PROTOCOL

for lab-exercise

I^{2}C

Group / Class 7 / 5BHELS	Script writer A. Hofstätter	Signature
Date of Exercise / Hand-in Date 15.12.2015 22.12.2015	Team member S. Biehl	Signature
Teacher CRHA	Team member	Signature
Grade	Team member	Signature

I^{2}C

ATmega 32u4

USED DEVICES

Number	Device	Company	Type	Inventory Number
1	Oscilloscope	Agilent Technologies	DSO-X 2014A	-
2				
3				

Stored on el-lab file Server: ________________

1)Task Description

Portexpander PCF8574

1)Task

- A Bit pattern “1010 1010” have to be produced on the expander pins
- The data and clock Signal on the I2C Bus should be Protocolled and interpreted

2)Task

- Investigate the maximal frequency a port pin on the expander can be toggled

3)Task

- 2 LEDs and a Button have to be connected to the port expander
- When the Button is not pushed the LEDs should toggle
- When the Button is pushed both LEDs should shine

2.Circuit

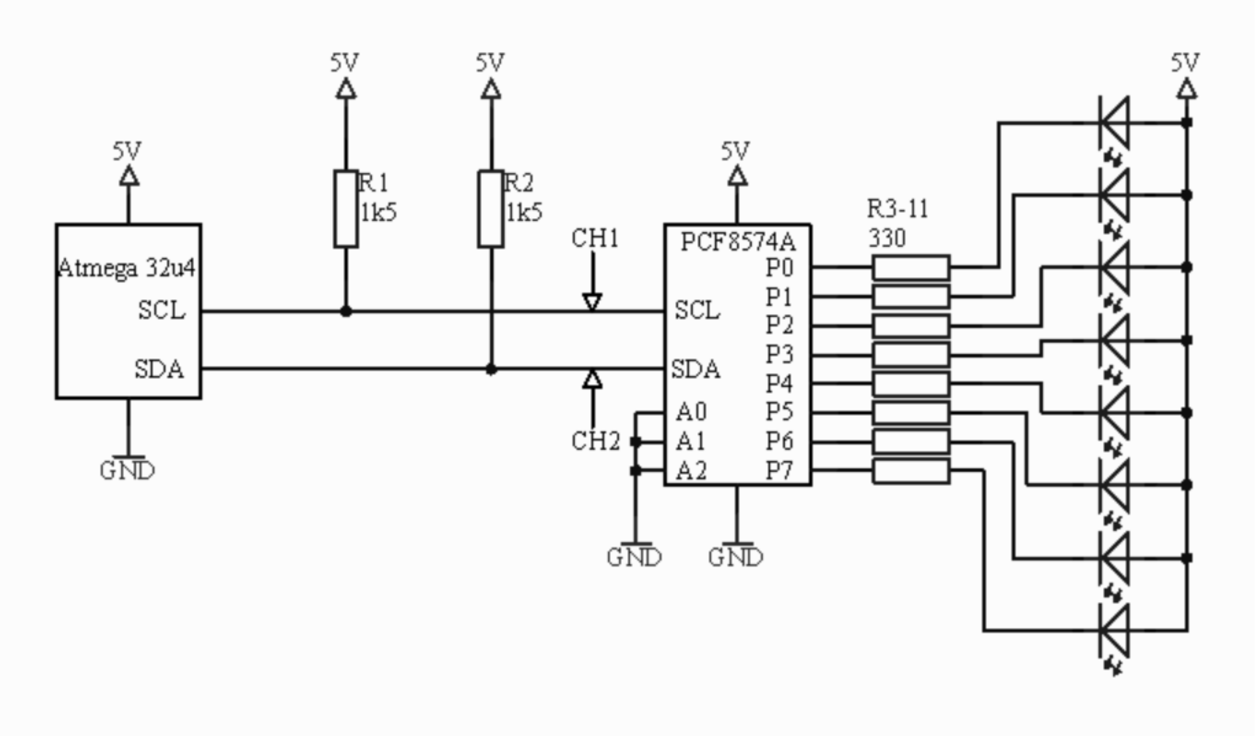

Fig.1: Circuit

In Fig.1 you can see the circuit, which was used for this Lab exercise. The power supply of this circuit is the USB port (5V). To send the information from the ATmega32u4 to the port expander the two I^{2}C Pins (SDA and SCA) are used. The two Pull up resistors on the SDA and SCA Pins are necessary because a defined potential is always needed, the 1k5 resistors are recommended by the datasheet of the PCF8574. The Ports A0, A1, A2 are used to set the address of the PCF8574.

1) A Bit pattern “1010 1010” have to be produced on the expander pins

```
#define F_CPU 16000000

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

int main(void)
{

        CLKPR = 0x80;//Interner Clock-Prescaler to 1
        CLKPR = 0x00;//f_CPU

        DDRD = DDRD | (1<<DDD0) | (1<<DDD1);//PIN0 & PIN1 as Output

        TWBR = 32;

        while(1)
        {
                TWCR = TWCR | (1<<TWINT) | (1<<TWSTA) | (1<<TWEN); //START

                while(!(TWCR&(1<<TWINT)));//wait until Start is sent

                TWDR = 0b01110000;//Adr. 0111 0000W + Write (W=0)

                TWCR = (1<<TWINT) | (1<<TWEN);//Send

                while(!(TWCR&(1<<TWINT))); //wait until Address is sent

                TWDR = 0b10101010;//Data Byte

                TWCR = (1<<TWINT) | (1<<TWEN);//Send Data Byte

                while(!(TWCR&(1<<TWINT))); //Wait until Data is sent

                TWCR = TWCR | (1<<TWINT) | (1<<TWSTO) | (1<<TWEN);           //STOP

        }
}
```

The Clock on the IC is set on 16MHz. The SDA and the SCL PINs of the ATmega are on PORTD, because of that the PIND0 and PIND1 are set as output. At the beginning of the main loop the “Start” bits are sent, after this there is a “while-loop” to wair until the bits are completely sent. Then the address bits must be sent, this bit sequence can be read out of the Datasheet. According to that the data byte is written into the TWDR registry and sent, after waiting until the bits are sent, the “Stop” sequence is sent and the loop starts from new.

1.1 The data and clock Signal on the I2C Bus should be Protocolled and interpreted

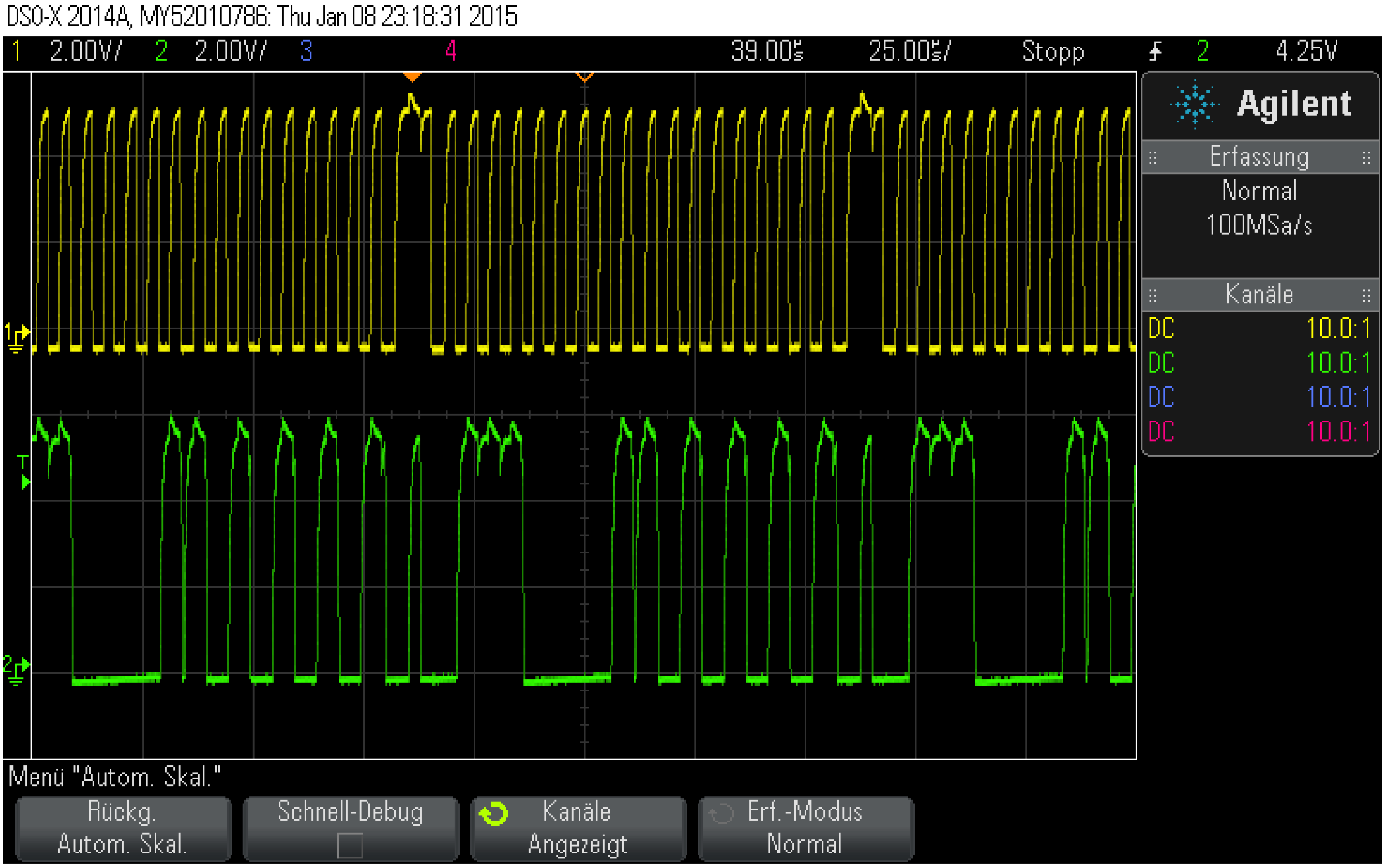

Fig.2: Data and Clock Signal on the I2C Bus

In Fig.2 you can see the bit sequence of SDA and SDL which are sent from the uC to the port expander.

2) Investigate the maximal frequency a port pin on the expander can be toggled

```
#define F_CPU 16000000

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

int main(void)
{

       CLKPR = 0x80; //Interner Clock-Prescaler to 1
       CLKPR = 0x00; //f_CPU

       DDRD = DDRD | (1<<DDD0) | (1<<DDD1);
       DDRB = DDRB | (1<<DDB0);

       TWBR = 32;

       while(1)
    {
              TWCR = TWCR | (1<<TWINT) | (1<<TWSTA) | (1<<TWEN); //START
              while(!(TWCR&(1<<TWINT))); //wait until Start is sent

              TWDR = 0b01110000; //Adr. 0111 0000W + Write (W=0)
              TWCR = (1<<TWINT) | (1<<TWEN);//Send
              while(!(TWCR&(1<<TWINT))); //wait until Address is sent


              while(1)
              {
                    TWDR = 0xff; //Data Byte
                    TWCR = (1<<TWINT) | (1<<TWEN);//Send Data Byte
                    while(!(TWCR&(1<<TWINT)));//Wait until Data is sent

                    TWDR = 0x00;
                    TWCR = (1<<TWINT) | (1<<TWEN); //Data Byte sende
                    while(!(TWCR&(1<<TWINT)));
              }

              TWCR = TWCR | (1<<TWINT) | (1<<TWSTO) | (1<<TWEN); //STOP

    }
}
```

The program of the first task was modified. Instead of sending the data byte once, there is a “while-loop” between the address and the Stop sequence. In the loop the port Pins are toogled and the stop bit is never sent.