/\*

\* main.c

\*

\* Created: 9/1/2023 12:21:16 PM

\* Author: 2161908

\*/

#define *F\_CPU* 1000000UL

#include <util/delay.h>

#include <avr/io.h>

#include <avr/sfr\_defs.h>

#define DDR\_SPI DDRB

#define CS 2

#define CS\_SoftCode 1

#define MOSI 3

#define MISO 4

#define SCK 5

void SPI\_MasterInit()

{

/\* Set MOSI, CS and SCK output, all others input \*/

DDR\_SPI = (1<<CS)|(1<<MOSI)|(1<<SCK);

/\* Enable SPI, Master, (j'ai aussi fuck avec le clock)set clock rate fck/16 \*/

SPCR = (1<<SPE)|(1<<MSTR)|(1<<CPOL)|(1<<CPHA)|(1<<SPR1);

/\* pour tous les mettres passif (les Slaves) \*/

PORTB |= (1<<CS)|(1<<CS\_SoftCode);

*\_delay\_ms*(10);

}

char SPI\_MasterTransmit\_Receive(char cData)

{

/\* Start transmission \*/

SPDR = cData;

/\* Wait for transmission complete \*/

while(!(SPSR & (1<<SPIF))); //le ! veux dire while not (so basicly... SPIF veut dire que les 8 bytes ont ete Read/Wright donc cest basicly un loop qui t'empeche d'envoyer plus information avant que le 8 bytes a été envoyé)

return SPDR;

}

void Blink(){

PORTC |= (1<<0);

*\_delay\_ms*(1000);

PORTC &= ~(1<<0);

*\_delay\_ms*(1000);

PORTC |= (1<<1);

*\_delay\_ms*(1000);

PORTC &= ~(1<<1);

*\_delay\_ms*(1000);

}

int main(void)

{

/\* mettre les deux portes en sortie \*/

DDRC |= (1<<0)|(1<<1);

/\* test de lumière pour confirmer \*/

Blink();

/\* initialiser le master SPI \*/

SPI\_MasterInit();

/\* pour envoyer plusieurs char test \*/

char Master\_data[] = {0x55,0x33,0x44};

/\* choisir quel chip on veut communiquer (doit envoyer LOW logic)\*/

PORTB &= ~(1<<CS); //pour slave 1

*\_delay\_ms*(10);

for (int i=0; i<4; i++)

{

char data = SPI\_MasterTransmit\_Receive(Master\_data[i]);

if(data == 0x51){

PORTC |= (1<<0);

}

if(data == 0x31){

PORTC |= (1<<1);

}

if(data == 0xAA){

PORTC &= ~(1<<0);

PORTC &= ~(1<<1);

//ici cest i<4 cuz quand tu envoie au slave et il change sont registre SPDR, il va etre envoié dans le prochain echange de SPDR, comme si il y avait un delay de 1 char quand le slave parle au master

PORTB |= (1<<CS); //pour mettre slave 1 a passive

}

*\_delay\_ms*(1000);

}

/\* choisir quel chip on veut communiquer (doit envoyer LOW logic)\*/

PORTB &= ~(1<<CS\_SoftCode);//pour slave 2

*\_delay\_ms*(10); //ici le delay est important cuz sinon il active le slave en meme temps que le slave veux parler so ca fuck up

for (int i=0; i<4; i++)

{

char data = SPI\_MasterTransmit\_Receive(Master\_data[i]);

if(data == 0x51){

PORTC |= (1<<0);

}

if(data == 0x31){

PORTC |= (1<<1);

}

if(data == 0xAA){

PORTC &= ~(1<<0);

PORTC &= ~(1<<1);

//ici cest i<4 cuz quand tu envoie au slave et il change sont registre SPDR, il va etre envoié dans le prochain echange de SPDR, comme si il y avait un delay de 1 char quand le slave parle au master

PORTB |= (1<<CS\_SoftCode); //pour mettre slave 1 a passive

}

*\_delay\_ms*(1000);

}

return(0);

}

/\*

IMPORTANT !!!!

le master et le slave peuvent utiliser le même registre car les deux sont en train de

byte shift en meme temps comme SPDR master envoye 01010101 au SPDR slave et a mesure

que le SPDR slave envoie par pushing les bytes vers le master, il est en train de recevoir

les bytes 1 par 1 en meme temps

Master ==> 01010101

Slave ==> 00110011

master envoie 0,1,0,1,0,1,0,1

master recoit 0,0,1,1,0,0,1,1

son SPDR registre ce vide pour faire de la place au SPDR du slave a mesure qu'il envoie de bite

\*/