Anfragen von Daten vom Slave per SPI

Student Group

First Name	Surname	Matrikel Nr.

Table of Contents

Anfragen von Daten vom Slave per SPI

Ein einfaches Beispiel wie ein SPI-Master von einem SPI-Slave Daten anfragen kann ist hier zu finden: spi_slavetest.zip
In der Datei ist auch das passende Simulide File vorhanden

Code des Master

```
#define F_CPU               16000000UL

#include <avr/io.h>
#include <util/delay.h>
#include <stdint.h>

#define SET_BIT(BYTE, BIT)  ((BYTE) |=  (1 << (BIT)))
#define CLR_BIT(BYTE, BIT)  ((BYTE) &= ~(1 << (BIT)))

#define SLAVE_REQUEST       0x11            // The value can be changed

void SPI_MasterTransmit(uint8_t data);

int main()
{                                           // Set the following Pins as
output
    DDRB =     (1<<DDB2)                        //  - !SS
            |(1<<DDB3)                        //  - MOSI
            |(1<<DDB4)                        //  - MOSI
            |(1<<DDB5);                       //  - SCK
    SPCR =   (1<<SPE)|(1<<MSTR)|(1<<SPR0);    // Enable SPI, Master mode,
clock rate fck/16
    DDRD = 0xFF;                            // Set PORTD as output
    while(1) {
        SPI_MasterTransmit(SLAVE_REQUEST);    // Send request to slave

        uint8_t dataFromSlave = SPDR;        // Receive the answer from the
slave
        PORTD = dataFromSlave^0xFF;           // Output the answer on
PORTC, but all Bits inverted
    }
}

void SPI_MasterTransmit(uint8_t data)
{
    CLR_BIT(PORTB, PORTB2);                   // clear !SS to activate
Slave
    SPDR = data;                            // Start transmission
    while(!(SPSR & (1<<SPIF)));               // Wait for transmission
complete
```

```
    SET_BIT(PORTB, PORTB2);                    // Set !SS to deactivate
Slave
}
```

Code des Slave

```
#include <avr/io.h>
#include <avr/interrupt.h>

#define SLAVE_REQUEST       0x11        // The value can be changed

volatile uint8_t dataToMaster   = 0x00;
volatile uint8_t dataFromMaster = 0x00;

int main() {
    DDRB = (1<<DDB4);                    // MISO as output
    PORTD = 0xFF;                        // Activate all pull-up resistors
for PIND, to read switch states correctly
    SPCR = (1<<SPE)|(1<<SPIE);            // Enable SPI and SPI interrupt
    sei();                                // Enable global interrupts

    while(1)
    {
                                        // Main application code
    }
}

ISR(SPI_STC_vect) {
    dataFromMaster = SPDR;
    if (dataFromMaster == SLAVE_REQUEST)
    {
        dataToMaster=PIND;
        SPDR = dataToMaster;            // Load data into the buffer
    }
}
```