­

**PROTOKOLL**zur Laborübung

***AVR UART***



|  |  |  |
| --- | --- | --- |
| Gruppe / Klasse | Protokollführer | Unterschrift |
| 7 / **5BHELS** | HOFSTÄTTER A. |  |
| Übungs- / Abgabedatum | Mitarbeiter | Unterschrift |
| 13. Okt 2015  13. Okt 2015 | BIEHL S. |  |
| Lehrer | Mitarbeiter | Unterschrift |
| CRHA | HIRSCH L. |  |
| Note | Mitarbeiter | Unterschrift |
|  |  |  |
| ***AVR UART***  *ATmega32U4* | | |
| **Verwendete Geräte**   |  |  |  |  |  | | --- | --- | --- | --- | --- | | Nr. | Gerätebezeichnung | Hersteller | Typ | Platznummer | | - | - | - | - | - |   **Verwendete Programme**   |  |  |  | | --- | --- | --- | | Nr. | Name | Version | | 1. | XCode | - | | 2. | DFU-Programmer | - | | | |

ÜBUNGS-/ABGABE-DATUM

Klasse /Gruppe

NOTE

LEHRER

# Inhaltsverzeichnis

[1 Inhaltsverzeichnis 2](#_Toc432498249)

[2 Aufgabenstellung 3](#_Toc432498250)

[2.1 Messablauf 3](#_Toc432498251)

[3 Grundprogramm 4](#_Toc432498252)

[4 Program-Listing Verzeichnis 6](#_Toc432498253)

# Aufgabenstellung

Aufgabe dieser Laborübung war es mit Hilfe eines ATmega32U4 ein per RS323 an den PC zu senden. Dies wurde über eine UART Kommunikation mit dem ST232 realisiert.

Folgende Aufgabenstellung war zu erledigen:

* Buchstaben senden; „C“ senden
* String senden; z.B. „Wir sind im Labor“
* Aufnahme der Oszillogramme
* Steuerung einer LED per PC (Serial)

## Messablauf

Zuerst wurde der ATmega mit dem Pegelwandler verbunden. Dieser sorgt dafür, dass das UART Signal auf die korrekten Pegel für eine serielle Verbindung gebracht wird.

Anschließend wurde ein Programm im C-Code geschrieben und durch den avr-gcc kompiliert. Mit Hilfe des „DFU-Programmers“ wurde das kompilierte und umgewandelte HEX File auf den AVR geflasht.

**Compiler Aufrufe unter UNIX**

**avr-gcc** main.c -Os -std=c99 -o output.elf -mmcu=atmega32u4;

**avr-objcopy** -R .eeprom -R .fuse -R .lock -R .signature -O ihex output.elf output.hex

**Loadhex.sh** atmega32u4 erase flash output.hex start

**Inhalt des benutzerdefinierten Scripts “LoadHex.sh” zum flashen unter UNIX**

**dfu-programmer $1 $2**

**dfu-programmer $1 $3 $4**

**dfu-programmer $1 $5**

**exit**

# ****Schaltung****



ATmega32U4

# Berechnung der Baudrate

Der Controller verwendet Timer, um die gewünschte Bitdauer und somit Baudrate zu erzeugen. Im Datenblatt ist dazu eine Formel zu finden.

In der Laborübung wurde eine Baudrate von 115200 verwendet. Dieser Wert wurde im Programm definiert.

# Grundprogramm

#define F\_CPU 16000000UL //Takt

#define BAUD 115200UL //gewünschte Baudrate

#define UBRR\_CALC (F\_CPU/16UL/BAUD-1) //Baudrtate aus Takt berechnen

#include <avr/io.h>

#include <avr/interrupt.h>

char i=0; //Laufvariable

char eingang; //Variable für empf. Zeichen

char Line[40]; // String mit maximal 39 zeichen

void init\_usart (void)

{

UBRR1H = (unsigned char)(UBRR\_CALC>>8); //Baudrate einstellen

UBRR1L = (unsigned char)(UBRR\_CALC);

UCSR1B |= (1<<RXEN1)|(1<<RXCIE1)|(1<<TXEN1);//Empf. ein, Empf.-Interr. ein, Sender ein

UCSR1C |= (1<<UCSZ11)|(1<<UCSZ10); //asynchr., 8 Daten-, kein Parity-, 1 Stop

}

/\* Einzelnes Zeichen empfangen \*/

uint8\_t uart\_getc(void)

{

while (!(UCSR1A & (1<<RXC1))) // warten bis Zeichen verfuegbar

;

return UDR1; // Zeichen aus UDR an Aufrufer zurueckgeben

}

// gesamten string empfangen

void uart\_gets( char\* Buffer, uint8\_t MaxLen )

{

uint8\_t NextChar;

uint8\_t StringLen = 0;

NextChar = uart\_getc(); // Warte auf und empfange das nächste Zeichen

// Sammle solange Zeichen, bis:

// \* entweder das String Ende Zeichen kam

// \* oder das aufnehmende Array voll ist

while( NextChar != '.' && StringLen < MaxLen - 1 ) { // solange einlesen bis abschließender "." kommt

\*Buffer++ = NextChar;

StringLen++;

NextChar = uart\_getc();

}

// Noch ein '\0' anhängen um einen Standard

// C-String daraus zu machen

\*Buffer = '\0';

}

ISR(USART1\_RX\_vect) //Interrupt für UART empfang

{

eingang = uart\_getc;

if (eingang == '1') { // wenn variable für eingelesene zeichen "1" ist

PORTD &=~(1<<PD7); // led auf PD7 einschalten

}

if (eingang == '2') {

PORTD |= (1<<PD7);

}

if (eingang != '1' && eingang != '2') { // wenn taste ungleich 1 und ungleich 2 --> string prüfen

uart\_gets(Line, 40);

if (!(strcmp(Line,"ein"))) { // prüfen ob string passt; wenn string "ein" ist -> LED ein

PORTD &= ~(1<<PD7);

}

if (!(strcmp(Line,"aus"))) { // prüfen ob string passt; wenn string "aus" ist -> LED aus

PORTD |= (1<<PD7);

}

}

}

int main(void)

{

DDRD |= (1<<DDD7); //PB0 OUTPUT

PORTD |= (1<<PD7); // standard: led aus

CLKPR = 0x80;

CLKPR = 0x00; //CLK-Prescaler 1

init\_usart(); //USART initialisieren

sei(); //Interrupts global freigeben

UDR1 = 'C';

char text[]="\n\r\n\rWir sollten im labor sein...\n\r\n\r"; // string definieren

while(text[i]!='\0') {

while(!(UCSR1A & (1<<UDRE1))); //warten bis fertig

UDR1 = text[i]; //Zeichen and UDR senden

i++;

}

while(1);

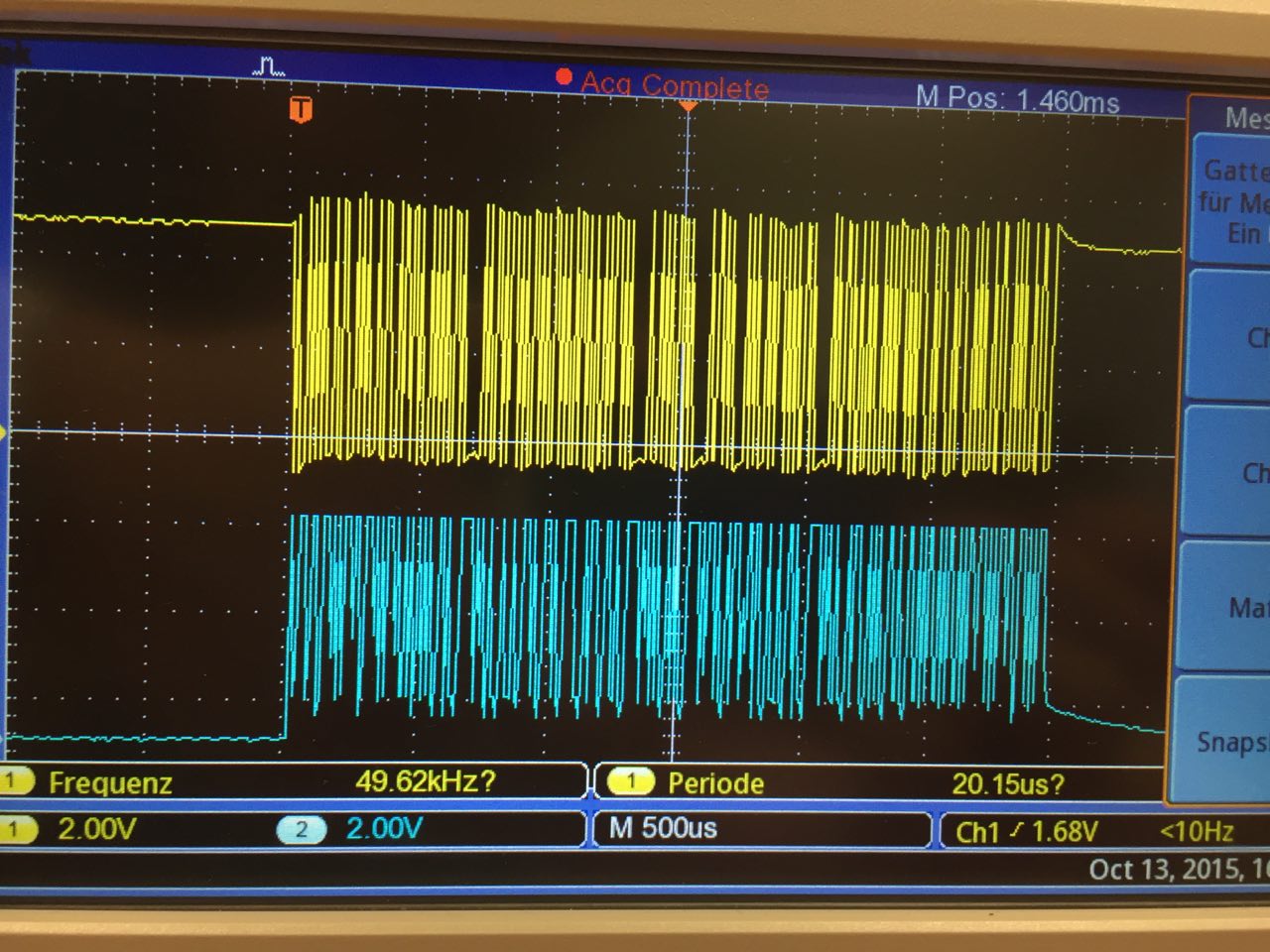
}

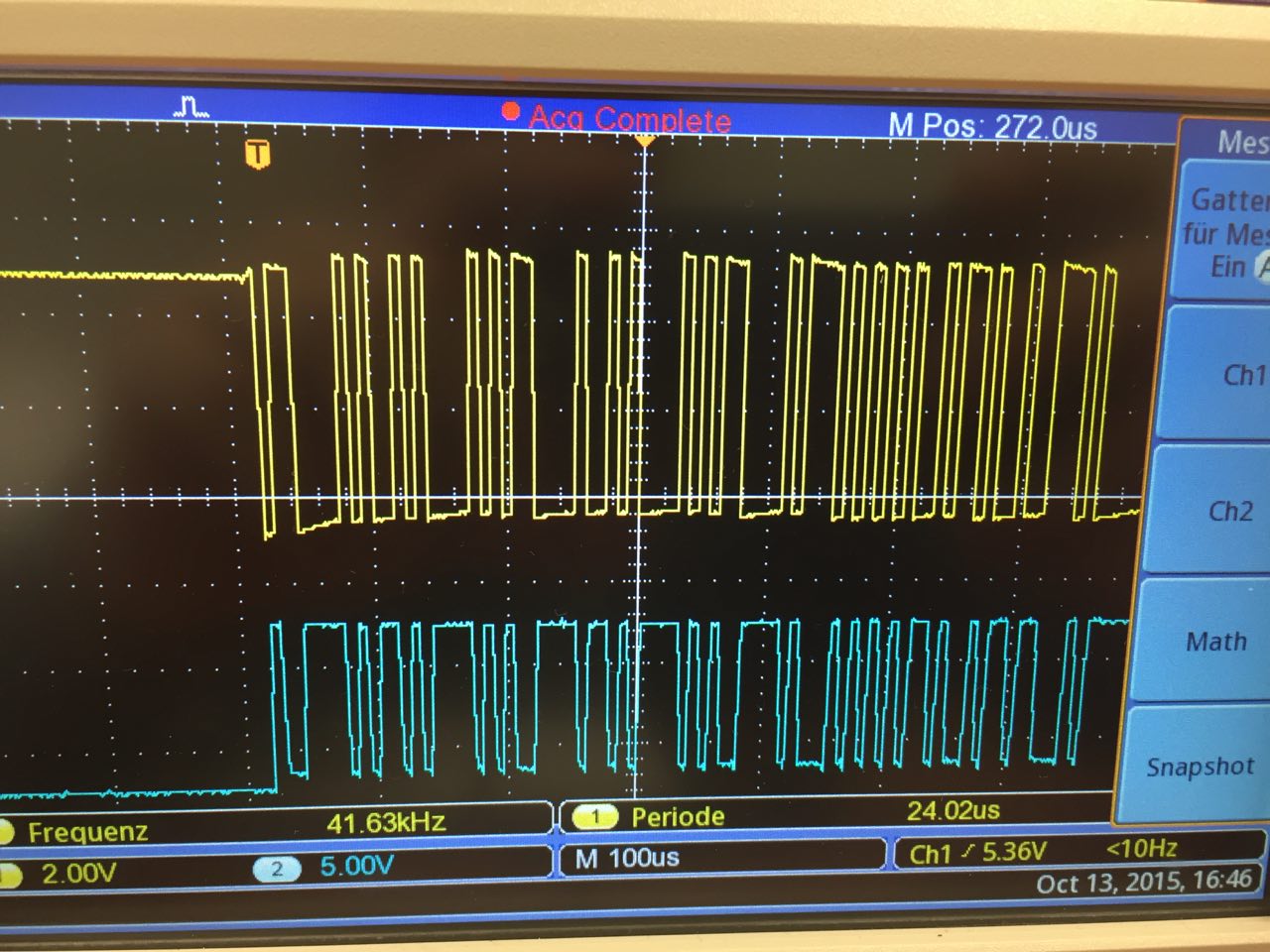
Listing 1. – Program zur Ansteuerung einer RS323 Verbindung

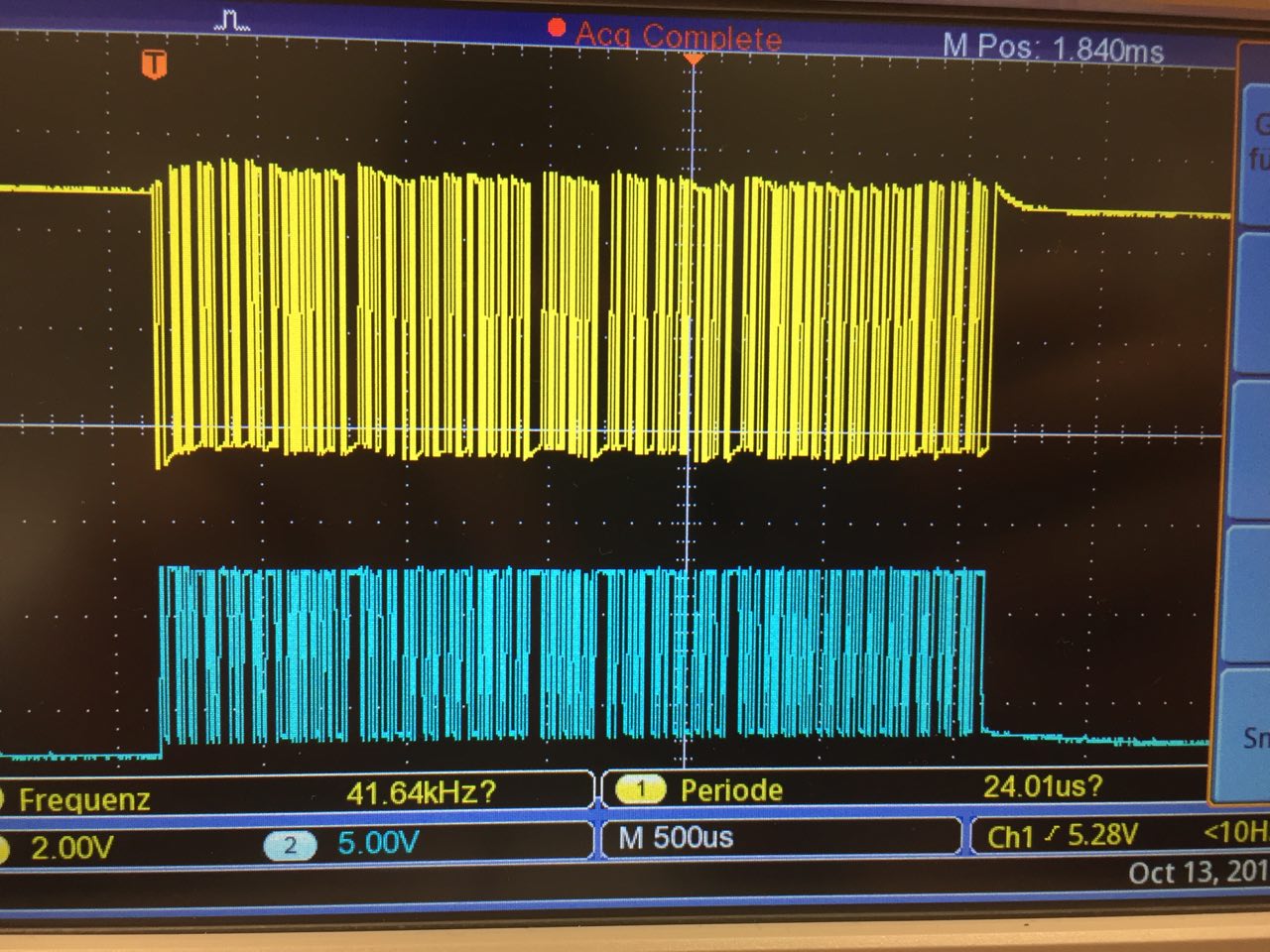
# Messungen

Bei dieser Messung ist am CH1 das serielle Signal vom µC zu sehen, und am CH2 das Signal mit den an die RS232-Schnittstelle angepassten Pegel.

Der µC liefert nur 0V und 5V, der MAX232 macht daraus +10V und -10V.







# Program-Listing Verzeichnis

Listing 1. – Program zur Ansteuerung einer RS323 Verbindung 5