

# spi\_kommunikation

## Student Group

| First Name | Surname | Matrikel Nr. |
|------------|---------|--------------|
|            |         |              |
|            |         |              |
|            |         |              |

## Table of Contents

**SPI Kommunikation**

Prelude

Master

Slave

.....

.....

.....

.....

2

2

2

3

# SPI Kommunikation

Bei der Simulide Software 1255 und jünger kommt es vor, dass am Slave die Daten nicht durch das SPI eingelesen werden können.

Im Folgenden ist ein Bugfix beschrieben.

## Prelude

- **ACHTUNG: Der beschriebene Bugfix erlaubt nur eine Kommunikation vom Master zum Slave. Eine Rückkommunikation ist hiermit NICHT möglich.**
- Eine Verwendung des SPI Interrupt ( `ISR(SPI_STC_vect)` ) ist NICHT möglich.
- Der Code, die Beispiel hex Dateien und eine Sim1-Datei liegen unter folgendem Link ab: [spi\\_bugfix.zip](#)

## Master

### Änderungen

- Es sollte nur `SPCR = ... | (1<<SPR1) | (1<<SPR0);` oder `SPCR = ... | (1<<SPR1);` verwendet werde und keine schnelleren SPI-Frequenzen
- Der Bugfix wurde nur mit einer zeitlichen Verzögerung von ca.  $10\mu s$  zwischen dem Versenden von Nachrichten getestet.  
Bei kürzeren Verzögerungen kann es dazu kommen, dass beim Empfänger die Daten im SPDR Register gestört ankommen.

```
#ifndef F_CPU
#define F_CPU 12288000UL
#endif

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

uint8_t i=0;
int main()
{
    DDRC  &= 0b11111100;
    PORTC |= 0b00000011;
    DDRB  = 0b00111100;
    SPCR  = (1<<SPE)|(1<<MSTR)|(1<<SPR1)|(1<<SPR0);    // bugfix nur mit
(SPR1=1, SPR0=0) oder (SPR1=1, SPR0=1) möglich
    while(1)
    {
        if(!(PINC & 1<<PINC0))
        {
            PORTB &= ~(1 << PORTB2);
            SPDR = 0b10101100;
            while(!(SPSR & (1<<SPIF)));
            PORTB |= 1 << PORTB2;
        }
    }
}
```

```

        _delay_us(10);                // für SPI Bugfix: 10us sollte
reichen, falls die Daten nicht korrekt übertragen werden, dann Wert erhöhen
    }
}
}

```

## Slave

### Änderungen

- Es ist zusätzlich die Unterfunktion `SPIsoft_Init()` hinzugekommen, welche einen Pin Change Interrupt auf die CLK und SS Leitung legt.
- Ein Rückversand der Daten vom Slave zum Master (MISO) ist nicht möglich. Entsprechend ist von einem Beschreiben von SPDR (Zeile 22) abzusehen, da diese den Bugfix stören.
- Das SPIF Bit kann genutzt werden. Vor der Verwendung der Daten muss aber eine zeitliche Verzögerung von ca.  $10\mu s$  gewartet werden.  
Bei kürzeren Verzögerungen kann es dazu kommen, dass beim Empfänger die Daten im SPDR Register gestört ausgewertet.
- Zusätzlich wird der Interrupt `ISR(PCINT0_vect)` zur Signalerkennung genutzt.

```

#ifndef F_CPU
#define F_CPU 12288000UL
#endif

#include <avr/io.h>
#include <avr/interrupt.h>           // Definition von Interrupts
#include <util/delay.h>

void SPIsoft_Init();

int main()
{
    DDRB = (1<<DDB4);

    DDRD = 0xFF;
    SPCR = (1<<SPE);
    SPIsoft_Init();                  // für SPI Bugfix - Init
Routine
    while(1)
    {
        // SPDR = 0x01;
        while(!(SPSR & (1<<SPIF)));
        _delay_us(10);              // für SPI Bugfix: 10us sollte
reichen, falls die Daten nicht korrekt übertragen werden, dann Wert erhöhen
        PORTD = SPDR;
    }
}

void SPIsoft_Init()

```

```
{
    PCICR |= 1<<PCIE0;                // Pin Change Interrupt, auf PortB
aktivieren
    PCMSK0 |= 1<<PINB5 |              // Pin Change Interrupt, wenn CLK
Pin geändert
    1<<PINB2;                        // Pin Change Interrupt, wenn SS
Pin geändert
    sei();
}

ISR(PCINT0_vect)
{
    if( !(PINB>>PINB5) & 1 ) {return;}; // Falls nicht
steigende Flanke , dann abbrechen
    SPDR = ( SPDR << 1 ) + ((PINB>>PINB3)& 1); // Bit-Wert
von PinB3 in SPDR schieben
}
```

From:  
<https://first.mexle.te.hs-heilbronn.de/> - **MEXLE Wiki**

Permanent link:  
[https://first.mexle.te.hs-heilbronn.de/microcontrollertechnik/spi\\_kommunikation?rev=1656057588](https://first.mexle.te.hs-heilbronn.de/microcontrollertechnik/spi_kommunikation?rev=1656057588)

Last update: **2022/06/24 09:59**

