V\_MIN = 1;//address V\_MIN

    for( i = 0; i < 1000; i++){asm("nop");}

    HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_11, GPIO\_PIN\_RESET);//Pull CS Pin LOW - slave selected

    HAL\_SPI\_Transmit(&SPIHandle, &addr\_V\_MAJ, 1, 10);//Transmit the V\_MAJ register address

    for( i = 0; i < 1000; i++){asm("nop");}

    HAL\_SPI\_TransmitReceive(&SPIHandle,&dummyChar, &V\_MAJ, 1, 100);//store major version number

    HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_11, GPIO\_PIN\_SET);//Pull CS Pin HIGH - slave not selected

    //

    //Above is one cycle of SPI transmit

    //

    for( i = 0; i < 1000; i++){asm("nop");}

    HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_11, GPIO\_PIN\_RESET);//Pull CS Pin LOW - slave selected

    HAL\_SPI\_Transmit(&SPIHandle, &addr\_V\_MIN, 1, 10);//Transmit the V\_MAJ register addres

    for(i = 0; i < 1000; i++){asm("nop");}

    HAL\_SPI\_TransmitReceive(&SPIHandle,&dummyChar,&V\_MIN, 1, 100);//store major version number

    HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_11, GPIO\_PIN\_SET);//Pull CS Pin HIGH - slave not selected

}

void changeDeviceId(void){

    unsigned char bit2 = 2;

    unsigned char bit7 = 128;

    unsigned char bit9 = 9;

    unsigned char ID = 0;

    unsigned char DeviceID = 0;

    int i = 0;

    HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_11,GPIO\_PIN\_RESET);//hardpull the NSS down

    HAL\_SPI\_Transmit(&SPIHandle, &bit9 ,1,10); // locate register 9 : DEVID Device identication number

    for(i=0; i<1000;i++){

        asm("nop");// wait 10 ms

    }

    HAL\_SPI\_Receive(&SPIHandle,&DeviceID,1,10);// read device ID

    HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_11,GPIO\_PIN\_SET);//hardpull the NSS up

    printf("\rOld Device ID:%c\r\n",DeviceID);

    printf("Type in new Device ID:\r\n");

    ID = getchar();

    HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_11,GPIO\_PIN\_RESET);//hardpull the NSS down

    HAL\_SPI\_Transmit(&SPIHandle,&bit2 ,1,10); // locate register 2 : CTL\_REG Control register

    for(i=0; i<1000;i++){

        asm("nop");// wait 10 ms

    }

    HAL\_SPI\_Transmit(&SPIHandle,&bit7,1,10);// change bit 7 of register 2 to 1// change the divice ID to write mode.

    HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_11,GPIO\_PIN\_SET);//hardpull the NSS up

    for(i=0; i<1000;i++){

        asm("nop");// wait 10 ms

    }

    HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_11,GPIO\_PIN\_RESET);//hardpull the NSS down

    HAL\_SPI\_Transmit(&SPIHandle, &bit9 ,1,10); // locate register 9 : DEVID Device identication number

    for(i=0; i<1000;i++){

        asm("nop");// wait 10 ms

    }

    HAL\_SPI\_Transmit(&SPIHandle,&ID,1,10);// send data

    HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_11,GPIO\_PIN\_SET);//hardpull the NSS up

}

void readTemp(void){

    unsigned char addr\_CTL = 2;//Address of control register

    unsigned char addr\_STS = 3;//Address of status register

    unsigned char addr\_TMP\_L = 5;//Address of TMP\_L register

    unsigned char addr\_TMP\_H = 6;//Address of TMP\_H register

    unsigned char status;//the status of tmp reading register

    unsigned char Bit1 = 2;//BIt 1'

    int i = 0;

    HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_11, GPIO\_PIN\_RESET);//Pull CS Pin LOW - slave selected

    HAL\_SPI\_Transmit(&SPIHandle, &addr\_CTL, 1, 10);//Transmit the control register address

    for( i = 0; i < 1000; i++){asm("nop");}

    HAL\_SPI\_Transmit(&SPIHandle, &Bit1 , 1, 10);//Write control register bit 1 to 1 to request a temp read.

    HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_11, GPIO\_PIN\_SET);//Pull CS Pin HIGH - slave not selected

    //Above is the first cycle

    while((status&8) == 0){//will stay in loop until tmp read is ready

        HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_11, GPIO\_PIN\_RESET);//Pull CS Pin LOW - slave selected

        HAL\_SPI\_Transmit(&SPIHandle, &addr\_STS, 1, 10);//Transmit the status register address

        for( i = 0; i < 1000; i++){asm("nop");}

        HAL\_SPI\_Receive(&SPIHandle, &status, 1, 10);//Read status register bit 3  to request a temp read.

        HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_11, GPIO\_PIN\_SET);//Pull CS Pin HIGH - slave not selected

    }

    //above is the second cycle

    HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_11, GPIO\_PIN\_RESET);//Pull CS Pin LOW - slave selected

    HAL\_SPI\_Transmit(&SPIHandle, &addr\_TMP\_L , 1, 10);//Transmit the TMP\_LO register address

    for( i = 0; i < 1000; i++){asm("nop");}

    HAL\_SPI\_Receive(&SPIHandle, &TMP\_LO , 1, 10);//Read TMP\_LO data and store

    HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_11, GPIO\_PIN\_SET);//Pull CS Pin HIGH - slave not selected

    //above is the third cycle

    HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_11, GPIO\_PIN\_RESET);//Pull CS Pin LOW - slave selected

    HAL\_SPI\_Transmit(&SPIHandle, &addr\_TMP\_H , 1, 10);//Transmit the TMP\_LO register address

    for( i = 0; i < 1000; i++){asm("nop");}

    HAL\_SPI\_Receive(&SPIHandle, &TMP\_HI , 1, 10);//Read TMP\_LO data and store

    HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_11, GPIO\_PIN\_SET);//Pull CS Pin HIGH - slave not selected

}

void ResetSlaveTerminal(void){

    unsigned char bit2 = 2;

    unsigned char bit2\_3 = 0x06;

    int i = 0;

    HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_11,GPIO\_PIN\_RESET);//hardpull the NSS down

    HAL\_SPI\_Transmit(&SPIHandle,&bit2 ,1,10); // locate register 2 : CTL\_REG Control register

    for( i = 0; i < 1000; i++){asm("nop");}

    HAL\_SPI\_Transmit(&SPIHandle,&bit2\_3,1,10);// change bit 2 and 3 of register 2 to 1// clear the STATS terminal.

    HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_11,GPIO\_PIN\_SET);//hardpull the NSS up

}

void menu(void){

    char user\_in;

    printf("\033[7;0H");//Move to the position to print menu

    fflush(stdout);//flush stdout

    //print menu

    printf("1 -> Read the slave’s firmware\r\n");

    printf("2 -> Read Temp\r\n");

    printf("3 -> Change Device ID\r\n");

    printf("4 -> Clear Terminal\r\n");

    user\_in = getchar();

    if(user\_in == '2'){

        readTemp();

        printf("\033[4;50H");//Move right to print tempreture

        fflush(stdout);//flush stdout

        int temp = (((int)TMP\_HI<<8) + TMP\_LO);

        printf("TMP: %u\r\n",temp);

    }

    if(user\_in == '1'){

        readFWVerion();

        printf("\033[1;24H");//Center cursor for head line print

            fflush(stdout);//flush stdout

            printf("STaTS Firmware Version is %u.%u\r\n", V\_MAJ,V\_MIN);

    }

    if(user\_in == '3'){

        printf("\033[24;24H");//Center cursor for head line print

        fflush(stdout);//flush stdout

        changeDeviceId();

    }

    if(user\_in == '4'){

        printf("\033[2J");

        fflush(stdout);

        ResetSlaveTerminal();

    }

}

int main(void)

 {

    Sys\_Init();

    //Configure SPI setups

    configureSPI();

    //Pull NS pin HIGH

    HAL\_Delay(1000);

    HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_11, GPIO\_PIN\_SET);

    readFWVerion();//Initial read of firmware version

    printf("\033[1;24H");//Center cursor for head line print

    fflush(stdout);//flush stdout

    printf("STaTS Firmware Version is %u.%u\r\n", V\_MAJ,V\_MIN);

    printf("\033[4;50H");//Move right to print tempreture

    fflush(stdout);//flush stdout

    int temp = (((int)TMP\_HI<<8) + TMP\_LO);

    printf("TMP: %u\r\n",temp);

    char c;

    while(1){

        printf("Enter ESC key for menu\r\n");

        c = getchar();

        if(c == 27){//ESC

            menu();//show menu

        }

    }

    //excution code

    while(1){

    }

// See 769 Description of HAL drivers.pdf, Ch. 58.2.3 or stm32f7xx\_hal\_spi.c

//

//  HAL\_StatusTypeDef HAL\_SPI\_TransmitReceive(SPI\_HandleTypeDef \*hspi, uint8\_t \*pTxData, uint8\_t \*pRxData, uint16\_t Size, uint32\_t Timeout)

//

}