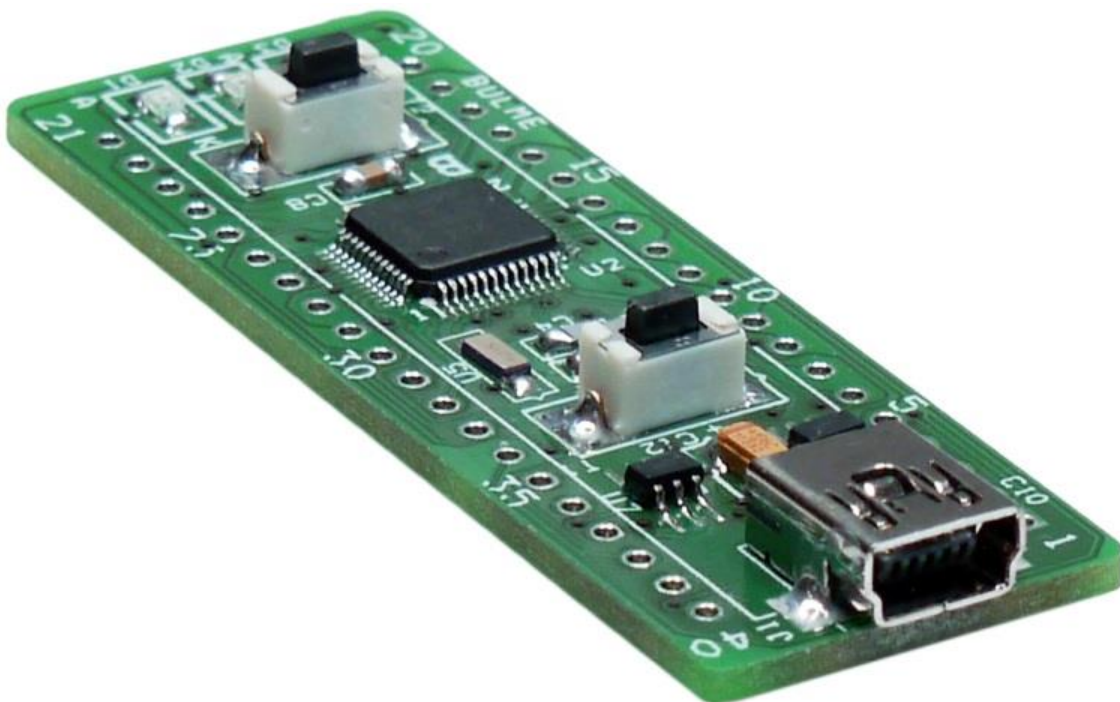


HTL - BULME GRAZ, Cortex M0+, Microboard

Dezember 10

2017

Das Microcontrollerboard M0+ Microboard ist an der BULME Graz, Abteilung Elektronik und Technische Informatik, für den Unterricht im fachtheoretischen und fachpraktischen Bereich entwickelt worden. Die Bauform DIL40 ermöglicht einen universellen Einsatz an diversen Steckboards oder Prototypen.



Inhaltsangabe

1	Das Board	4
1.1	Allgemeines	4
1.2	Anschlussbelegung	5
1.3	Schaltplan	6
1.3.1	Microcontroller.....	6
1.3.2	Taktversorgung	8
1.3.3	USB Anschluss.....	8
1.3.4	USB/UART.....	9
1.3.5	RESET / BOOT	10
1.3.6	Längsregler 3,3 V	11
1.3.7	LIPO Lademanagement.....	11
1.3.8	Externe Kontakte	12
1.4	Bestückungsplan	13
1.5	Layout	14
1.6	Stückliste	15
2	Programmbeispiele.....	16
2.1	Digital-Out - Blinky	16
2.2	4 Bit Laufflicht.....	18
2.3	4 Bit Zähler	19
2.4	4 Bit Zähler mit beliebiger Zählfolge	20
2.5	1 Hz – Ticker	22
2.6	UART – „Hello world“	23
2.6.1	Allgemeines	23
2.6.2	Programm.....	24
2.6.3	Ausgabe	24
2.7	Einfacher Taschenrechner	25
2.7.1	Programm.....	25
2.7.2	Visualisierung	26

2.8	Zeichenumkehr	27
2.8.1	Programm.....	27
2.8.2	Anzeige	28
2.8.3	Spezielle Zeichendarstellung	29
2.8.4	Anzeige	30
2.9	Elektronischer Würfel	31
2.9.1	Allgemeines	31
2.9.2	Schaltung	31
2.9.3	Programm.....	33
2.9.4	Ausgabe	34
2.10	PWM.....	35
2.10.1	Allgemeines	35
2.10.2	Beispiel.....	35
2.10.3	Anzeige	36
2.11	2 Kanal-PWM.....	37
2.12	Analog-Digital-Umsetzer	38
2.12.1	Allgemeines	38
2.12.2	Portdefinition.....	39
2.12.3	Einlesen einer Analogspannung.....	39
2.13	Spannungsmessung an einem Potentiometer.....	40
2.13.1	Allgemeines	40
2.13.2	Programm.....	40
2.13.3	Ausgabe	41
2.14	Temperaturmessung	42
2.14.1	Messung mit Analogsensor LM235.....	42
2.14.2	Messung mit NTC-Sensor	45
2.14.3	Beispiel NTC.....	49
2.14.4	Vergleich NTC vs LM235	51
2.15	RTC.....	53
2.15.1	Allgemeines	53
2.15.2	Datum und Uhrzeit.....	53
2.15.3	Ausgabe	54

2.15.4	Datum und Uhrzeit stellen.....	55
2.15.5	Individuelle Zeit und Datum Ein-Ausgabe.....	57
2.16	Timer	59
2.16.1	Allgemeines	59
2.16.2	Messung der Dauer einer UART Ausgabe.....	60
2.16.3	Messung einer PWM Spannung	62
3	<i>Anhang</i>	66
3.1	<i>Abbildungsverzeichnis</i>	66
3.2	Tabellenverzeichnis.....	67
3.3	Literaturverzeichnis	68
3.3.1	Internetquellen.....	68
3.3.2	Bücher	68

1 Das Board

1.1 Allgemeines

Das Steuermodul ist an der BULME GRAZ, Abteilung Elektronik und Technische Informatik entwickelt worden. Die Hauptanwendung ist der Einsatz im fachtheoretischen und fachpraktischen Unterricht an der HTL. Die Abmessungen sind auf das genormte Maß eines DIL-40 Gehäuses gelegt worden, damit ist ein universeller Einsatz in der Hardwareentwicklung gegeben.

Als Anschlüsse werden neben den Versorgungsleitungen und USB auch die freien Portleitungen des Microcontrollers verwendet. Die USB-Schnittstelle wird sowohl für die Programmierung als auch für die UART-Schnittstelle verwendet. Für einen Betrieb mit LIPO-ACCU ist bereits ein Lademanagement integriert.

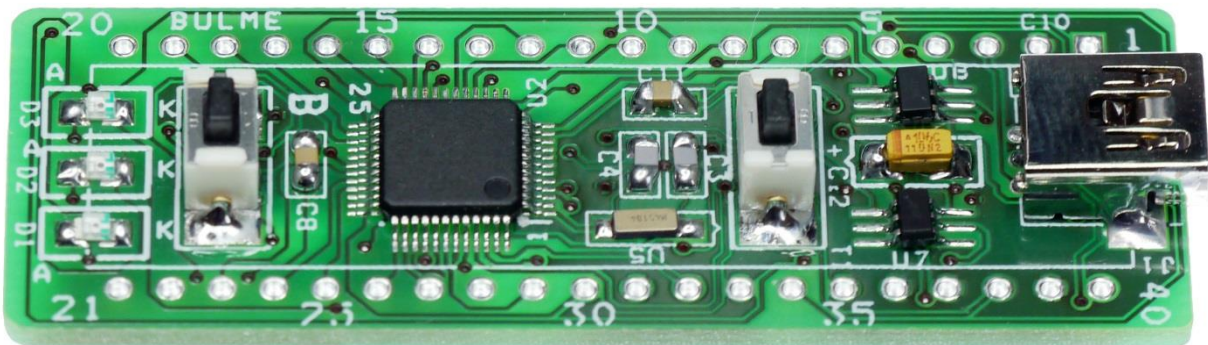


Abbildung 1: Oberseite des Microboards © Foto Schönauer (SHO)

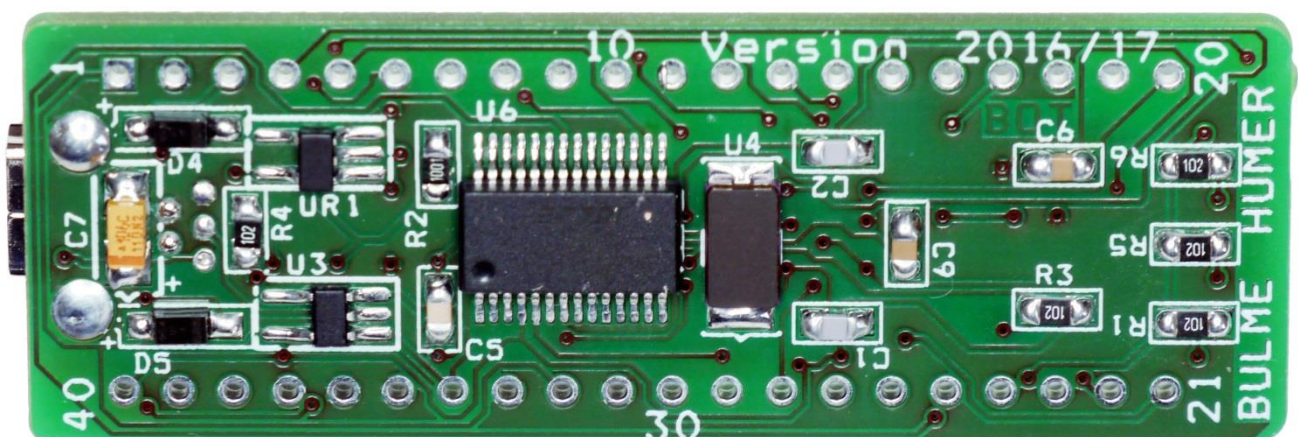
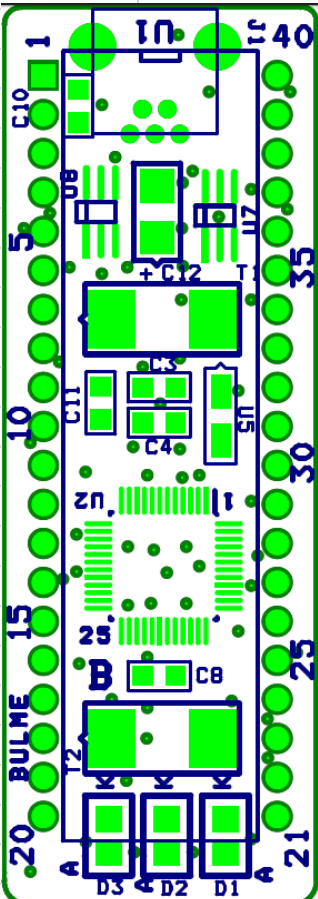


Abbildung 2: Unterseite des Microboards © Foto Schönauer (SHO)

1.2 Anschlussbelegung

				CORTEX M0+							
				LPC 11U68/48							
Debug	Alt. Fkt.	Port	PIN					PIN	Port	Alt. Fkt.	Debug
	Akku	VB	1					40	3V3		
		GND	2					39	GND		
	Ausgang	Vcc	3					38	RES#	Ausgang	
		P2.5	4					37	GND		
	RxD_2	P0.20	5					36	GND		
reserviert	CLK_out	P0.2	6					35	GND		
	PWM	P2.2	7					34	Vcc	5V	
		P1.20	8					33	GND		
OpenDrain	SCL_0	P0.4	9					32	VB	Akku	
OpenDrain	SDA_0	P0.5	10					31	GND		
		P0.21	11					30	P1.13	PWM	
	TxD_2	P1.23	12					29	P0.17		
	PWM	P2.7	13					28	P0.22	ADC11	
	SDA_1	P1.24	14					27	P0.16	ADC2	
		P0.6	15					26	P0.15	ADC3	
	SCL_1	P0.7	16					25	P0.14	RxD_1	JTAG_TRST#
		P1.21	17					24	P0.13	TxD_1	JTAG_TDO
	MISO	P0.8	18					23	P0.12	ADC8	JTAG_TMS
	MOSI	P0.9	19					22	P0.11	ADC9	JTAG_TDI
JTAG_TCK	SCK	P0.10	20	21	P0.23	ADC1					

TxD RxD LIPO

Abbildung 3: Anschlussbelegung des Microboards

Im obigen Bild sind folgende Elemente dargestellt:

- Anschluss-PIN (Nummer der DIL40 Reihe)
- Portanschluss am Microcontroller
- Alternative Funktion der Portleitung des Microcontrollers
- JTAG – Anschluss für die Verwendung eines Debuggers
- LEDs D1,D2,D3 zur Darstellung der Signale Akkuladen, RxD und TxD vom UART

1.3 Schaltplan

1.3.1 Microcontroller

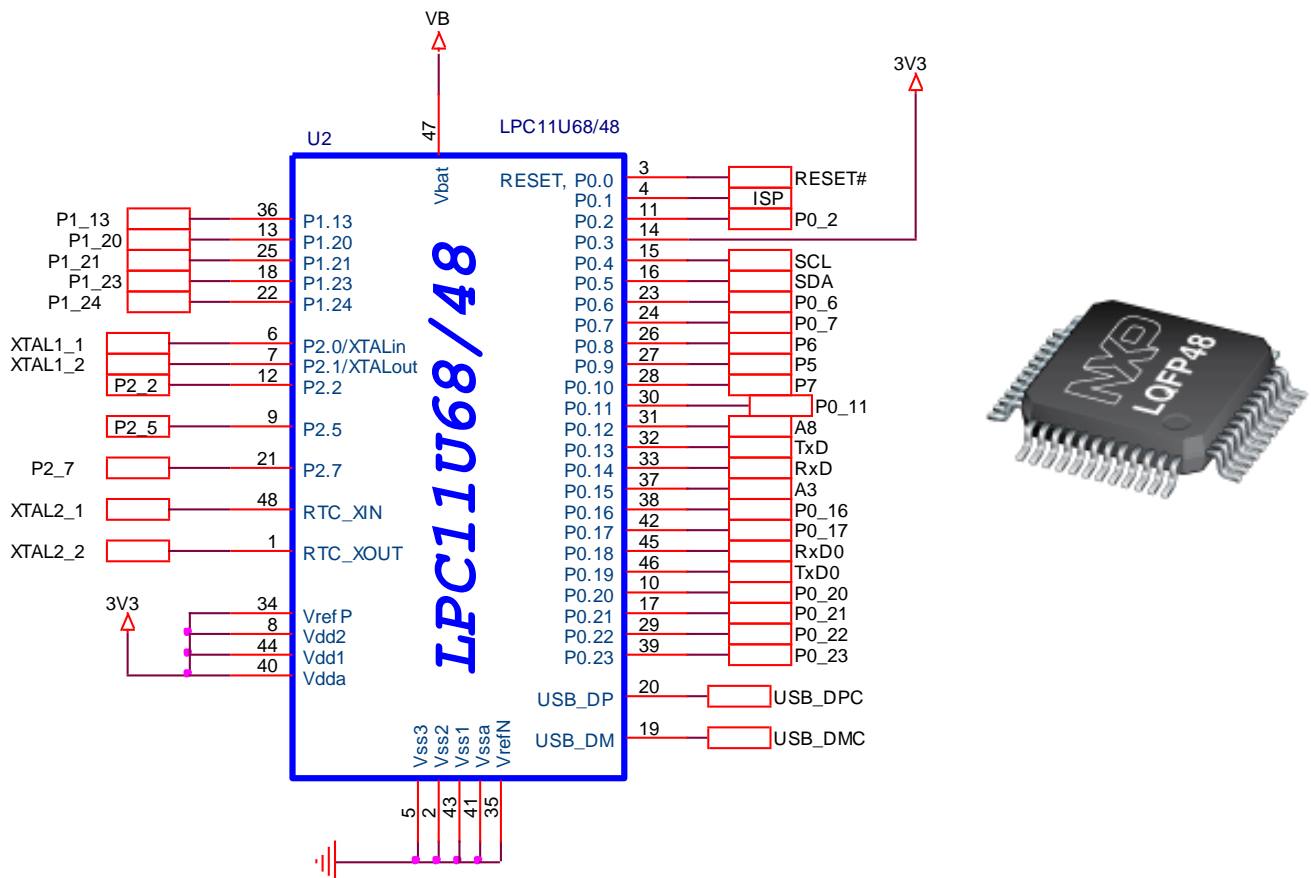


Abbildung 4: Der Microcontroller LPC11U68-48PIN mit Anschlussdefinitionen

Der Microcontroller der Firma NXP¹ hat einen Basiskern der Firma ARM² Cortex M0+. In dieser verwendeten Form kann er über USB, UART oder über JTAG³ programmiert bzw. geflashed werden.

Der Controller wird mit einer Taktfrequenz von 12 MHz versorgt und intern mit einer PLL⁴ auf 48 MHz erhöht. Der nutzbare Programmspeicher beträgt 256 KB, das nutzbare RAM⁵ 32 KB, ein EEPROM⁶ von 4 KB steht als permanenter Datenspeicher zur Verfügung. Wichtige Schnittstellen wie UART, I2C, SPI, Analog-Digital-Umsetzer mit einer Auflösung von 12 Bit, RTC und PWM bzw. zahlreiche I/Os sind implementiert. Eine Echtzeituhr (RTC) und ein Temperatursensor on Chip runden die Besonderheiten dieses Chips ab.

¹ www.nxp.com

² Acorn RISC Machines

³ Joint Test Action Group

⁴ Phase Locked Loop

⁵ Arbeitsspeicher, random access memory

⁶ electrically erasable programmable read-only memory

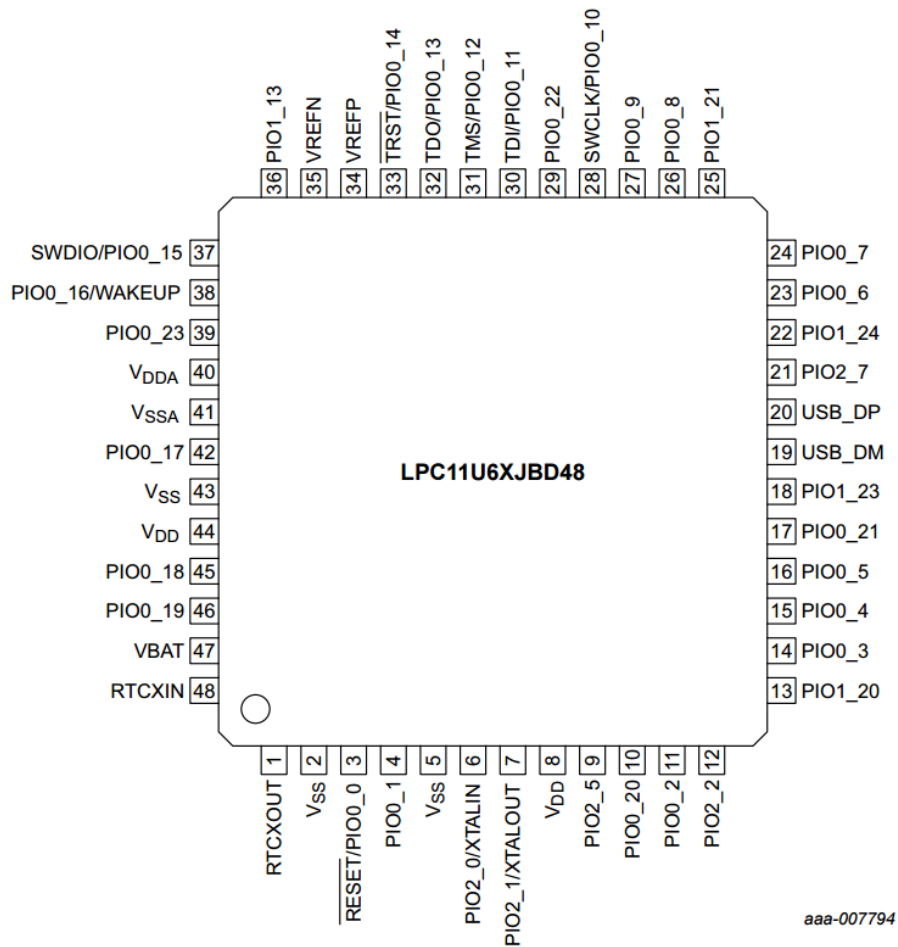


Abbildung 5: Microcontroller LPC11U68 - LQFP48 pinning

Type number	Flash/ KB	EEPROM/ KB	SRAM/ KB	USB	USART0	USART1	USART2	USART3	USART4	I ² C	SSP	Timers with PWM	12-bit ADC channels	GPIO
LPC11U66JBD48	64	4	12	1	Y	Y	Y	N	N	2	2	6	8	34
LPC11U67JBD48	128	4	20	1	Y	Y	Y	N	N	2	2	6	8	34
LPC11U67JBD64	128	4	20	1	Y	Y	Y	N	N	2	2	6	10	48
LPC11U67JBD100	128	4	20	1	Y	Y	Y	Y	Y	2	2	6	12	80
LPC11U68JBD48	256	4	36	1	Y	Y	Y	N	N	2	2	6	8	34
LPC11U68JBD64	256	4	36	1	Y	Y	Y	N	N	2	2	6	10	48
LPC11U68JBD100	256	4	36	1	Y	Y	Y	Y	Y	2	2	6	12	80

Tabelle 1: Lieferbare Typen des Microcontrollers

Das Microboard ist mit einem LPC11U68JBD48 bestückt.

1.3.2 Taktversorgung

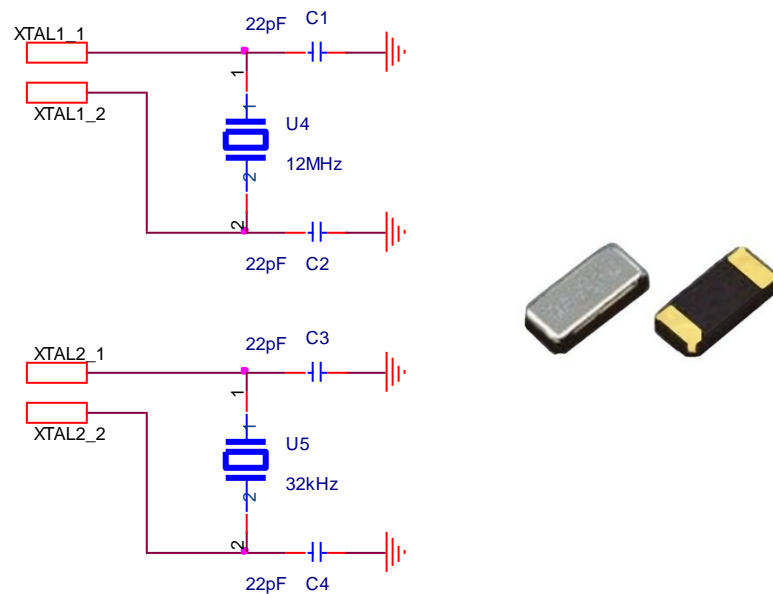


Abbildung 6: Taktversorgung für den Microcontroller und der Echtzeituhr (RTC)

Der Quarz U4 mit 12 MHz dient zur Taktgenerierung für den Microcontroller. Die bereits integrierte Echtzeituhr (RTC) arbeitet mit einer eigenen Spannung (VB). Für den Betrieb der RTC ist noch ein externer Uhrenquarz U5 mit einer Taktfrequenz von 32768 Hz notwendig. Die Genauigkeit der Echtzeituhr ist maßgeblich von der Genauigkeit und der Temperaturstabilität des Uhrenquarzes abhängig.

1.3.3 USB Anschluss

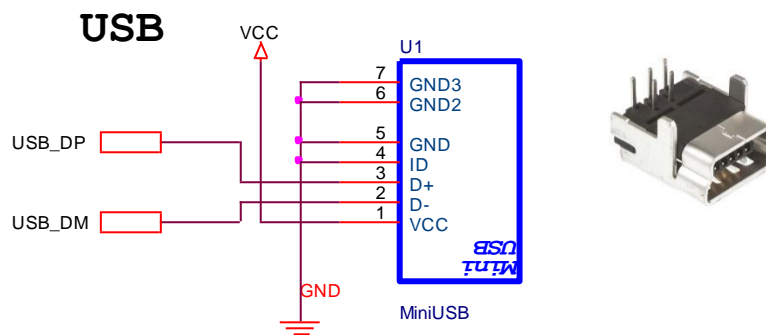


Abbildung 7: Pin-Belegung des USB-Mini Anschlusses

1.3.4 USB/UART

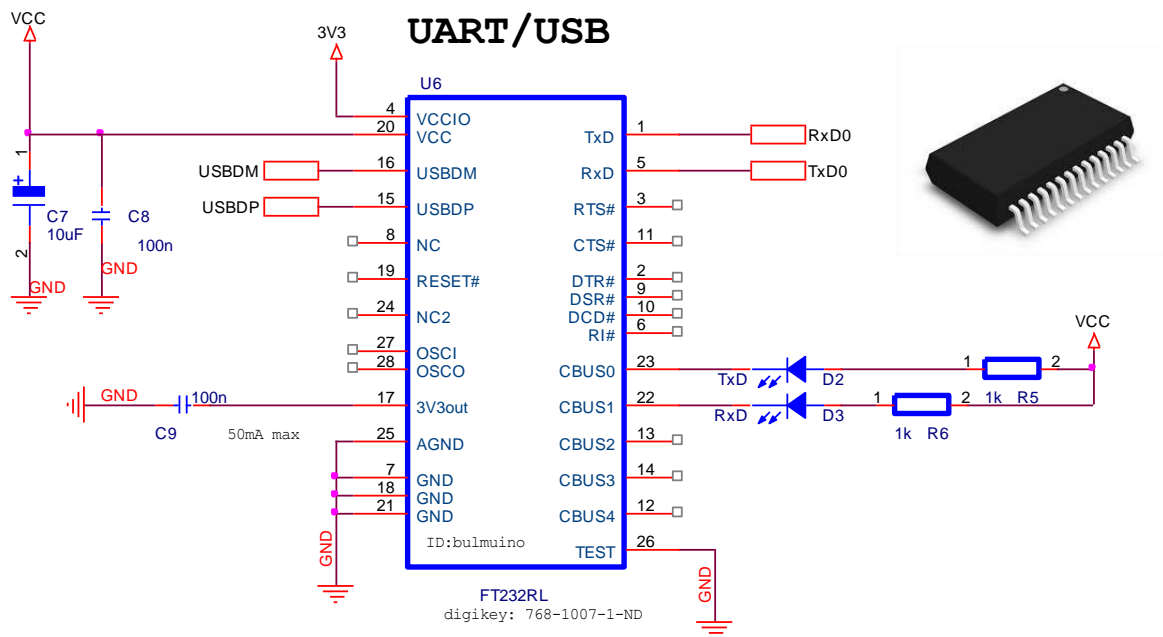


Abbildung 8: USB-UART-Controller der Firma FTDI

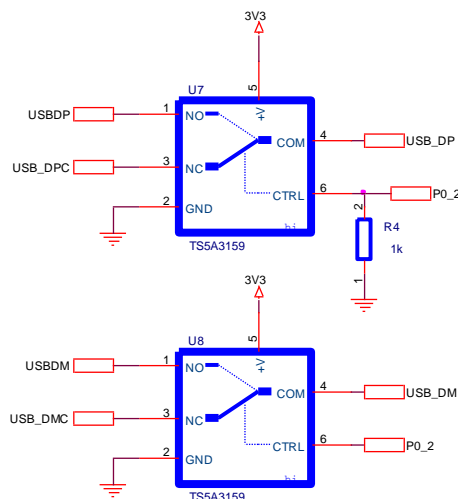


Abbildung 9: Umschaltung UART-USB durch den PortPin P0.2

Um in der jeweiligen Anwendung nicht 2 USB Buchsen (einmal USB und dann UART) verwenden zu müssen ist ein Umschalter (Analogschalter) verwendet worden. Im Zustand nach dem RESET ist der USB Connector direkt mit dem USB Eingang des Microcontrollers verbunden. Der Widerstand R4 sorgt für ein LO am Control-Eingang von U7 und U8. Durch die Aktivierung des BOOT-Modus kann direkt über die USB Schnittstelle programmiert werden.

Während der Betriebsphase RUN kann durch den PortPin P0.2 (High-Pegel) die UART Schnittstelle an die USB Buchse angekoppelt werden. Durch die Software können dann sehr einfach über den „printf“-Befehl Daten an einen Host weitergeleitet werden. Siehe Kapitel „UART“.

1.3.5 RESET / BOOT

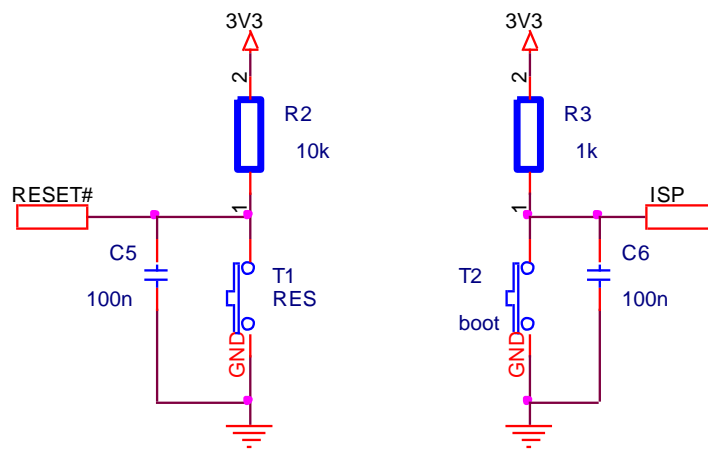
RESET / BOOT

Abbildung 10: Schaltung für RESET und BOOT (für Programmdownload)

Um das Board in den BOOT-Modus zu bringen, ist es notwendig das Board zuerst an den PC über ein USB-Mini-Kabel anzuschließen. Als nächstes wird die Taste T2 gedrückt und lässt man sie gedrückt. Im Anschluss dann kurz die Taste T1 drücken. Taste T2 kann dann wieder losgelassen werden.

Je nach PC kann es etwas dauern, bis das Betriebssystem einen USB-Stick erkennt und dem Microcontrollerboard einen Laufwerksbuchstaben zuweist. Weiteres im Kapitel „Programm downloaden“ (www.mbed.org).

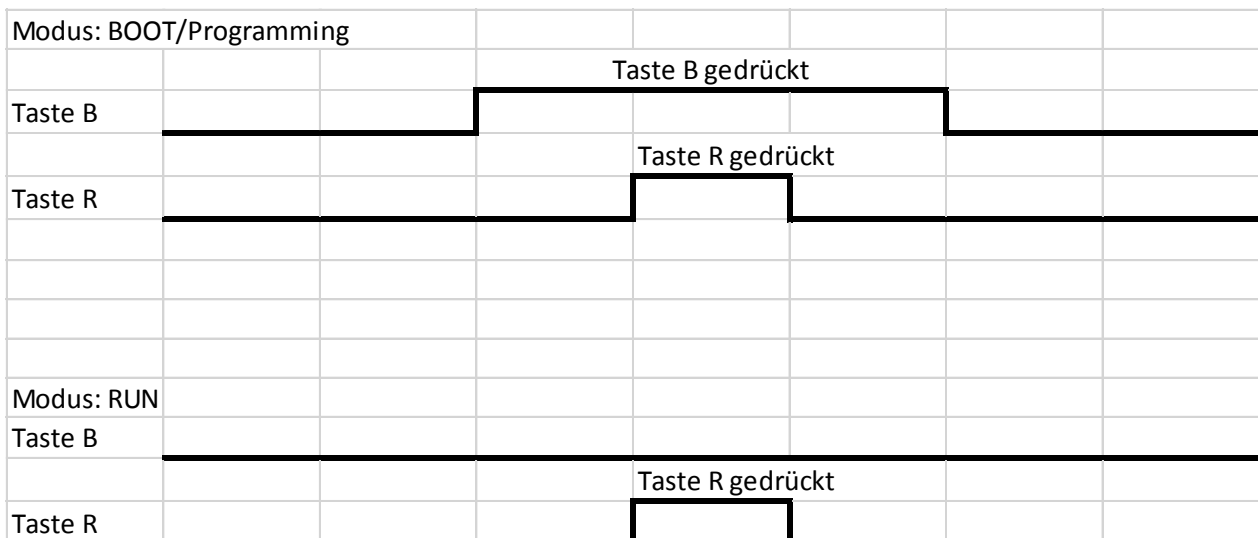


Abbildung 11: Tastenfunktion für RESET und BOOT (für Programmdownload)

1.3.6 Längsregler 3,3 V

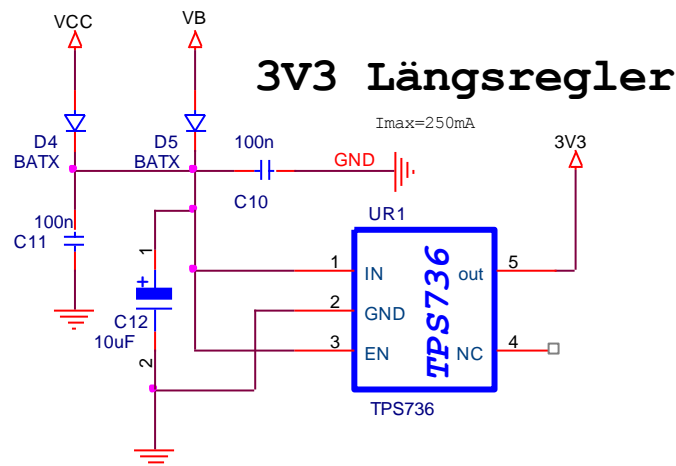


Abbildung 12: Schaltung für den 3,3 V Regler

Auf diesem Board wird der Baustein TPS736-3.3 verwendet. Dieser Baustein zeichnet sich durch einen kleinen Längsspannungsabfall (ca. 120 mV) aus und er kann damit auch bei einer Versorgung mit einer LIPO Zelle eine Ausgangsspannung mit 3.3 V liefern, wenn die Zellenspannung auf 3,5 V abgesunken ist. Die maximale Stromentnahme ist dabei 250 mA. Die 2 Schottky Dioden D4 und D5 erlauben eine universelle Speisung von Akku (3,7 V) und USB (VCC).

1.3.7 LIPO Lademanagement

Akku Ladeschaltung

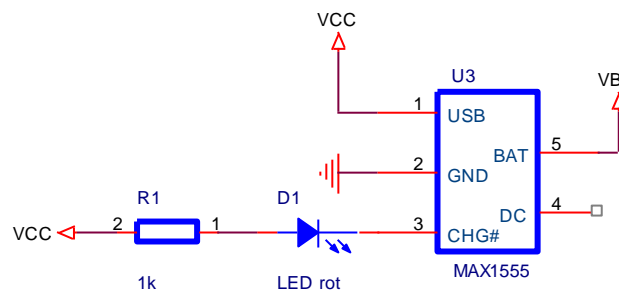


Abbildung 13: Schaltung für das Lademanagement einer LIPO-Zelle.

Der Baustein U3 steuert den Ladestrom einer Zelle mit einer maximalen Ladespannung von 4,2 V. Ein entsprechender Tiefentladungsschutz ist dabei nicht integriert.

Der Ladezyklus wird durch die Leuchtdiode D1 visualisiert.

1.3.8 Externe Kontakte

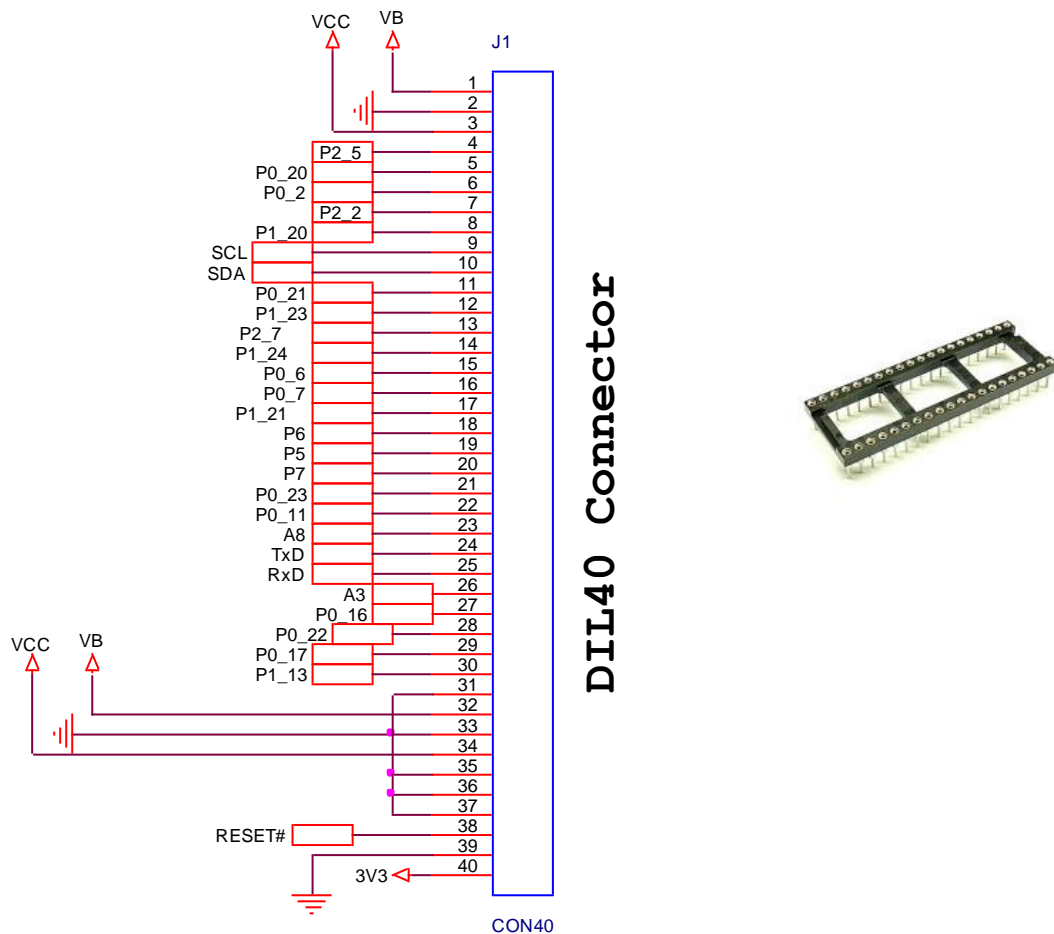


Abbildung 14: Anschlussbelegung des DIL40 – Connectors

Bedeutung folgender Anschlüsse:

Vcc	5 V, verbunden mit der USB-Buchse
VB	Eingangsspannung einer Lithiumzelle (3,7 V)
PX_X	jeweiliger Portanschluss des Microcontrollers laut Datenblatt
PX	Nomenklatur entsprechend mbed (www.mbed.org)
GND	Masseanschluss
RESET#	Reset-Anschluss (LOW-active) (Ausgang)

1.4 Bestückungsplan

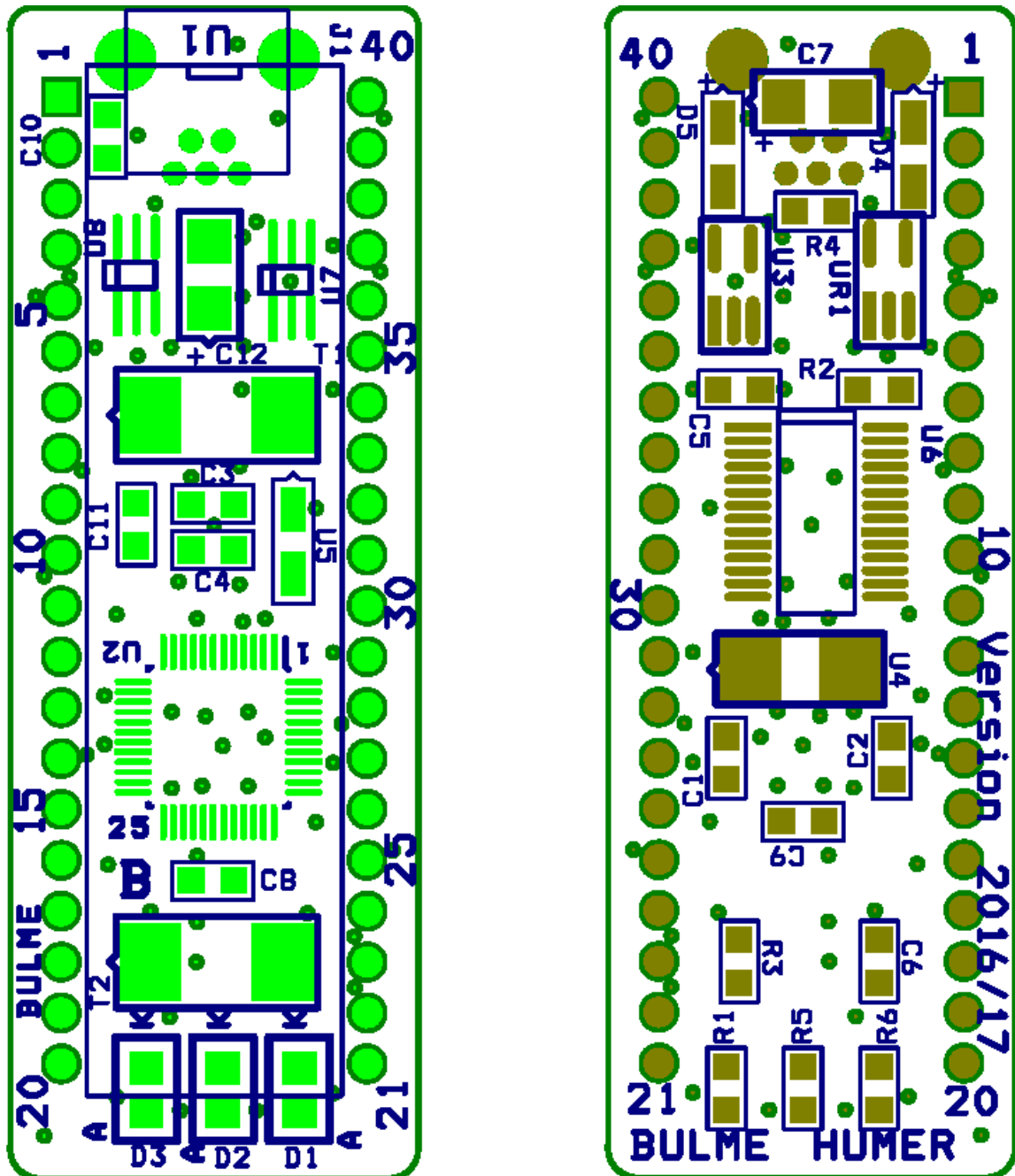


Abbildung 15: Bestückungsplan oben und unten (nicht maßstabsgetreu)

1.5 Layout

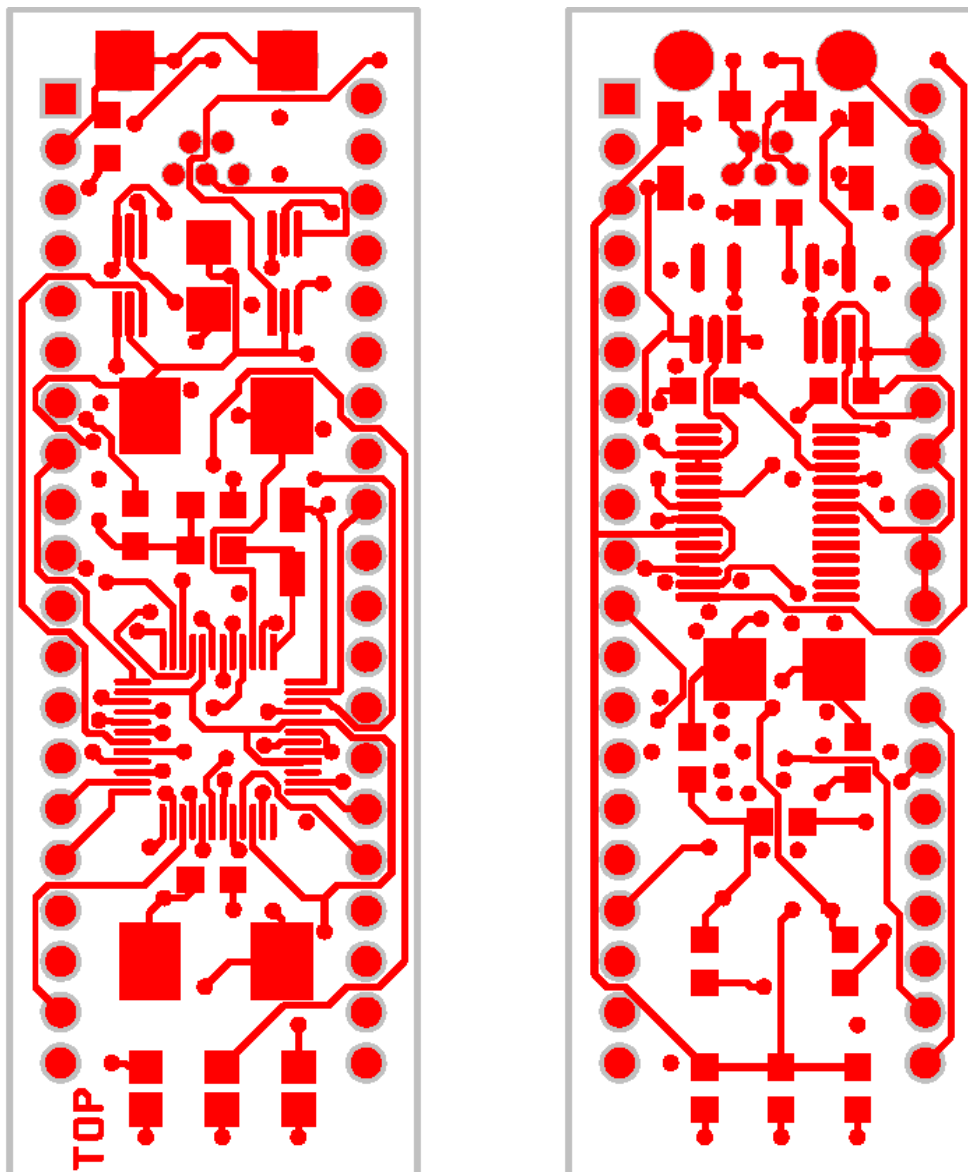


Abbildung 16: Darstellung des Layouts oben und unten, ohne Masseflächen (nicht maßstäblich)

Die doppelseitige Leiterplatte hat eine Größe von 50x30 mm. Sie wird durchkontaktiert, mit doppelseitiger Lötstopmmaske und Bestückungsdruck geliefert. Der Preis ist stark von der Stückzahl abhängig und bewegt sich etwa um €1,-. Die Pads⁷ sind um etwa 25 % größer definiert, damit ist ein Löten von Hand leichter möglich.

Einige Beispiele für Leiterplattenlieferanten:

www.pcb-pool.com

www.multi-circuit-boards.eu

www.seeedstudio.com u.a.

⁷ Footprint = Lötflächen der einzelnen elektrischen Anschlüssen

1.6 Stückliste

Pos.	Stk.	Benennung	Bezeichnung	Gehäuse	Schaltplan	RS-Nr.	Preis
1	1	Microcontroller	LPC11U68/48	LQFP48	U2;	kA	kA
2	1	Lith. Lader, Maxim	MAX1555	SOT-23	U3;	732-8730	1,02
3	1	Spannungsregler 3V3	TPS736-33	SOT-23	UR1;	102-5594	1,03
4	2	Analog Switch	TS5A3159	SOT-23	U7; U8;	662-2808	0,45
5	1	UART/USB	FT232RL	SSOP	U6;	406-580	2,95
6	1	LED, rot, 100 mcd	rot	0805	D1;	466-580	0,05
7	2	LED	gelb	0805	D2; D3;	466-3835	0,10
8	3	SMD-Widerstand	1k	0805	R1; R3; R4;	Rolle – 4k	
9	1	SMD-Widerstand	10k	0805	R2;	Rolle – 4k	
10	4	Kondensator	12pF	0805	C1;C2; C3; C4;	Rolle - 4k	
11	2	Kondensator	100nF	0805	C5; C6; C8; C9; C10; C11,	Rolle – 4k	
12	2	Elko	10uF	0805	C7;C12	Rolle – 4k	
13	2	Schottky Diode 30V	0,5A - BATXX	0805	D4; D6;	545-3291	0,15
14	1	Mini-USB Buchse PCB	90 Grad	USB-B	U1;	515-2005	1,02
15	1	Quarz	12MHz	SMD-2	U4;	703-1947	0,95
16	1	Quarz	32,768kHz	SMD-2	U5;	727-6279	0,76
17	2	Mini Taster	1 polig	SMD	T1; T2;	378-6325	0,21
18	1	PCB (Platine)	50x13mm	2 Lagig		kA	1,0

Tabelle 2: Stückliste, RS-Nr. bedeutet Bestell-Nr. der Firma RS-Components (BBG Lieferant)

Stückliste, RS-Nr. bedeutet Bestellnr. der Firma RS-Components (BBG⁸ Lieferant)

Anmerkungen zu genannter Position:

ad1) Der Microcontroller ist über www.mouser.at, www.digikey.at, www.farnell.at etc. erhältlich. Die Kosten liegen dabei etwa zwischen 2 und 4 Euro (abhängig von der Stückzahl und Transportkosten).

ad 8+9+10+11+12) Rollenware, 4000 Stück pro Rolle, Preis wenige Euro

⁸ Bundes Beschaffungsgesellschaft

2 Programmbeispiele

2.1 Digital-Out - Blinky

In diesem Beispiel soll eine LED im Sekundentakt blinken. Die Verwendung eines kostengünstigen Protoboards (auch Steckboard oder Steckbrett genannt) erleichtert den Aufbau.

Materialbedarf:

1x M0+ Microboard

1x Protoboard mit 3 Verbindungskabel

1x Widerstand, Wert zwischen 75 und 220 Ω ($75 \leq R1 \leq 220 \Omega$)

1x Leuchtdiode (z.B. rot)

Die Helligkeit der Leuchtdiode wird durch den durchfließenden Strom bestimmt, der durch die Widerstandsgröße von R1 bestimmt wird. Die Durchflussspannung einer roten LED beträgt 1,8V.

$$I = \frac{3,3 - 1,8 \text{ [V]}}{R1 \text{ [\Omega]}} \text{ [A]}$$

Für einen Widerstand von 75 Ω ergibt sich ein Strom von 20 mA, für einen Wert von 220 Ω stellt sich ein Strom von 6,8 mA ein.

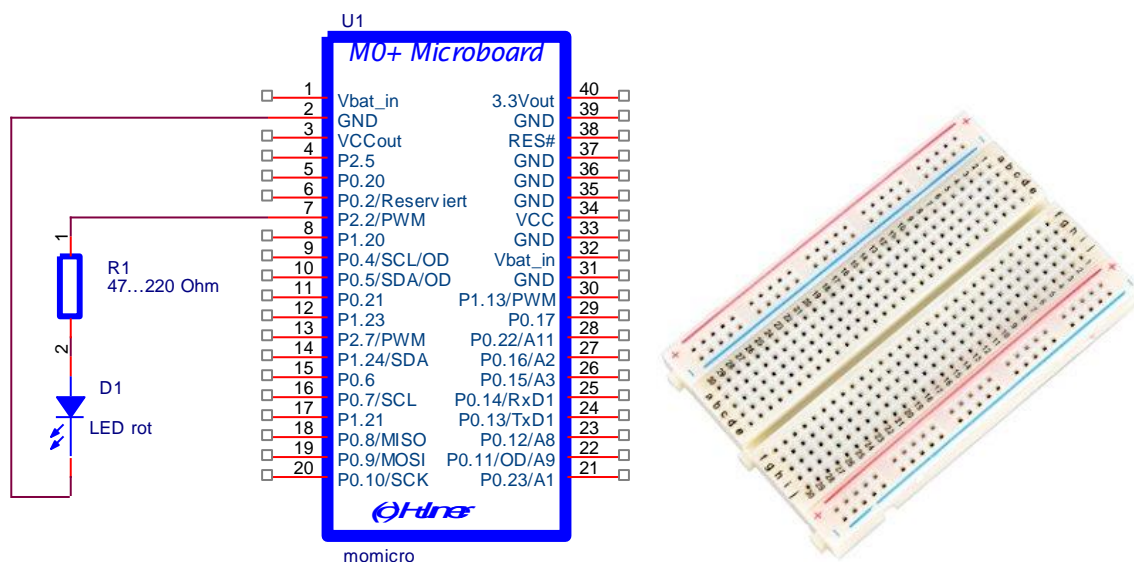


Abbildung 17: Schaltplan für den Aufbau auf einem Protoboard, R1 und D1 werden laut Schaltplan verdrahtet.

Die Portleitungen an einem Microcontroller können unterschiedlich konfiguriert werden. In diesem Beispiel wird die Portleitung P2.2 als Ausgang definiert.

Bei der Ausgangskonfiguration wird dabei der Mode „push pull“ verwendet. Dabei hat hi den Pegel 3,3 V und low den Pegel 0 V.

```

/* ***** */
/* ***** LED Blinklicht mit der Periode von 1 Hz ***** */
/* **** BULME Graz, Elektronik und Technische Informatik, Humer **** */

#include "mbed.h"          // Einbindung - Library mbed
DigitalOut myled(P2_2);   // Portleitung P2.2 ist ein Ausgang

int main() {
    while(1) {            // Endlosschleife
        myled = 1;        // LED = on, Ausgang hat 3,3 V
        wait(0.5);        // Warte 0,5 Sekunden
        myled = 0;        // LED = off, Ausgang hat 0 V
        wait(0.5);        // Warte 0,5 Sekunden
    }                     // Ende Endlosschleife
}                         // end main

```

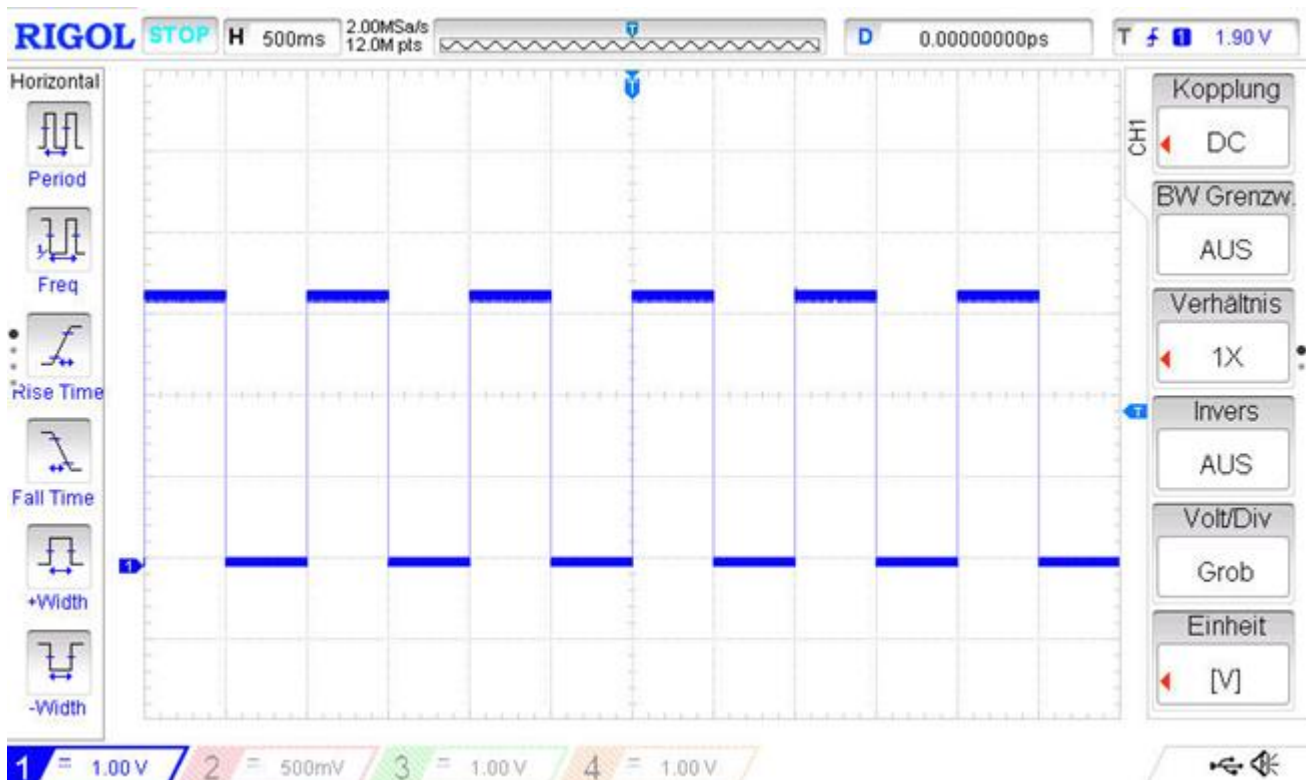


Abbildung 18: Darstellung des Ausgangssignals P2.2 an einem Oszilloskop

Am Signalverlauf (x-Achse = 500 ms/Div. und y-Achse = 1 V/Div.) lässt sich der Sekundentakt mit den Spannungspegeln 0 und 3,3 V erkennen.

2.2 4 Bit Lauflicht

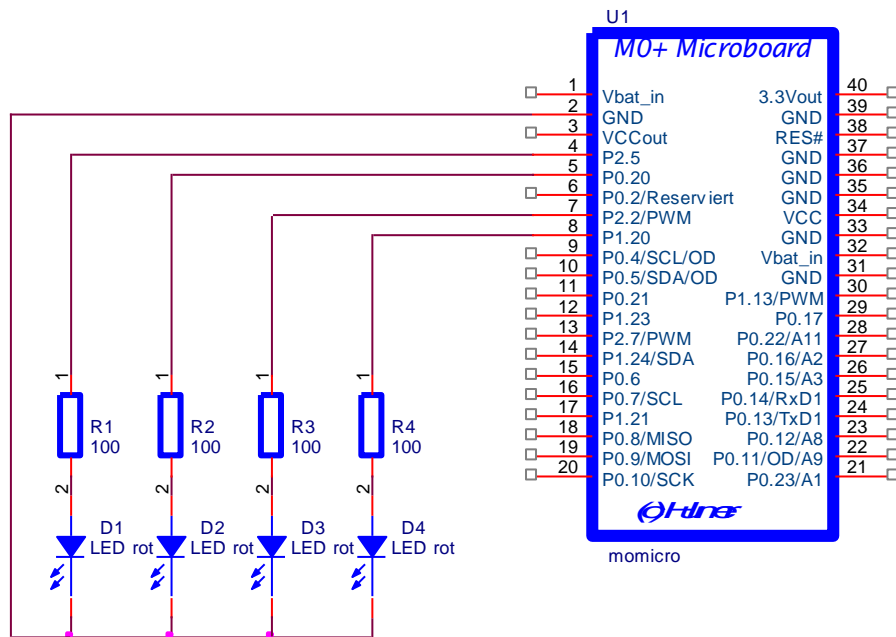


Abbildung 19: Aufbau eines 4 Bit Lauflichtes am Protoboard

Als Ausgänge werden die Portleitungen P2.2, P1.20, P0.4 und P0.5 verwendet. Die einzelnen Portleitungen müssen dabei aber nicht einzeln als DigitalOut, sondern gleich als BusOut definiert werden. Dies bedeutet, dass mehrere Portleitungen als ein Bus definiert und implementiert werden können (hier 4 Leitungen oder 4 Bit).

```

/* ***** */
/* *** 4 Bit Lauflicht mit Bus-Konfiguration der Portleitungen ***** */
/* **** BULME Graz, Elektronik und Technische Informatik, Humer **** */

#include "mbed.h"
BusOut myleds(P2_5,P2_20,P2_2,P1_20); // Definition der 4 LEDs als Bus

/* ***** Variablendefinition ***** */
int i; // Definition der Variable i, Integer

/* ***** Hauptprogramm ***** */
main()
{
    while(1) // Endlosschleife
    {
        myleds=1<<i; // Daten um 1 Bit nach links schieben
        wait(0.2); // warte 200msec
        i++; // Variable i um den Wert 1 erhöhen
        if(i==4) i=0; // Wenn i=4, dann Rücksetzen
    } // end while
} // end main

```

2.3 4 Bit Zähler

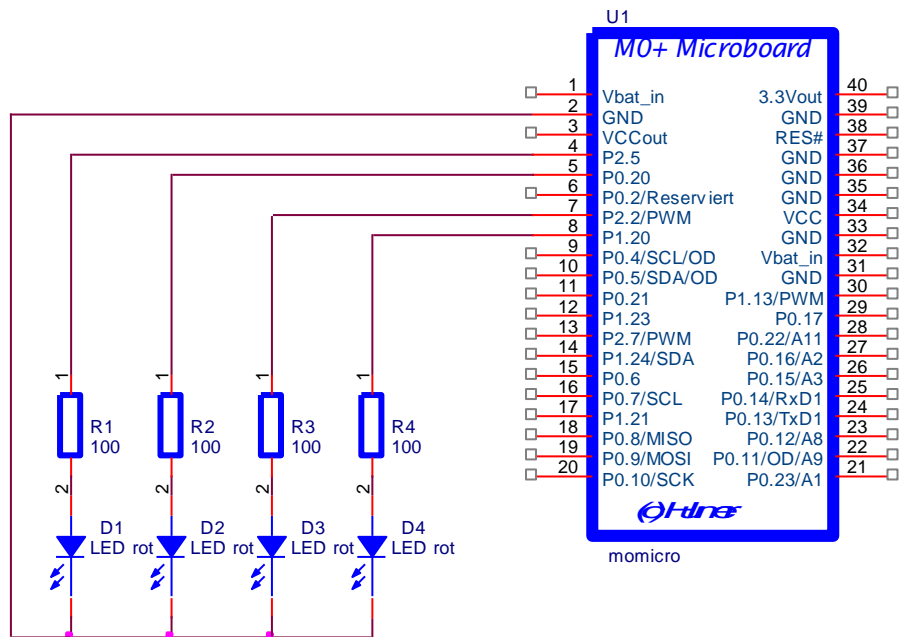


Abbildung 20: Aufbau eines 4 Bit Zählers am Protoboard

Die Zählfolge soll dabei 0,1,2,3,4,5,6,7,0.....(modulo 8) sein.

```

/* ***** */
/* ***** 4 Bit Zähler mit Bus-Konfiguration der Portleitungen ***** */
/* ***** BULME Graz, Elektronik und Technische Informatik, Humer ***** */

#include "mbed.h"
BusOut myleds(P2_5,P2_20,P2_2,P1_20); // Definition der 4 LEDs als Bus

/* ***** Variablendefinition ***** */

/* ***** Hauptprogramm ***** */
main()
{
    myleds=0; // Zählerstand auf 0 setzen
    while(1) // Endlosschleife
    {
        wait(0.5); // warte 500 msec
        myleds++; // Variablenwert um 1 erhöhen
        if(myleds==8) myleds=0; // Wenn myleds=8, dann Rücksetzen
    } // end while
} // end main

```

2.4 4 Bit Zähler mit beliebiger Zählfolge

In diesem Beispiel soll ein Zähler mit folgender Zählfolge realisiert werden.

Zählfolge: 0,3,5,1,6,7,2,4,15,0.....

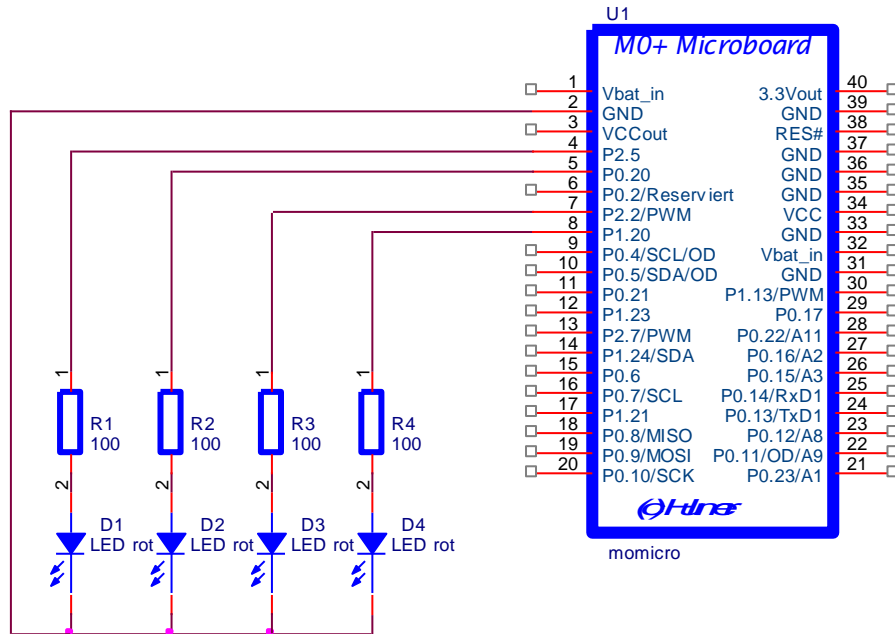


Abbildung 21: Aufbau der Schaltung am Protoboard

```

/* ***** */
/* ***** Zähler mit folgender Abfolge: 0,3,5,1,6,7,2,4,0,15 ***** */
/* ***** */

#include "mbed.h"
BusOut myleds(P2_5,P2_20,P2_2,P1_20); // Definition der 4 LEDs als Bus

/* ***** Variablendefinition ***** */
char tabelle[9]={0,3,5,1,6,7,2,4,15}; // Tabelle der Abfolge
int i=0; // Variable i

/* ***** Hauptprogramm ***** */
int main() // Beginn Hauptprogramm
{ // Beginn Endlosschleife
    while(1) // Tabellenaufruf 0<=i<=7
    { // Warte 200 ms
        myleds = tabelle[i]; // Erhöhe den Wert von i um 1
        wait(0.2); // Rücksetzen wenn i = 9
        i++;
        if(i==9) i=0;
    } // end while
} // end main

```

Ausgangspegel (4 Bit) im 200 ms Zeitraster:

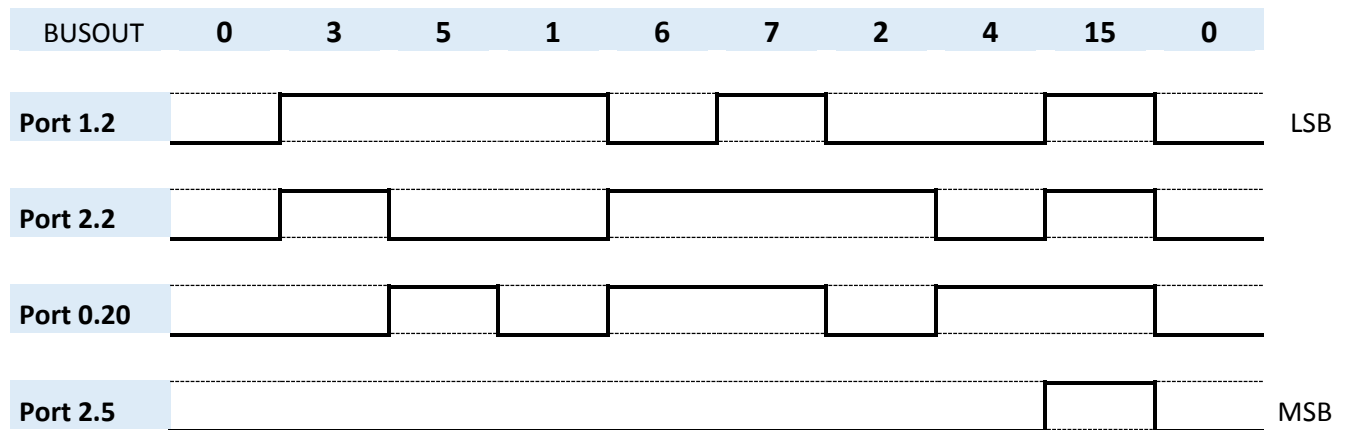


Abbildung 22: Ausgangspegel (4 Bit) im 200ms Zeitraster:

Im obigen Diagramm ist der jeweilige Pegel an den Ausgängen (P0.5 ... P2.2) im Zeitraster 200 ms dargestellt. Durch die 4 Bit am Ausgang sind 16 Werte möglich, hier sind nur 9 verwendet worden. Die Verwendung einer Tabelle im Programm ist universell möglich.

2.5 1 Hz - Ticker

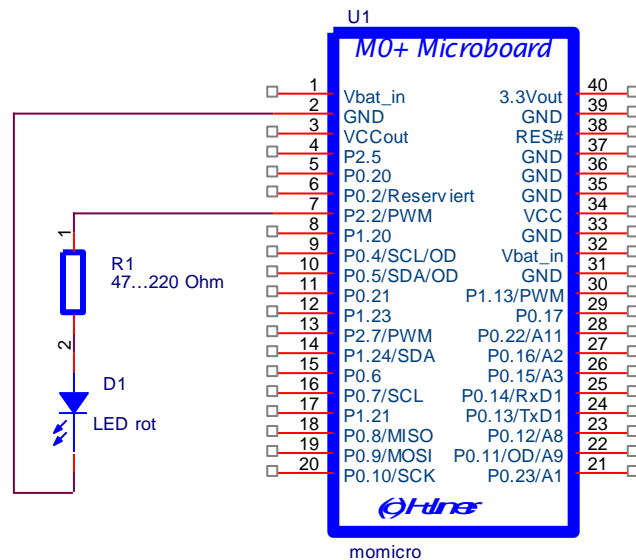


Abbildung 23: Aufbau am Protoboard, Blinklicht 1 Hz

```

/* ***** */
/* *****  Verwenden einer Tickerfunktion mit 500 ms Basis  ***** */
/* ***** */

#include "mbed.h"                // Library mbed

Ticker flipper;                 // Def. der Tickerfunktion mit Namen flipper
DigitalOut led1(P2_2);          // LED am Port P2.2

/* *****  Interruptfunktion  ***** */
void flip()                      // Interruptfunktion flip()
{
    led1 = !led1;                // Logisches Invertieren des Spannungspegels
}

/* *****  Hauptprogramm  ***** */
int main()
{
    led1 = 1;
    flipper.attach(&flip, 0.5); // Interruptaufruf alle 500 ms
    while(1);                    // Endlosschleife
}

```

Bei diesem Programm wird der interne Interrupt als Ticker verwendet. Die Interrupt-Funktion ist im Programmcode mit einem Rahmen hinterlegt. Die dazugehörige Definition des Tickers mit dem Namen „flipper“ ist am Programmbeginn definiert. Die Definition des Interrupts und der dazugehörige Funktionsaufruf ist im Hauptprogramm realisiert.

2.6 UART - „Hello world“

2.6.1 Allgemeines

Die Beispielsoftware sendet den Textstring „Hello world“ über die serielle Schnittstelle (UART) an den PC. Über einen virtuellen COM kann die Zeichenkette mit Hilfe eines Terminalprogramms auf dem Schirm dargestellt werden. Bei der Erstellung des Programms darf nicht auf die Umschaltung USB-UART vergessen werden. Für die Visualisierung muss auf dem PC ein Terminalprogramm installiert sein.

Als Beispiele für Terminalprogramm seinen hier angeführt:

- Putty066
- HTerm
- RealTerm
- uva.

Beim erstmaligen Betrieb des Bausteins FT232RL muss das Betriebssystem erst eine Geräteinstallation durchführen. Bei Windows10 erfolgt dies automatisch, bei Windows7 muss eventuell ein Treiber des Herstellers (FTDI) geladen werden. Für das Senden der Daten über UART wird in diesem Beispiel die Funktion:

printf(„Format String“, <wert1>.....) verwendet. Für den Format String gilt folgende Notation.

%d oder %i	“decimal” - Integerwert mit Vorzeichen
%u	“unsigned” - vorzeichenloser Integerwert
%x und %X	“HEX” - Integer in hexadezimaler Schreibweise
%p	“pointer” - Zeiger bzw. Speicheradresse
%f	“float” - Fließkommazahl
%e	“exponential” - Fließkommazahl in wissenschaftlicher Notation (m.nnExx)
%c	“character” - ein einzelnes Zeichen
%s	“string” - Zeichenkette

Die Umschaltung zwischen USB und UART erfolgt durch die Portleitung P0.2. Ein HI-Pegel verbindet die Funktionsgruppe UART mit der USB Buchse, Daten können übertragen werden.

2.6.2 Programm

```

/* ***** */
/* *****  BULME GRAZ, Serielle Schnittstelle, UART, Hello World  ***** */
/* *****  Abteilung Elektronik und Technische Informatik / Humer  ***** */

#include "mbed.h"           // Einbindung der mbed Library
DigitalOut sconhi(P0_2);   // P0.2=Ausgang (Umschalter für USB-UART)
Serial pc(P0_19, P0_18);   // TxD, RxD Definition der Portleitung
/* ***** Variablendefinition ***** */

/* ***** Hauptprogramm ***** */
int main()                 // Hauptprogramm
{
    pc.baud(9600);         // Datenübertragungsgeschw. 9600 Bit/sec
    sconhi=1;             // UART an USB verbinden
    wait(5);              // Warte 5 Sekunden
    while(1)              // Endlosschleife Beginn
    {
        pc.printf("Hello World\n\r"); // Ausgabe der Zeichenkette
    }                      // Endlosschleife Ende
}                          // end main

```

2.6.3 Ausgabe

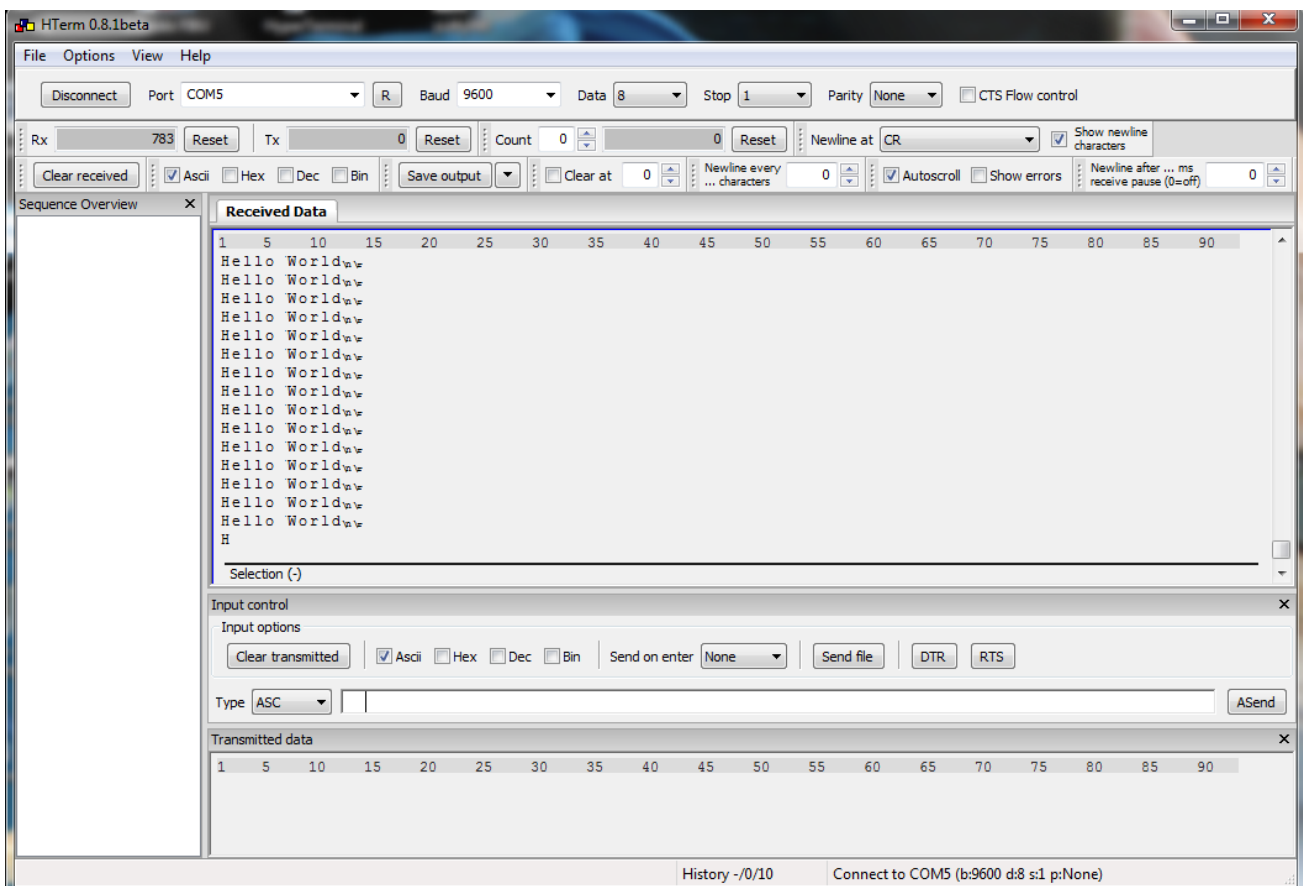


Abbildung 24: Visualisierung am PC mit Hilfe der Software HTerm

2.7 Einfacher Taschenrechner

2.7.1 Programm

Beispiel für einen sehr einfachen Taschenrechner. Die Grundfunktionen (+, -, /, *) sind integriert. Eine Minimalbeschreibung wird vor der Eingabe der Zahlen ausgegeben. Die Zahlenwerte haben den Datentyp float (Gleitkomma).

```

/* ***** */
/* ***** BULME GRAZ, einfacher Taschenrechner über UART ***** */
/* ***** Abteilung Elektronik und Technische Informatik / Humer ***** */
/* ***** */

#include "mbed.h"
DigitalOut sconhi(P0_2);          // P0.2=Ausgang (Umschalter für USB-UART)
Serial pc(P0_19, P0_18);        // TxD, RxD Definition der Portleitung
/* ***** Variablen definitionen ***** */
float zahl1;
float zahl2;
float erg;
char op;
/* ***** Hauptprogramm ***** */
main()
{
    pc.baud(9600);                // Datenübertragungsgeschw. 9600 Bit/sec
    sconhi=1;                    // UART an USB verbinden
    wait(5);                     // Warte 5 Sekunden, Verbindung am PC herstellen

while(1)
{
    pc.printf("Geben Sie zwei Zahlen ein: \n"); // Ausgabe
    pc.printf("Zahl1 Operator Zahl2 <return>, Komma mit Punkt\n");
    pc scanf("%f %c %f",&zahl1,&op,&zahl2); // Einlesen der Daten
    pc.printf("Zahl 1 = %7.2f \n",zahl1); // Ausgabe der Zahl1
    pc.printf("Operator = %c\n",op); // Ausgabe des Operators
    pc.printf("Zahl 2 = %7.2f \n",zahl2); // Ausgabe der Zahl2
    pc.printf("-----\n");
    switch(op)
    {
    case '+':
    pc.printf("Ergebnis = %7.2f\n",(float)zahl1+zahl2); // Ergebnis für +
    break;
    case '-':
    pc.printf("Ergebnis = %7.2f\n",(float)zahl1-zahl2); // Ergebnis für -
    break;
    case '/':
    pc.printf("Ergebnis = %7.2f\n",(float)zahl1/zahl2); // Ergebnis für /
    break;
    case '*':
    pc.printf("Ergebnis = %7.2f\n",(float)zahl1*zahl2); // Ergebnis für *
    break;
    } /* end switch */
    pc.printf("\n\r"); // Leerzeile
} /* end while */
} /* end main */

```

2.7.2 Visualisierung

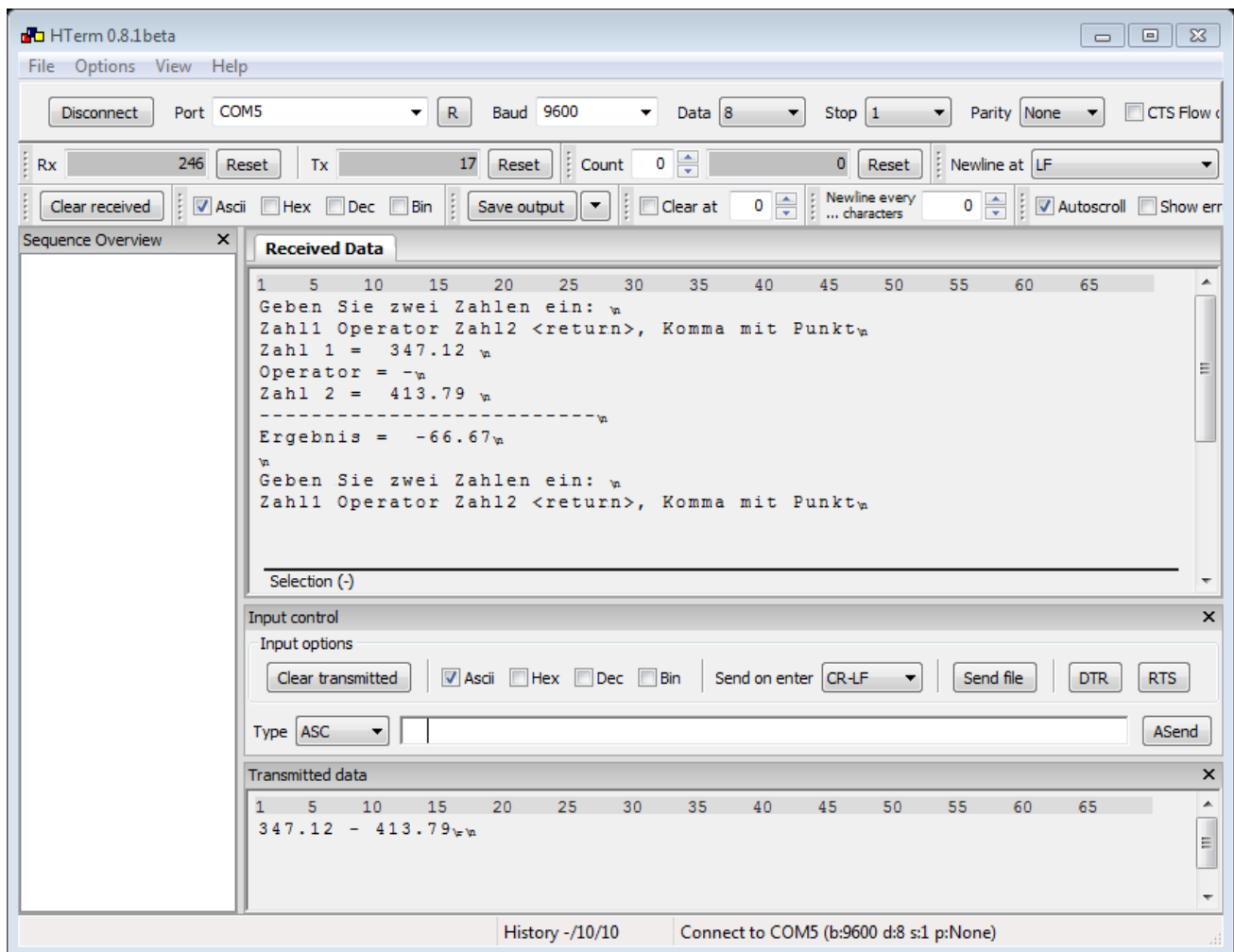


Abbildung 25: Visualisierung am PC mit HTerm

Die Verbindung wird sehr einfach hergestellt:

Das Programm HTerm wird gestartet, alle Einstellungen wie Port, Baudrate, Newline und Sendeabschluss werden eingestellt. Als nächstes wird am Microboard die RESET-Taste gedrückt. Das Board meldet sich am PC an und bekommt die installierte Portadresse zugewiesen. Durch den Button „Connect“ in der HTerm-App wird die Verbindung hergestellt.

2.8 Zeichenumkehr

2.8.1 Programm

```

/* ***** */
/* ***** BULME GRAZ, eingegebene Zeichenfolge umdrehen ***** */
/* ***** Abteilung Elektronik und Technische Informatik / Humer ***** */
/* ***** */

#include "mbed.h"
DigitalOut sconhi(P0_2);          // P0.2=Ausgang (Umschalter für USB-UART)
Serial pc(P0_19, P0_18);        // TxD, RxD Definition der Portleitung

/* ***** Variablendefinitionen ***** */
int i=0;                          // Zählvariable
char eingabe[80];                 // Speicher für max. 80 Zeichen (Array)
char *pointer;                    // Definition Pointer

/* ***** Hauptprogramm ***** */
main()
{
    pc.baud(9600);                // Datenübertragungsgeschw. 9600 Bit/sec
    sconhi=1;                     // UART an USB verbinden
    wait(5);                      // Warte 5 Sekunden
    while(1)                      // Endlosschleife
    {
        pc.printf("\nGeben Sie einen Text ein:\n"); // Ausgabe
        pointer=eingabe;          // Pointer auf die Adresse von Eingabe setzen
        i=0;                      // Zählvariable
        while ((*pointer=pc.getc())!=13) // Zeicheneingabe speichern
        {                          // bis die Taste Enter gedrückt
            pc.putc(*pointer); // lesen und auf die Adresse von Pointer schreiben
            i++;                // increment i
            pointer++;          // Zeiger, Zeigeradresse um 1 erhöhen
        }

        pc.printf("\nDer eingegeben Text lautet:\n %s \n",eingabe);
        pc.printf("Die Anzahl der eingegebenen Buchstaben sind: %d\n",i);

/* ***** Zeichenfolge umdrehen ***** */
        i=strlen(eingabe)-1;      // i=Anzahl der Zeichen - 1
        pointer=eingabe;          // Setze Pointer auf Array eingabe
        pointer+=i;               // Setze Zeiger auf das letzte Zeichen
        while(i>=0)
        {
            pc.putc(*pointer); // Ausgabe der Zeichen aus dem Array
            pointer--;          // Zeiger um eine Stelle tiefer
            i--;                // decrement i
        } // end while
    } // end while
} // end main

```

2.8.2 Anzeige

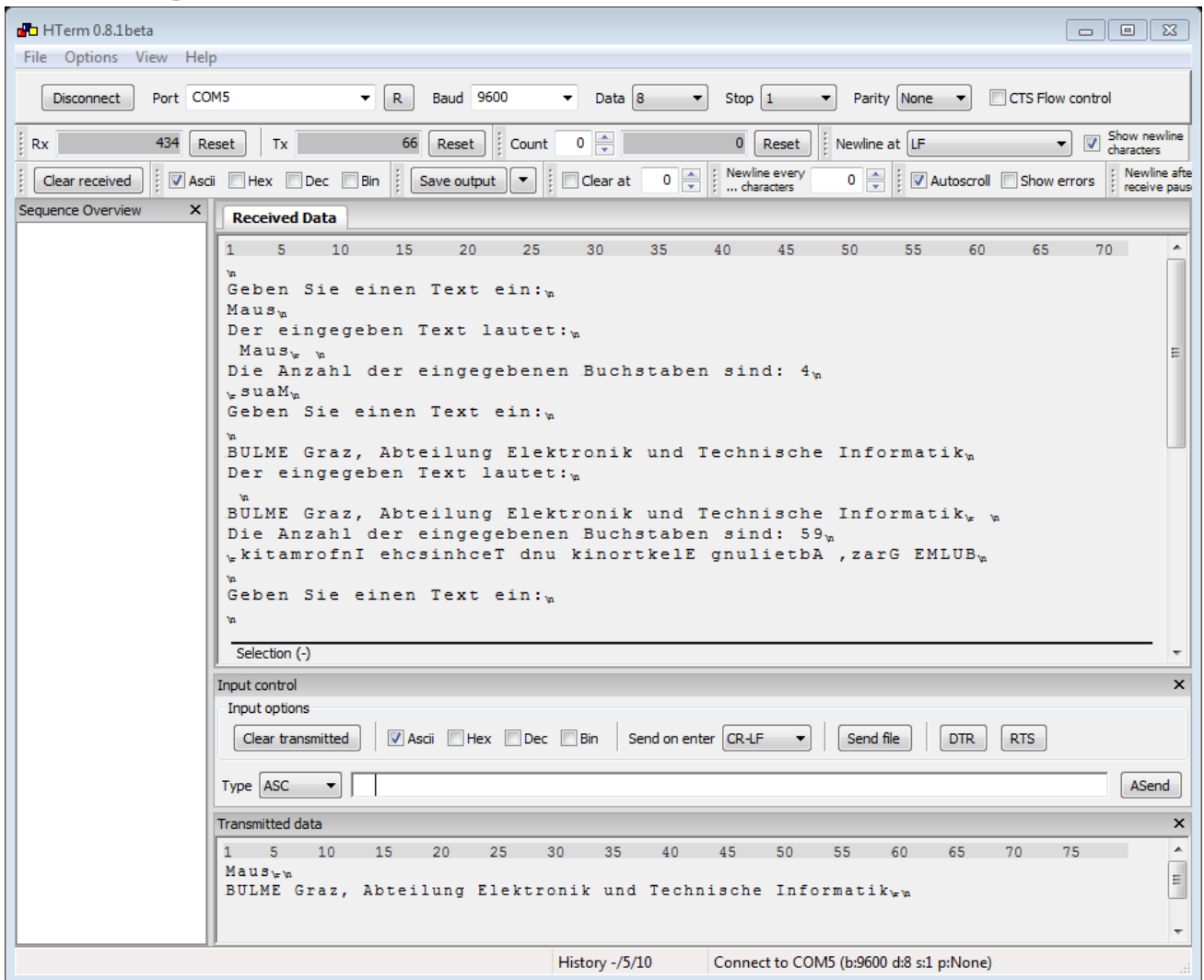


Abbildung 26: Visualisierung Ein/Ausgabe am PC

In diesem Beispiel werden 2 Wörter eingegeben:

- a) Maus
- b) BULME Graz, Abteilung Elektronik und Technische Informatik

2.8.3 Spezielle Zeichendarstellung

```

/* ***** */
/* ***** BULME GRAZ, eingegebene Zeichenfolge umdrehen, Ausg. in Dec,Hex ** */
/* ***** Abteilung Elektronik und Technische Informatik / Humer ***** */
/* ***** */
#include "mbed.h"
DigitalOut sconhi(P0_2);          // P0.2=Ausgang (Umschalter für USB-UART)
Serial pc(P0_19, P0_18);        // TxD, RxD Definition der Portleitung
/* ***** Variablendefinition ***** */
int laenge;
int i;
char eingabe[40],invers[40];

/* ***** Hauptprogramm ***** */
main()
{
    pc.baud(9600);                // Datenübertragungsgeschw. 9600 Bit/sec
    sconhi=1;                    // UART an USB verbinden
    wait(5);                     // Warte 5 Sekunden
    while(1)
    {
        pc.printf("\nGeben Sie einen Text ein:\n");
        pc.scanf("%s",&eingabe); // Wort einlesen und in Array speichern
        pc.printf("\nDer eingegeben Text lautet:\n%s \n",eingabe); // Ausgabe
        laenge=strlen(eingabe); // Ermittlung der Wortlaenge
        pc.printf("Die Anzahl der eingegebenen Buchstaben sind: %d\n",(int)laenge);

        /* ***** Ausgabe des Wortes mit einer Buchstabenweite von 5 Zeichen **** */
        for(i=0;i<laenge;i++)
            pc.printf(" %c ",eingabe[i]);
        pc.printf(" Buchstaben\n");

        /* ***** Ausgabe des ASCII-Codes in Hexadezimaler Darstellung ***** */
        for(i=0;i<laenge;i++)
            pc.printf("%3X ",(int)eingabe[i]);
        pc.printf(" HEX-Code\n");

        /* ***** Ausgabe des ASCII-Codes in Dezimaler Darstellung ***** */
        for(i=0;i<laenge;i++)
            pc.printf("%3d ",(int)eingabe[i]);
        pc.printf(" Dezimal-Code\n");

        /* ***** Zeichenfolge umdrehen und in ein ARRAY invers schreiben ***** */
        for(i=laenge-1;i>=0;i--)
            invers[(laenge-1)-i]=eingabe[i];

        /* ***** Ausgabe des Wortes mit einer Buchstabenweite von 5 Zeichen *** */
        pc.printf("Eingabe invers= %s\n",invers);
        for(i=0;i<laenge;i++)
            pc.printf(" %c ",invers[i]);
        pc.printf(" Buchstaben\n");

        /* ***** Ausgabe des ASCII-Codes in Hexadezimaler Darstellung ***** */
        for(i=0;i<laenge;i++)
            pc.printf("%3X ",(int)invers[i]);
        pc.printf(" HEX-Code\n");

        /* ***** Ausgabe des ASCII-Codes in Dezimaler Darstellung ***** */
        for(i=0;i<laenge;i++)
            pc.printf("%3d ",(int)invers[i]);
        pc.printf(" Dezimal-Code\n");
    } //end main while
}

```

2.8.4 Anzeige

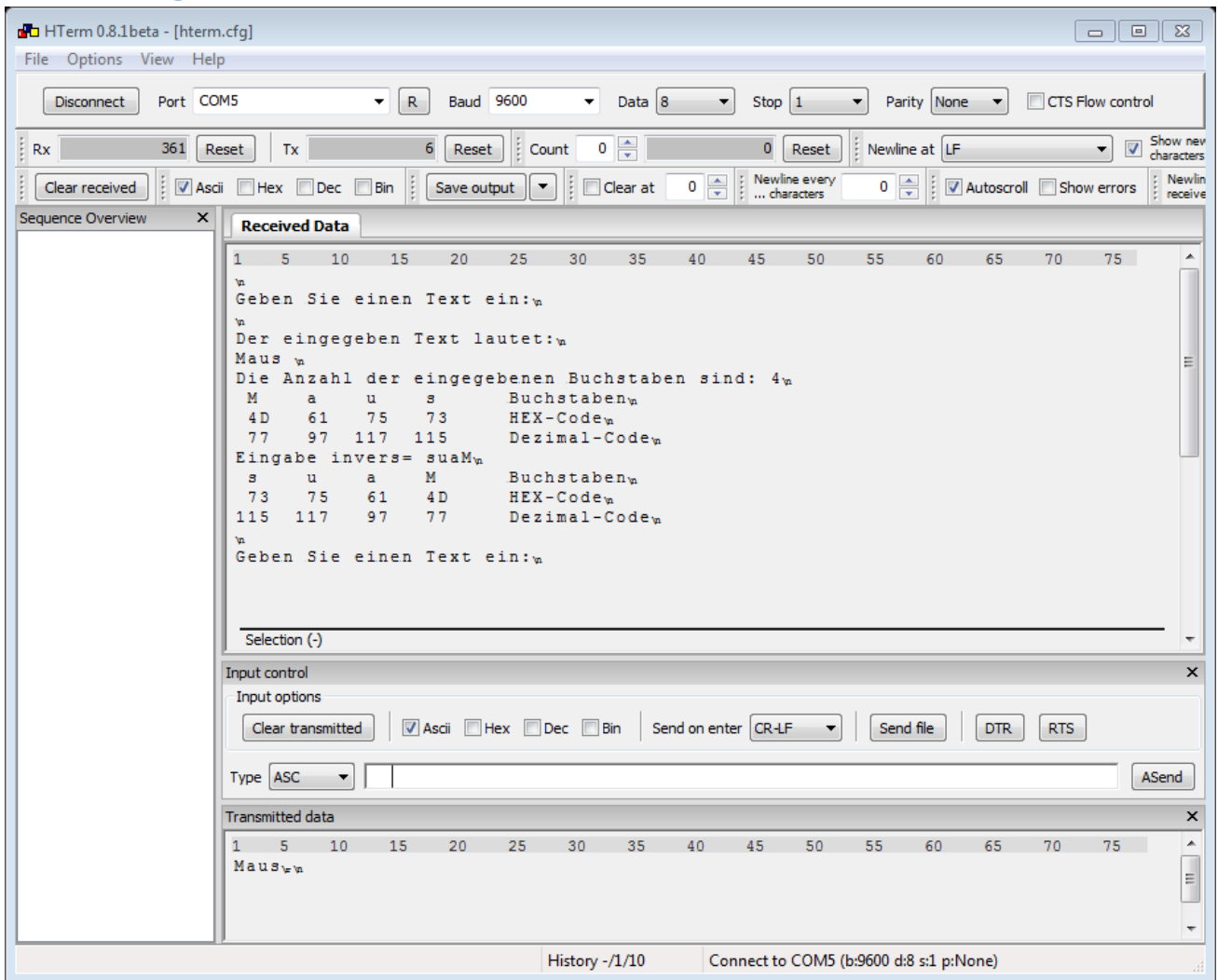


Abbildung 27: Visualisierung der Eingabe „Maus“ und deren Codes am PC

2.9 Elektronischer Würfel

2.9.1 Allgemeines

Für einen elektronischen Würfel wird ein Pseudozufallszahlengenerator verwendet. Für die Zufallszahl benötigt der Microcontroller Befehle für die Generierung einer Zufallszahl. Durch die Befehle ist die ermittelte Zahl per Definition nicht zufällig. In den meisten Anwendungen ist die Pseudozufallszahl jedoch ausreichend.

2.9.2 Schaltung

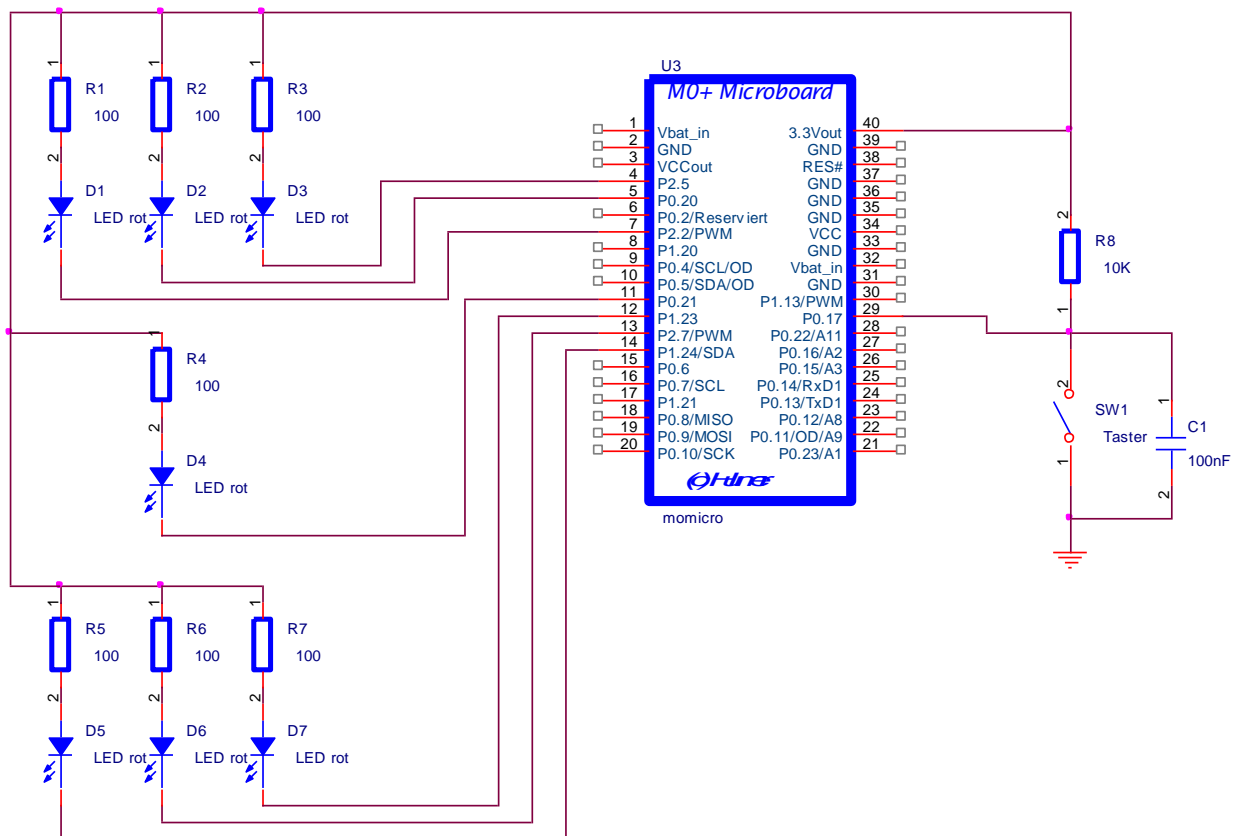
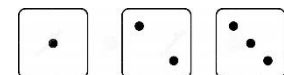


Abbildung 28: Schaltung für den elektronischen Würfel

Es werden Zahlen von 1 ... 6 werden, dabei wird folgender Zusammenhang festgelegt:

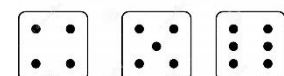
Zahl 1: Leuchtdiode D4 aktiv

Zahl 2: Leuchtdioden D1 und D7 aktiv



Zahl 3: Leuchtdioden D1, D4 und D7 aktiv

Zahl 4: Leuchtdioden D1, D3, D5 und D7 aktiv



Zahl 5: Leuchtdioden D1, D3, D5, D7 und D4 aktiv

Zahl 6: Leuchtdioden D1, D2, D3 und D5, D6, D7 aktiv

Ein weiterer wichtiger Punkt ist die negative Logik in der angeführten Schaltung. Ein LO-Signal am Ausgang bringt die LED zum Leuchten. Im Programm sind die einzelnen Ausgänge in einem Bus zusammengefasst, wobei die Portleitung P1.24 als LSB⁹ definiert ist.

```
BusOut wuerfel(P2_5, P0_20, P2_2, P0_21, P1_23, P2_7, P1_24);
```

Zahl	P2.5	P0.20	P2.2	P0.21	P1.23	P2.7	P1.24	Wert
LED	D1	D2	D3	D4	D5	D6	D7	
1	1	1	1	0	1	1	1	0x77
2	0	1	1	1	1	1	0	0x3E
3	0	1	1	0	1	1	0	0x36
4	0	1	0	1	0	1	0	0x2A
5	0	1	0	0	1	1	0	0x26
6	0	0	0	1	0	0	0	0x08

Tabelle 3: Ausgabe für das entsprechende Würfelergbnis

Entsprechend der Tabelle wird ein Array definiert:

```
int anzeige[7]={0xFF,0x77,0x3E,0x36,0x2A,0x26,0x08};
```

Das Würfelergbnis „0“ gibt es nicht, in der Tabelle wurde der Wert 0xFF eingetragen.

Das Würfelergbnis wird in der Interrupt-Funktion mit der Funktion rand() ermittelt. Modulo 6 bedeutet, dass Ergebnisse von 0 ... 5 geliefert werden.

```
void flip() // Interruptfunktion flip()
{
wuerfelerg = rand()%6 + 1;
newerg=1;
} // end void
```

⁹ LSB=Least Signifikant Bit

2.9.3 Programm

In diesem Beispiel wird auf die Funktion `rand()` zurückgegriffen. Das „gewürfelte“ Ergebnis wird über 7 LEDs angezeigt, aber auch der Zahlenwert über die UART Schnittstelle an den PC geschickt. Der Port P0.17 ist als Interrupt definiert und durch die Beschaltung auf fallende Flanke getriggert.

```

/* ***** */
/* *****      BULME GRAZ, Temperatursensor LM235      ***** */
/* *****  Abteilung Elektronik und Technische Informatik / Humer  ***** */
#include "mbed.h"

/* *****      Definitionen      ***** */
BusOut wuerfel(P2_5, P0_20, P2_2, P0_21, P1_23, P2_7, P1_24);
DigitalOut sconhi(P0_2);          // P0.2=Ausgang (Umschalter für USB-UART)
Serial pc(P0_19, P0_18);         // TxD, RxD Definition der Portleitung
InterruptIn button(P0_17);       // Interrupteingang Taste

/* *****      Funktionen      ***** */
void init_mcboard();
char newerg=0;
int anzeige[7]={0xFF,0x77,0x3E,0x36,0x2A,0x26,0x08};
/* *****      Variablendeklarationen      ***** */
int wuerfelerg;

/* *****      Interruptfunktion      ***** */
void flip()                      // Interruptfunktion flip()
{
wuerfelerg = rand()%6 + 1;
newerg=1;
}                                // end void

/* *****      Hauptprogramm      ***** */
int main()
{
  init_mcboard();                // Initialisierung Microboard
  pc.printf("Elektronischer Wuerfel\n");
  while(1)
  {
    if(newerg==1)
    {
      pc.printf("Ergebnis=%d\n",wuerfelerg);
      wuerfel=anzeige[wuerfelerg];
      wait(0.2);
      newerg=0;
    }
  } // end while
} // end main

/* *****      Funktionen      ***** */
void init_mcboard()
{
  pc.baud(9600);                 // Datenübertragungsgeschw. 9600 Bit/sec
  sconhi=1;                      // UART an USB verbinden
  wait(5);                       // Warte 5 Sekunden
  button.fall(&flip);             // Interrupt bei fallender Flanke
}

```

2.9.4 Ausgabe

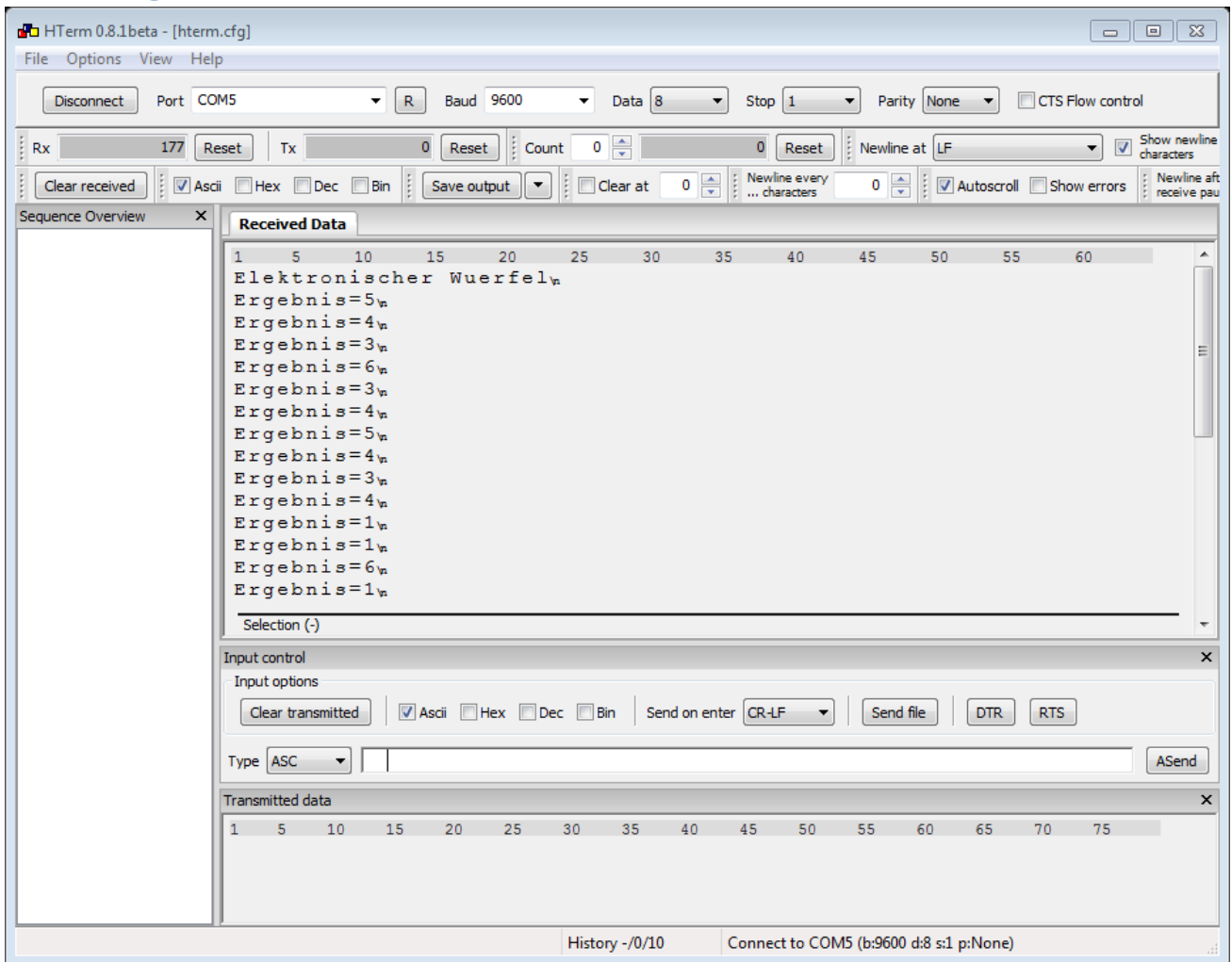


Abbildung 29: Anzeige der Würfelergebnisse am Bildschirm mit einem Terminalprogramm

2.10 PWM

2.10.1 Allgemeines

PWM (Pulse Weitenmodulation) ist für viele Anwendungen ein wichtiges Werkzeug in der Microcontrollertechnik. Es stehen 3 Portleitungen zur Verfügung.

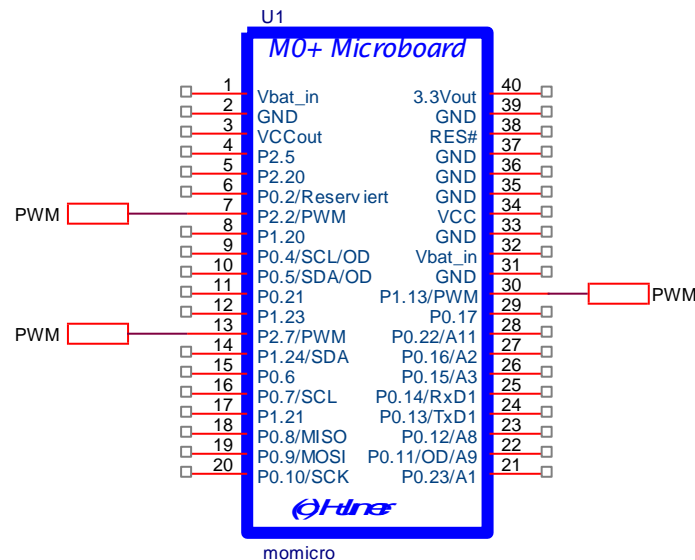


Abbildung 30: 3 Portleitungen für die interne PWM Einheiten

Für die Gehäuseform 48 Pin sind in der mbed-Bibliothek die Portleitung P2.2 oder P2.7 oder P1.13 einsetzbar. Für eine diskrete Programmierung ist nahezu jede Portleitung verwendbar.

2.10.2 Beispiel

```

/* ***** */
/* *** Programm: PWM - Unit, BULME Elektronik, Humer ***** */
/* ***** */

#include "mbed.h"

PwmOut mitzi(P1_13); // Definition Portleitung P1.13 als PWM Ausgang

int main() // Hauptprogramm
{
    mitzi.period(0.02); // 20msec period
    mitzi = 0.25; // 25% = 5msec Pulse (on)

    while(1); // Endlosschleife
} //end main

```

2.10.3 Anzeige

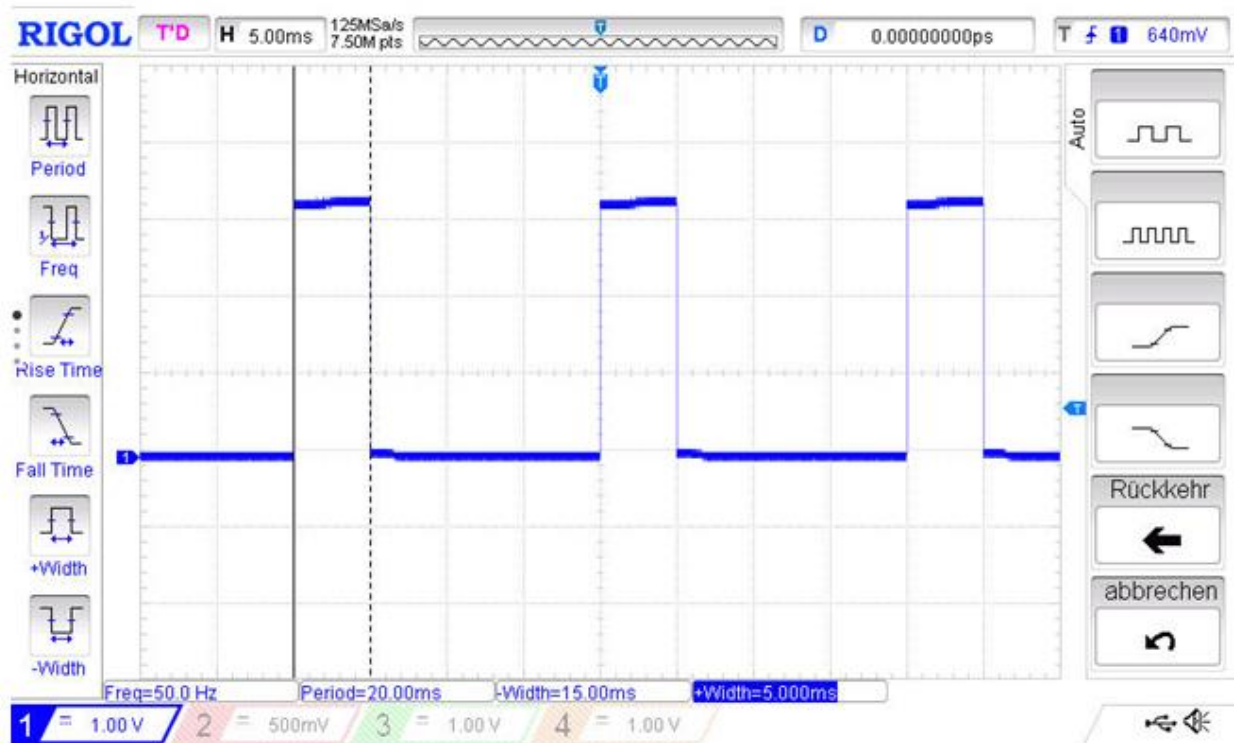


Abbildung 31: Visualisierung der Signalform am Ausgang eines Oszilloskops

2.11 2 Kanal-PWM

In diesem Beispiel werden 2 PWM Signale unterschiedlich erzeugt. Am Ausgang P1.13 wird die Library verwendet, am Ausgang P0.20 wird das PWM Signal mit duty-cycle 50% diskret erstellt.

```

/* ***** */
/* *****   BULME GRAZ, Pulsweitenmodulation 2 Varianten   ***** */
/* *****   Abteilung Elektronik und Technische Informatik / Humer   ***** */

#include "mbed.h"

PwmOut mitzi(P1_13);           // P1.13 - als PWM-Ausgang definieren
DigitalOut minki(P0_4);       // P0.4 - als Digital Out definieren

int main()
{
    mitzi.period(0.02);        // PWM-Signal, 20msec period
    mitzi = 0.25;              // 5msec Pulse (on), dc=25%, Wertebereich 0..1

    while(1)                   // Endlosschleife
    {
        minki=!minki;          // Portleitung invertieren
        wait(0.01);           // Warte 10msec
    }
}

```

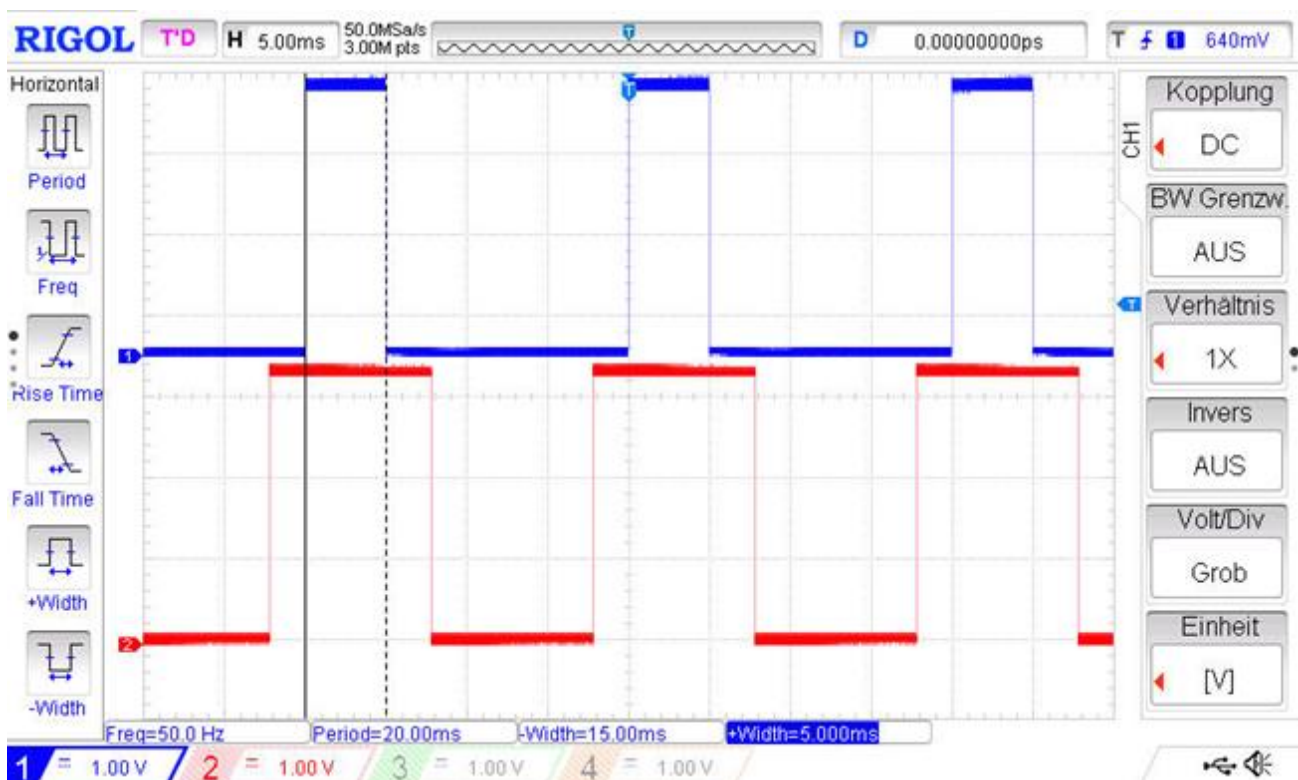


Abbildung 32: Die Darstellung der 2 PWM Signale am Oszilloskop

2.12 Analog-Digital-Umsetzer

2.12.1 Allgemeines

Der ADU¹⁰ ist ein sehr wichtiger Peripheriebaustein eines Microcontrollers, mit dessen Hilfe eine Spannung innerhalb des Versorgungsspannungsbereiches in eine Digitale Zahl umgesetzt wird. Dabei sind die wichtigsten Kenngrößen die Umsetzbreite in Bit, der Eingangsspannungsbereich und die Umsetzgeschwindigkeit. Der verwendete Microcontroller hat einen ADU-Kern mit einer Auflösung von 12 Bit, einen Eingangsspannungsbereich von 3,3 Volt und eine maximale Umsetzgeschwindigkeit von 2 Msps¹¹. Der Microcontroller LPC1114U68 hat einen einzigen Analog-Digital-Umsetzer, durch Anlogschalter können, je nach Gehäuseform, 12 Kanäle bedient werden. Die Gehäuseform 48 Pin hat 6 Kanäle zur Verfügung.

Eine Auflösung von 12 Bit bedeutet, dass der Eingangsspannungsbereich von 3,3 Volt auf 2^{12} ($2^{12} = 4096$) Schritte aufgeteilt wird, dabei ergibt sich eine Auflösung der Eingangsspannung von:

$$\text{Auflösung} = \frac{3,3[\text{V}]}{4096} = 0,806[\text{mV}].$$

Am Microboard sind folgende PINs für die Messung einer Analogspannung gekennzeichnet.

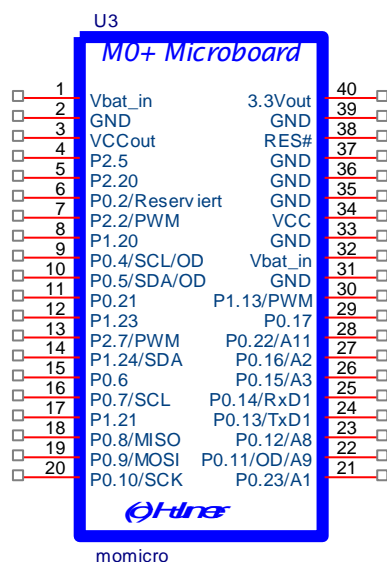


Abbildung 33: Pin-Belegung Microboard

Im obigen Bild ist zu erkennen, dass die Pin 21,22,23,26,27 und 28 für den Analog-Digital-Umsetzer geeignet sind.

¹⁰ Analog-Digital-Umsetzer, englisch: Analog Digital Converter

¹¹ Mega samples per second

2.12.2 Portdefinition

In der Entwicklungsumgebung mbed werden Analogeingänge wie folgt definiert:

```
AnalogIn <name> (PX_X);          z.B: AnalogIn minki (P0_23);
```

2.12.3 Einlesen einer Analogspannung

Die Analogspannung kann auf 2 verschiedene Arten eingelesen werden:

```
a) <variable>(float) = minki.read();
```

In diesem Fall wird ein Wert (float) mit einem Wertebereich von 0 bis 1 übergeben. Um die Spannung am PIN zu erhalten, muss noch mit der Referenzspannung (hier 3,3 V) multipliziert werden.

```
b) <variable>(int) = minki.read_u16();
```

Im Fall b) ist die Rückgabe der aufgerufenen Funktion ein normierter 16-stelliger Ganzzahlwert. Der Wertebereich ist dabei 0 ... 65535 oder Hexadezimal 0 ... 0xFFFF.

In beiden Fällen muss beachtet werden, dass der Analog-Digital-Umsetzer am Eingang eine Sample and Hold (S&H) Stufe hat und die Analogspannung nur in einem sehr kurzen Zeitfenster gelesen wird. Hier kann es durch die Leitungslängen zu Spannungsabfällen und somit zu Messungenauigkeiten kommen. Ein Kondensator in der Nähe des Microcontrollers stützt die Eingangsspannung und liefert die Energie, um einen Spannungseinbruch zu vermeiden.

2.13 Spannungsmessung an einem Potentiometer

2.13.1 Allgemeines

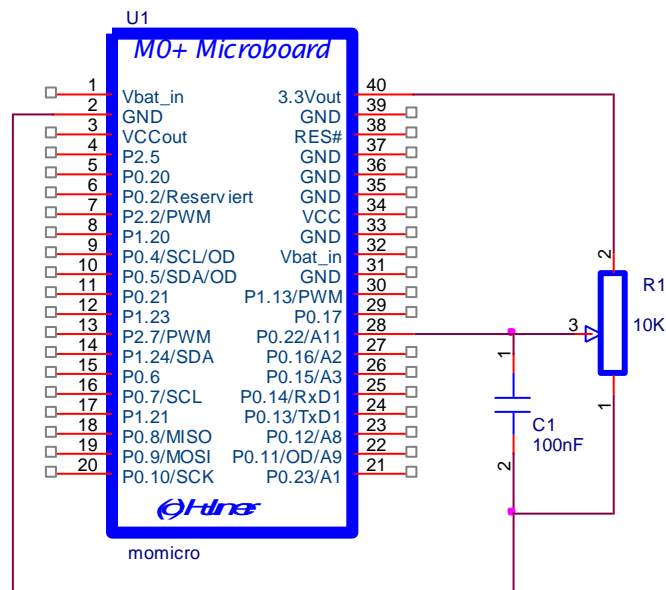


Abbildung 34: Versuchsaufbau für eine Spannungsmessung

Im obigen Bild ist ein Versuchsaufbau für eine Spannungsmessung an einem Potentiometer dargestellt. Die abgegriffene Spannung am Punkt 3 ist mit dem Analogkanal 11 (PIN 28) verbunden. Der Kondensator C1 dient als Stützkondensator für das S&H.

2.13.2 Programm

```

/* ***** */
/* ***** BULME GRAZ, Spannungsmessung an einem Potentiometer ***** */
/* ***** Abteilung Elektronik und Technische Informatik / Humer ***** */
#include "mbed.h" // Einbindung der mbed Library

DigitalOut sconhi(P0_2); // P0.2=Ausgang (Umschalter für USB-UART)
Serial pc(P0_19, P0_18); // Tx, Rx Definition der Portleitung
AnalogIn Poti(P0_22); // Analogeingang Kanal 11 - Potentiometer

/* ***** Variablendefinition ***** */
int Poti_i; // normierter ADC-Wert als 16 Bit Wert 0 ... 0xFFFF
float Poti_f; // normierter ADC-Wert als float Wert 0 ... 1

/* ***** Hauptprogramm ***** */
int main() // Hauptprogramm
{
    pc.baud(9600); // Datenübertragungsgeschw. 9600 Bit/sec
    sconhi=1; // UART an USB verbinden
    wait(5); // Warte 5 Sekunden
    while(1) // Endlosschleife Beginn
    {
        Poti_f = Poti.read(); // Wert aus dem ADC 0<=adcntc<=1, float
        Poti_i = Poti.read_ul16(); // Wert einlesen - Ganzzahl 0 ... 65535 (0xFFFF)
        wait(0.5); // Warte 0,5 Sekunden
        pc.printf("Poti: %1.3f[V] %d, 0x%04X\n\r", 3.3*Poti_f, Poti_i, Poti_i);
    } // end while
} // end main

```

2.13.3 Ausgabe

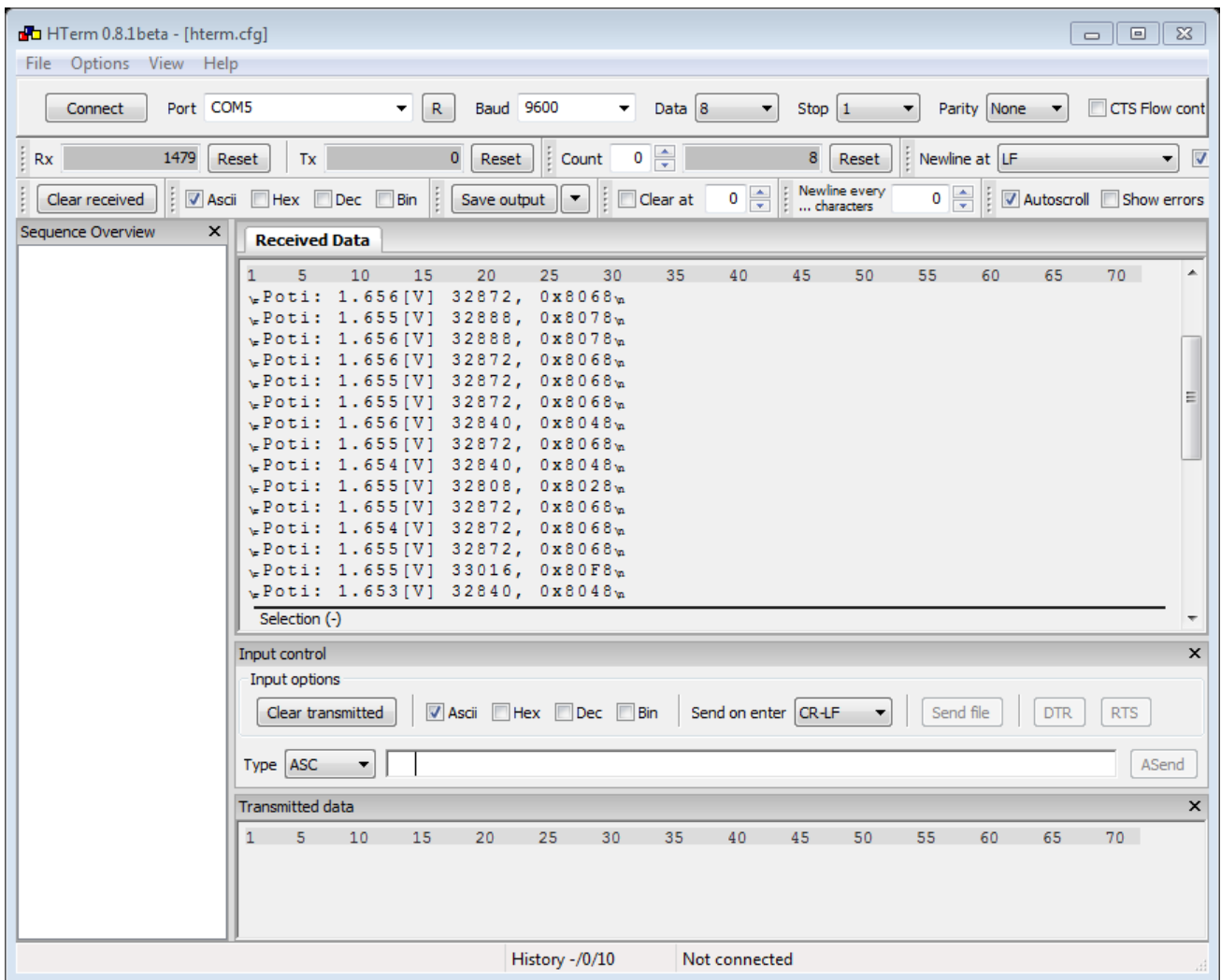


Abbildung 35: Ergebnis der Spannungsmessung am Potentiometer

Die Ausgabe erfolgt in diesem Beispiel mit der Spannung am Potentiometer, dem Dezimalwert und Hexadezimalwert des ADCs.

2.14 Temperaturmessung

2.14.1 Messung mit Analogsensor LM235

2.14.1.1 Allgemeines

Der Temperatursensor LM235 ist ein Präzisionsensor mit einer absoluten Genauigkeit von ± 1 K. Die Ausgangsspannung ist mit 10 mV/K angegeben.

Der Auszug aus dem Datenblatt lautet:

- Directly calibrated in K ■
- 1 °C initial accuracy ■
- Operates from 450 μ A to 5 mA ■
- Less than 1 Ω dynamic impedance



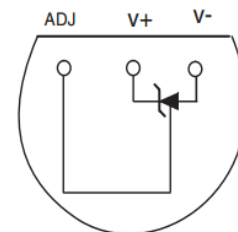
TO-92
(Plastic package)



SO-8
(Plastic micropackage)

Pin connections

TO-92
(Bottom view)



2.14.1.2 Protoboard Aufbau

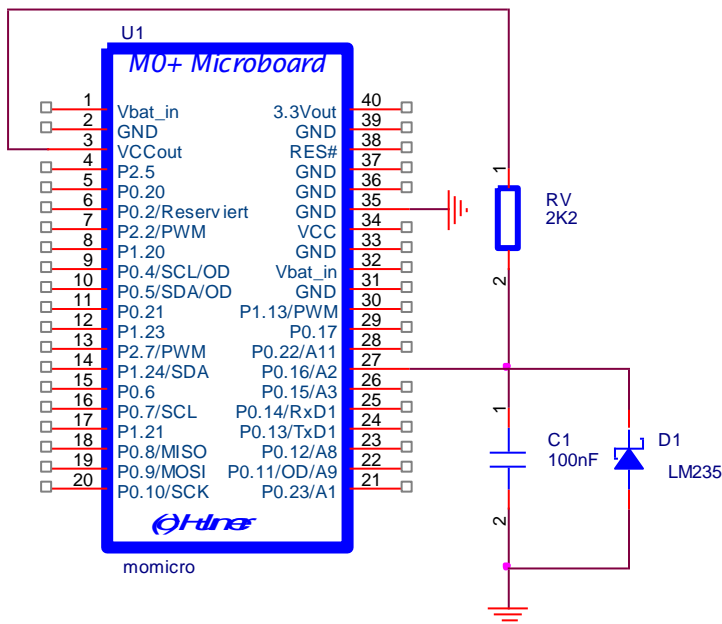


Abbildung 36: Aufbau der Schaltung mit dem IC LM235 auf dem Protoboard

Der Sensor wird über den Widerstand R1 (2200 Ω) mit einer Spannung von 5 V versorgt. Der Strom ist dabei etwa bei einer Raumtemperatur von 20 °C:

$$Strom = \frac{5 - (2,7315 + 0,20)}{2200} = 0,94 \text{ [mA]}$$

2.14.1.3 Programm

```

/* ***** */
/* *****      BULME GRAZ, Temperatursensor LM235      ***** */
/* *****  Abteilung Elektronik und Technische Informatik / Humer  ***** */
#include "mbed.h"

/* *****      Definitionen      ***** */
DigitalOut sconhi(P0_2);          // P0.2=Ausgang (Umschalter für USB-UART)
Serial pc(P0_19, P0_18);         // TxD, RxD Definition der Portleitung
AnalogIn templm235(P0_16);       // Temperatursensor LM235
/* *****      Funktionen      ***** */
void init_mcboard();

/* *****      Variablendeklarationen      ***** */

/* *****      Hauptprogramm      ***** */
int main()
{
    init_mcboard();              // Initialisierung Microboard

    while(1)
    {
        pc.printf("***** Temperaturmessung *****\n"); //Ausgabe
        pc.printf("Temp:= %3.1f Grad Celsius\n\r",330*templm235.read()-273.15);
        wait(0.5);                // Warte 0.5 Sekunden
    } // end while
} // end main

/* *****      Funktionen      ***** */
void init_mcboard()
{
    pc.baud(9600);                // Datenübertragungsgeschw. 9600 Bit/sec
    sconhi=1;                    // UART an USB verbinden
    wait(5);                      // Warte 5 Sekunden
}

```


2.14.2 Messung mit NTC-Sensor

2.14.2.1 Allgemeines

Ein NTC¹² Widerstand ist ein Halbleiter und wird in unterschiedlichen Bauformen eingesetzt. Die Widerstands- Temperaturkennlinie ist stark nichtlinear und kann durch folgende Formel beschrieben werden:

$$R_T = R_{T_x} \cdot \exp \left[\frac{\alpha_x}{100} \cdot (T_x + 273,15)^2 \cdot \left(\frac{1}{T + 273,15} - \frac{1}{T_x + 273,15} \right) \right]$$

R_T	Widerstandswert bei der Temperatur T
R_{T_x}	Widerstandswert am Beginn des betreffenden Temperaturintervalls
T_x	Temperatur in °C am Beginn des betreffenden Temperaturintervalls
T	Interessierende Temperatur in °C ($T_x < T < T_{x+1}$)
α_x	Temperaturkoeffizient bei der Temperatur T_x

Abbildung 38: Zusammenhang zwischen Temperatur und Widerstand (EPCOS)

Der Zusammenhang zwischen Temperatur und Widerstandswert des NTC kann auch näherungsweise durch folgende Formel¹³ berechnet werden:

$$R_T = R_N \cdot e^{B \left(\frac{1}{T} - \frac{1}{T_N} \right)}$$

R_N ... Widerstand bei der Bezugstemperatur (hier 10 K Ω),

T_N Bezugstemperatur (hier 25 °C)

R_T Widerstandswert der aktuellen Temperatur, T aktuelle Temperatur, B Materialkonstante (Datenblatt)

2.14.2.2 Aufbau auf dem Protoboard

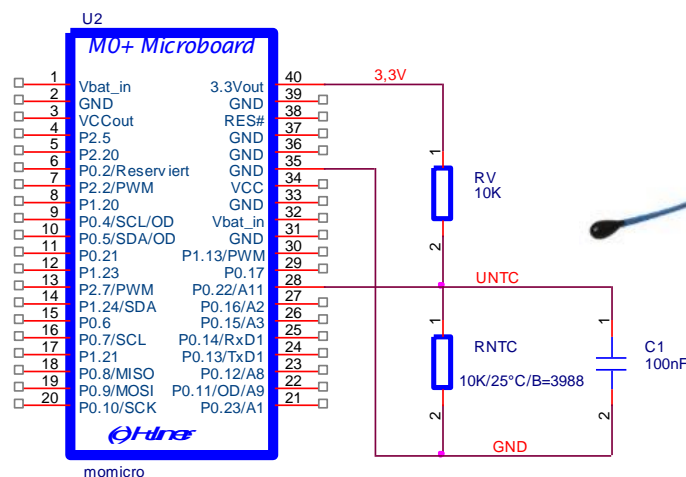


Abbildung 39: Schaltung für die Temperaturmessung mit einem NTC-Widerstand (Photo)

¹² NTC (Negative Temperaturkoeffizient)

¹³ Siemens Matsushita Components, Heißleiter - Datenbuch 1996

Temperatur	$R_{25}=3k\Omega$	$R_{25}=5k$	$R_{25}=10k\Omega$	$R_{25}=30k\Omega$		Temperatur	$R_{25}=3k\Omega$	$R_{25}=5k$	$R_{25}=10k\Omega$	$R_{25}=30k\Omega$
- 40,0	100950	168250	336500	907200		41	1535	2559	5117	15495
- 35,0	72777	121290	242600	663000		42	1475	2459	4917	14895
- 30,0	53100	88500	177000	489600		43	1418	2363	4726	14325
- 25,0	39111	65185	130400	365100		44	1363	2272	4543	13776
- 20,0	29121	48535	97070	274590		45	1311	2184	4369	13254
- 15,0	21879	36465	72930	208350		46	1261	2101	4202	12750
- 10,0	16599	27665	55330	159390		47	1213	2021	4042	12273
- 5,0	12695	21158	42320	122910		48	1167	1945	3889	11811
0	9795	16325	32650	954490		49	1123	1872	3743	11373
1	9308	15514	31030	90870		50	1081	1802	3603	10950
2	8849	14747	29490	86490		51	1041	1734	3469	10545
3	8414	14024	28050	82350		52	1002	1670	3340	10158
4	8004	13340	26680	78420		53	965	1609	3217	9786
5	7616	12694	25390	74730		54	930	1550	3099	9429
6	7250	12083	24170	71220		55	896	1493	2986	9090
7	6903	11505	23010	67890		56	863	1439	2878	8760
8	6575	10958	21920	64710		57	832	1387	2774	8448
9	6264	10440	20880	61740		58	802	1337	2675	8148
10	5970	9950	19900	58890		59	774	1290	2579	7857
11	5691	9485	18970	56190		60	746	1244	2488	7581
12	5426	9044	18090	53640		61	720	1200	2400	7314
13	5175	8626	17250	51210		62	695	1158	2316	7059
14	4938	8230	16460	48900		63	671	1118	2235	6818
15	4712	7854	15710	46710		64	647	1079	2158	6579
16	4499	7498	15000	44640		65	625	1042	2083	6354
17	4296	7160	14320	42660		66	603	1006	2011	6135
18	4103	6884	13680	40800		67	583	971	1943	5928
19	3921	6534	13070	39030		68	563	938	1877	5727
20	3747	6245	12490	37320		69	544	907	1813	5535
21	3582	5970	11940	35700		70	526	876	1752	5349
22	3425	5709	11420	34170		75	444	741	1481	4524
23	3276	5461	10920	32730		80	377	629	1258	3840
24	3135	5225	10450	31320		85	322	536	1072	3273
25	3000	5000	10000	30000		90	275	459	918	2799
26	2872	4786	9572	28737		95	237	394	789	2405
27	2750	4583	9165	27531		100	204	340	680	2073
28	2633	4389	8777	26385		105	177	294	589	1792
29	2523	4204	8408	25290		110	153	256	511	1555
30	2417	4029	8057	24249		115	134	223	445	1354
31	2317	3861	7722	23256		120	117	195	389	1182
32	2221	3701	7402	22305		125	103	171	342	1035
33	2129	3549	7098	21402		130	90	150	301	910
34	2042	3404	6808	20538		135	80	133	265	802
35	1959	3266	6531	19716		140	70	117	235	708
36	1880	3134	6267	18927		145	62	104	208	627
37	1805	3008	6016	18177		150	56	93	185	557
38	1733	2888	5775	17460		155	50	83	165	-
39	1664	2773	5546	16773						
40	1598	2664	5327	16119						

Tabelle 4: Widerstandswerte in Abhängigkeit von der Temperatur und NTC-Typ.

In diesem Beispiel wird ein NTC-Widerstand mit 10 K Ω bei 25 °C verwendet. Die Temperatur ist in °C angegeben.

2.14.2.3 Kennlinie

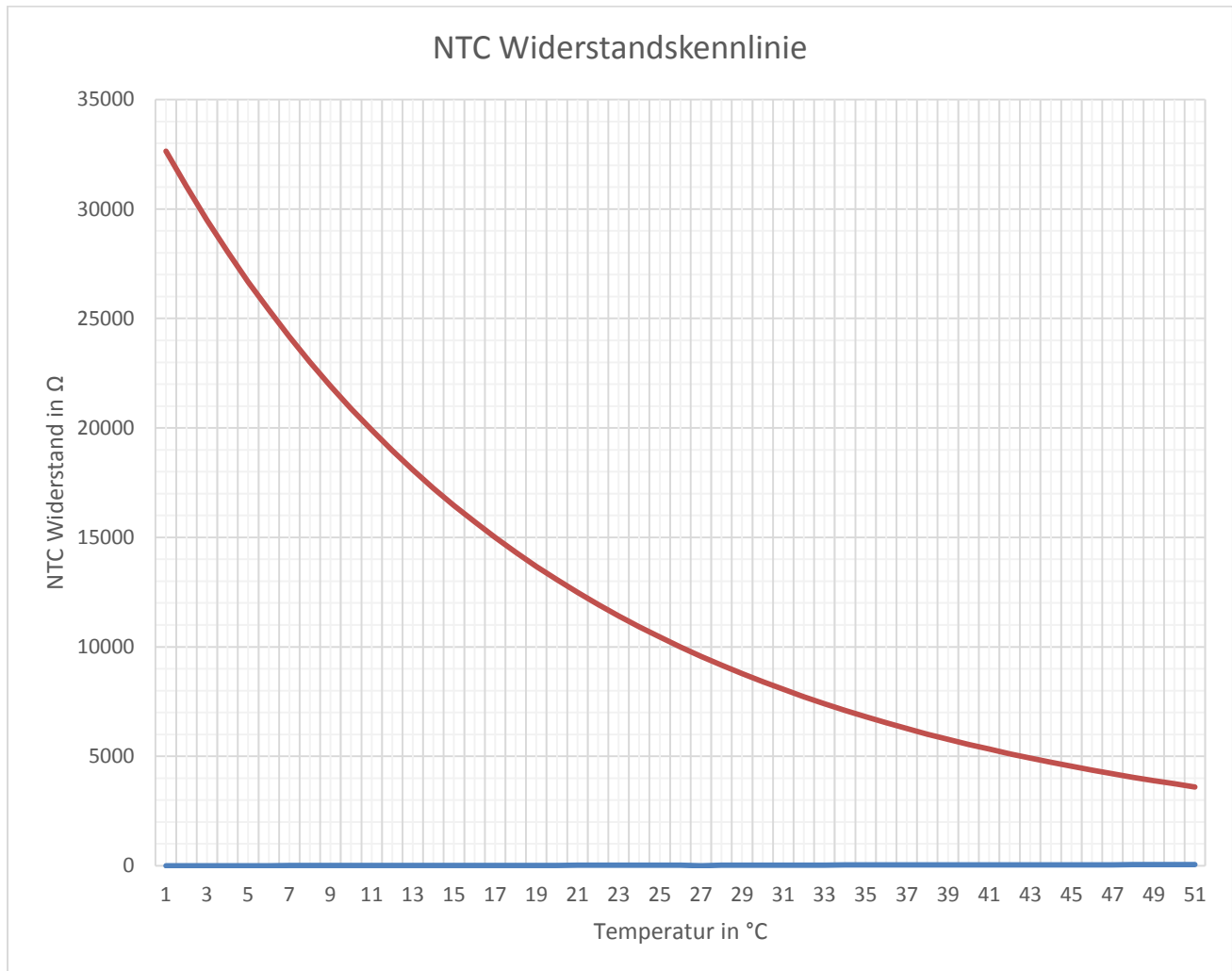


Abbildung 40: Kennlinie des verwendeten NTC-Widerstands

2.14.2.4 Berechnung

Aus der Abbildung 39 und der Spannungsteilerregel ergibt sich folgende Gleichung:

$$U_{NTC} = 3.3[V] \frac{R_{NTC} [\Omega]}{R_{NTC} + R_V [\Omega]} \quad \dots \text{Spannungsteilerregel}$$

Im Entwicklungssystem mbed wird das Ergebnis aus dem Analog-Digital-Umsetzers normiert. Der Wertebereich ist somit $0 \leq ADU - \text{Wert} \leq 1$, definiert als Float-Wert. Damit ist die Software leicht portierbar, der Wert ist unabhängig von der Versorgungsspannung und der Auflösung der Analog-Digital-Umsetzers.

$$\text{Daraus folgt: } U_{NTC} = 3.3[V] \frac{R_{NTC} [\Omega]}{R_{NTC} + R_V [\Omega]} = 1[V] \frac{R_{NTC} [\Omega]}{R_{NTC} + R_V [\Omega]} \Rightarrow R_{NTC} [\Omega] = \frac{ADWert * R_V}{1 - ADWert}$$

Aus der Gleichung (Abbildung 38) $R_T = R_N \cdot e^{B \left(\frac{1}{T} - \frac{1}{T_N} \right)}$ wird durch Umformung:

$$B * \left(\frac{1}{T} - \frac{1}{T_N} \right) = \ln \left(\frac{R_T}{R_N} \right) \text{ und daraus die gesuchte Temperatur:}$$

$$T[K] = \frac{1}{\frac{\ln \left(\frac{R_T[\Omega]}{R_N[\Omega]} \right)}{B} + \frac{1}{T_N[K]}}$$

Hinweis: Im obigen Diagramm ist nur ein Teilbereich des möglichen Temperaturmessbereiches dargestellt. Der vollständige Messbereich dieses NTC-Widerstandes ist von -55 °C bis +155 °C. Der nichtlineare Zusammenhang zwischen Temperatur und Widerstand ist aus der Kennlinie ersichtlich.

Der Kondensator C1 (100 nF, Keramik) ist in Verbindung mit dem ADC-Eingang des Microcontrollers (Sample&Hold) sehr wichtig. Die Spannung wird im Pulsbetrieb gemessen

2.14.3 Beispiel NTC

2.14.3.1 Allgemeines

In diesem Beispiel wird ein NTC der Firma EPCOS S863¹⁴ mit 10 K Ω / 25 °C verwendet.

Der B-Wert beträgt 3988 (berechnet aus den Punkten 25 und 100 °C). Um das Beispiel einfach zu halten, werden keine Filter (Mittelwertbildungen) implementiert.

2.14.3.2 Programm

```

/* ***** */
/* ***** BULME GRAZ, Temperaturmessung mit NTC Widerstand ***** */
/* ***** Abteilung Elektronik und Technische Informatik / Humer ***** */
#include "mbed.h"          // Einbindung der mbed Library

DigitalOut sconhi(P0_2);   // P0.2=Ausgang (Umschalter für USB-UART)
Serial pc(P0_19, P0_18);  // TxD, RxD Definition der Portleitung
AnalogIn NTC(P0_22);

/* ***** Variablendefinition ***** */
int RV = 10000;           // Vorwiderstand in Ohm
int BWert = 3988;        // BWert NTC - Widerstand
float rntc;              // Momentaner Widerstand NTC
float adcntc;            // Analogwert von UNTC, 0<=adcntc<=1
float temp;              // Temperatur in Grad Celsius

/* ***** Hauptprogramm ***** */
int main()               // Hauptprogramm
{
    pc.baud(9600);        // Datenübertragungsgeschw. 9600 Bit/sec
    sconhi=1;            // UART an USB verbinden
    wait(5);             // Warte 5 Sekunden
    while(1)             // Endlosschleife Beginn
    {
        adcntc=NTC.read(); // Wert aus dem ADC 0<=adcntc<=1, float
        rntc=((adcntc*RV)/(1.0-adcntc)); //Widerstand RNTC=?
        temp=(1/((log(rntc/RV)/(BWert)+(1/298.15f)))-273.15f); // Temperatur=?
        wait(0.3);
        pc.printf("NTC: %1.3f[V] %5.0f[Ohm] %3.1f[C]\n\r",3.3*adcntc,rntc,temp);
    }                    // end while
}                        // end main

```

¹⁴ RS Best.-Nr.706-2759, Herstellerteilenummer: B57861S0103F045, EPCOS

2.14.3.3 Visualisierung

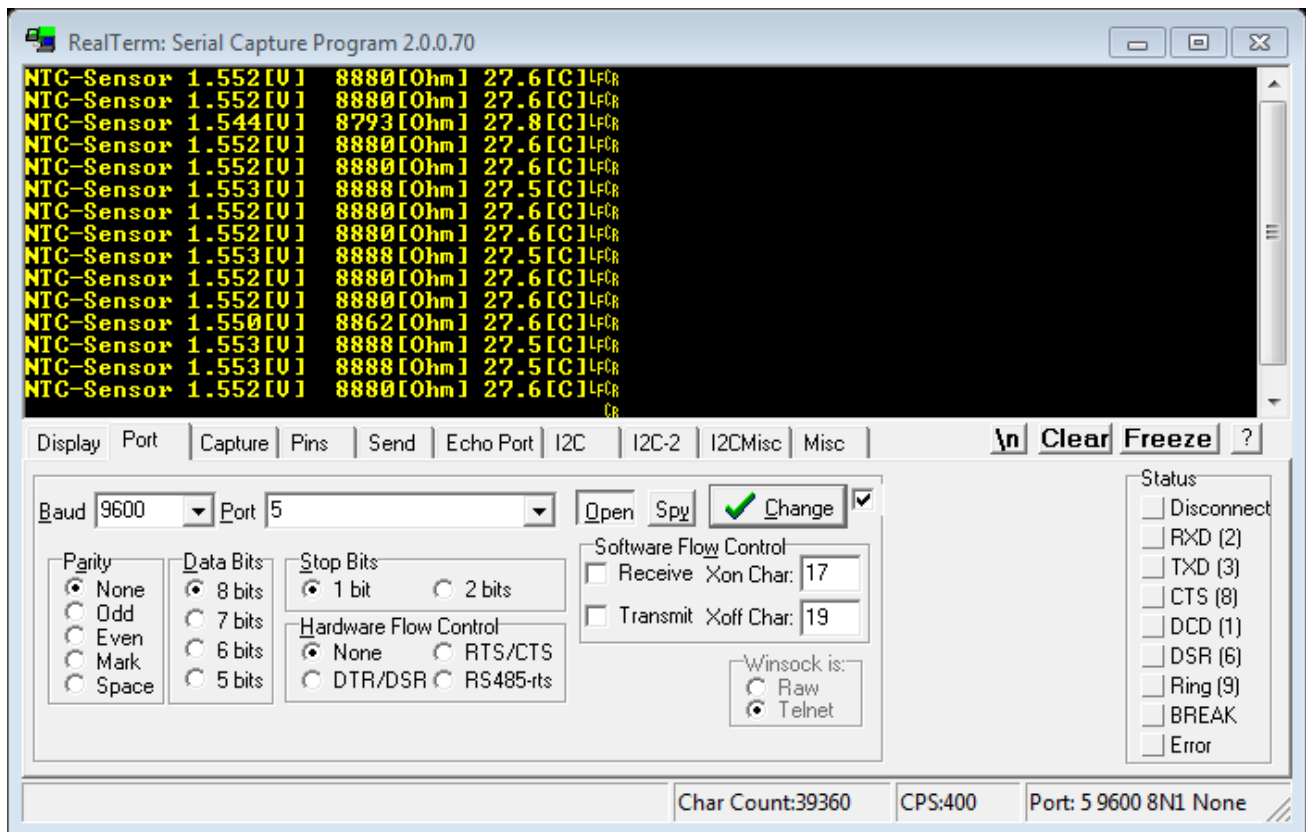


Abbildung 41: Ausgabe der Sensordaten am Bildschirm mit dem Terminalprogramm RealTerm

2.14.4 Vergleich NTC vs LM235

2.14.4.1 Allgemeines

In diesem Beispiel werden die 2 Temperatursensoren (NTC und LM235) gleichzeitig gemessen und bei der Datenausgabe gegenübergestellt. Der NTC ist am Port P0.22 und der LM235 am Port P0.16 angeschlossen. Die Messergebnisse werden nicht gefiltert sondern direkt ausgegeben.

2.14.4.2 Programm

```

/* ***** */
/* ***** BULME GRAZ, Temperaturmessung NTC vs LM235 ***** */
/* ***** Abteilung Elektronik und Technische Informatik / Humer ***** */
#include "mbed.h" // Einbindung der mbed Library

DigitalOut sconhi(P0_2); // P0.2=Ausgang (Umschalter für USB-UART)
Serial pc(P0_19, P0_18); // TxD, RxD Definition der Portleitung
AnalogIn NTC(P0_22); // NTC-Widerstand (als Temperatursensor)
AnalogIn templm235(P0_16); // Temperatursensor LM235
/* ***** Prototypendeklaration ***** */
void init_mcboard();

/* ***** Variablendefinition ***** */
int RV = 10000; // Vorwiderstand in Ohm
int BWert = 3988; // BWert NTC - Widerstand
float rntc; // Momentaner Widerstand NTC
float adcntc; // Analogwert von UNTC, 0<=adcntc<=1
float temp; // Temperatur in Grad Celsius
/* ***** Hauptprogramm ***** */
int main() // Hauptprogramm
{
    init_mcboard();
    while(1) // Endlosschleife Beginn
    {
        adcntc=NTC.read(); // Wert aus dem ADC 0<=adcntc<=1, float
        rntc=((adcntc*RV)/(1.0-adcntc)); //Widerstand RNTC=?
        temp=(1/((log(rntc/RV)/(BWert)+(1/298.15f)))-273.15f); // Temperatur=?
        wait(0.3);
        pc.printf("***** Temperaturmessung *****\n"); //Ausgabe
        pc.printf("NTC: %1.3f[V] %5.0f[Ohm] %3.1f[C]\n\r",3.3*adcntc,rntc,temp);
        pc.printf("Temp:= %3.1f Grad Celsius\n\r",330*templm235.read()-273.15);
    } // end while
} // end main

/* ***** Funktionen ***** */
void init_mcboard()
{
    pc.baud(9600); // Datenübertragungsgeschw. 9600 Bit/sec
    sconhi=1; // UART an USB verbinden
    wait(5); // Warte 5 Sekunden
}

```

2.14.4.3 Ausgabe

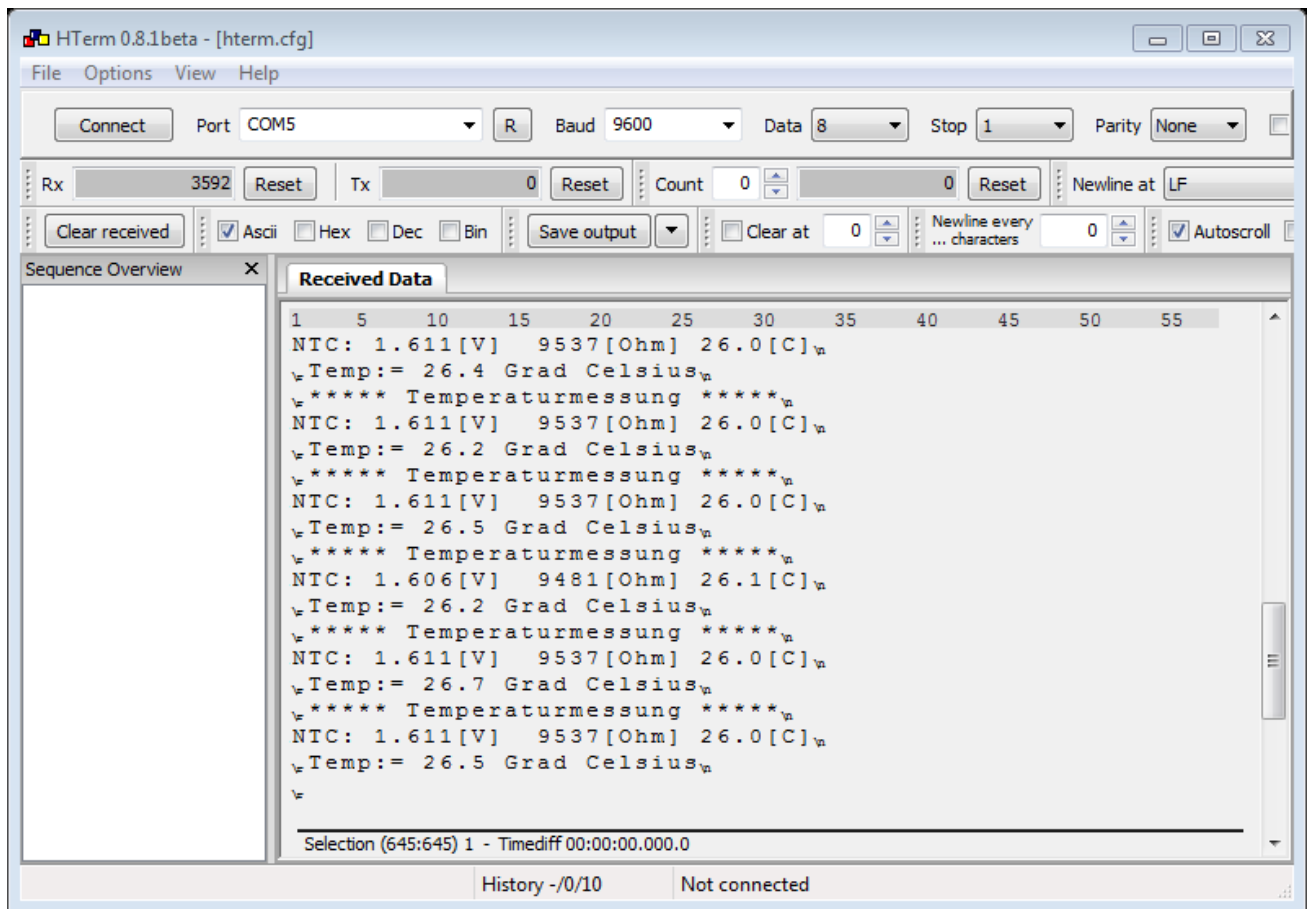


Abbildung 42: Ausgabe Temperaturdaten NTC versus LM235

2.15 RTC

2.15.1 Allgemeines

Die RTC¹⁵ ist in dieser Microcontrollerreihe bereits integriert und braucht als externe Beschaltung nur mehr einen Uhrenquarz mit den notwendigen 2 Kondensatoren. Als Basis dient der „UNIX Timestamp“. Die Versorgung der RTC ist über VB verknüpft (Akku, ohne Bestückung mit ca.3,5 V). Entsprechende Bibliotheksfunktionen sind in der mbed Library bereits integriert.

2.15.2 Datum und Uhrzeit

```

/* ***** */
/* ***** BULME GRAZ, Interne Echtzeituhr (RTC) ***** */
/* ***** Abteilung Elektronik und Technische Informatik / Humer ***** */
#include "mbed.h"

/* ***** Definitionen ***** */
DigitalOut sconhi(P0_2); // P0.2=Ausgang (Umschalter für USB-UART)
Serial pc(P0_19, P0_18); // TxD, RxD Definition der Portleitung

/* ***** Prototypendeklarationen ***** */
void init_mcboard();

/* ***** Variablendeklarationen ***** */

/* ***** Hauptprogramm ***** */
int main()
{
    init_mcboard();
    set_time(1507852800); // Set RTC time to 13 Oct 2017 00:00:00

    while(1)
    {
        time_t seconds = time(NULL); // Zuweisung Struktur seconds

        pc.printf("Zeit seit 1.1.1970 = %d sec\n\r",seconds); //Ausgabe
            // Ausgabe als normierter Zeichenkette
        pc.printf("Date&Time: %s \n\r",ctime(&seconds));

        wait(1); // Warte 1 Sekunde
    }
}

/* ***** Funktionen ***** */
void init_mcboard()
{
    pc.baud(9600); // Datenübertragungsgeschw. 9600 Bit/sec
    sconhi=1; // UART an USB verbinden
    wait(5); // Warte 5 Sekunden
}

```

¹⁵ Real Time Clock

2.15.3 Ausgabe

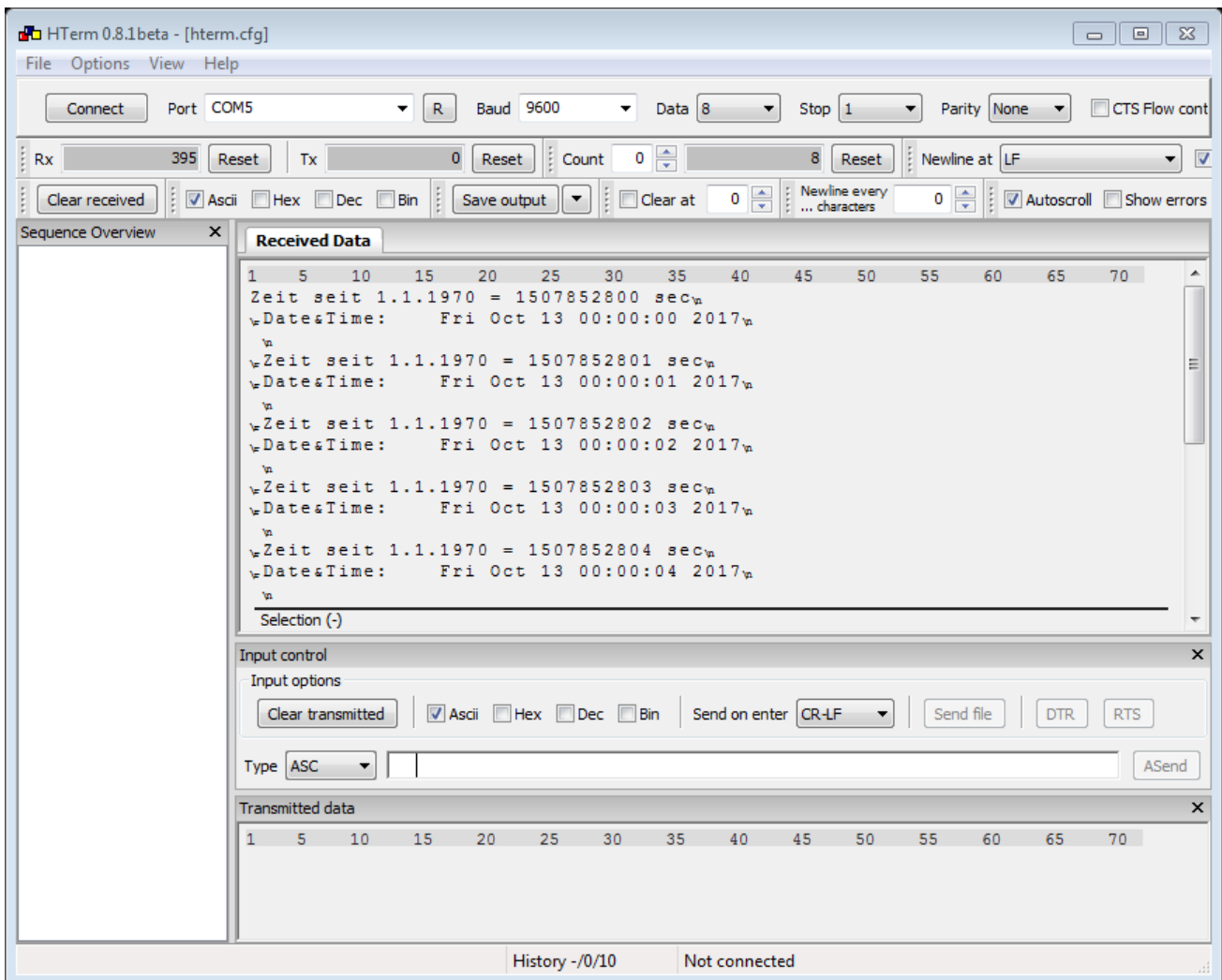


Abbildung 43: Ausgabe von Datum und Uhrzeit

2.15.4 Datum und Uhrzeit stellen

2.15.4.1 Allgemeines

Nachfolgendes Programm stellt eine Erweiterung zum Vorhergehenden dar. Der Quellcode wurde um eine Funktion für die Eingabe von Datum und Uhrzeit erweitert.

```

/* ***** */
/* *****      BULME GRAZ, Interne Echtzeituhr (RTC)      ***** */
/* *****  Abteilung Elektronik und Technische Informatik / Humer  ***** */
#include "mbed.h"

/* *****      Definitionen      ***** */
DigitalOut sconhi(P0_2);          // P0.2=Ausgang (Umschalter für USB-UART)
Serial pc(P0_19, P0_18);         // TxD, RxD Definition der Portleitung

/* *****      Prototypendeklarationen      ***** */
void init_mcboard();
void zeitstellen(void);

/* *****      Variablendeklarationen      ***** */
struct tm t;

/* *****      Hauptprogramm      ***** */
int main()
{
    init_mcboard();              // Initialisierung Microboard
    // set_time(1507852800);     // Set RTC time to 13 Oct 2017 00:00:00
    zeitstellen();               // Einstellung Datum und Uhrzeit

    while(1)
    {
        time_t seconds = time(NULL); // Zuweisung Struktur seconds
        pc.printf("Zeit seit 1.1.1970 = %d sec\n\r",seconds); //Ausgabe
                                   // Ausgabe normierter Zeichenkette
        pc.printf("Date&Time:   %s \n\r",ctime(&seconds));

        wait(1);                 // Warte 1 Sekunde
    }
}

/* *****      Funktionen      ***** */
void init_mcboard()
{
    pc.baud(9600);                // Datenübertragungsgeschw. 9600 Bit/sec
    sconhi=1;                     // UART an USB verbinden
    wait(5);                       // Warte 5 Sekunden
}

void zeitstellen(void)
{
    // get the current time from the terminal
    pc.printf("Geben Sie das aktuelle Datum und Uhrzeit ein:\n");
    pc.printf("YYYY MM DD HH MM SS[enter]\n");
    pc scanf("%d %d %d %d %d %d", &t.tm_year, &t.tm_mon, &t.tm_mday, &t.tm_hour,
&t.tm_min, &t.tm_sec);

                                   // adjust for tm structure required values
    t.tm_year = t.tm_year - 1900;
    t.tm_mon = t.tm_mon - 1;
    set_time(mktime(&t));         // Setzen von Datum und Uhrzeit
}

```


2.15.4.2 Ausgabe

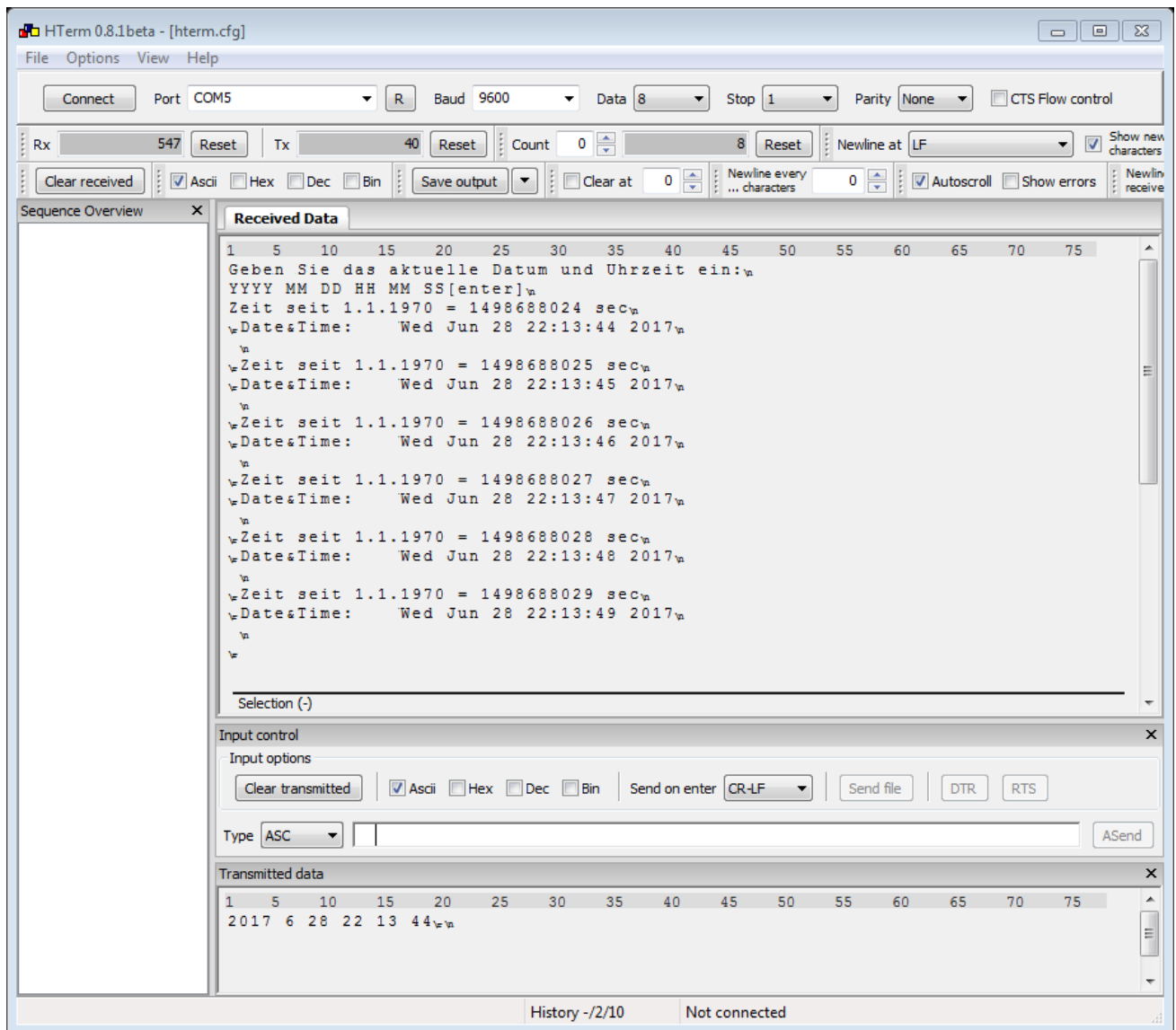


Abbildung 44: Normierte Ausgabe nach einer individuellen Datum und Uhrzeit Eingabe

2.15.5 Individuelle Zeit und Datum Ein-Ausgabe

```

/* *****
/* *****      BULME GRAZ, Interne Echtzeituhr (RTC)      *****
/* *****  Abteilung Elektronik und Technische Informatik / Humer *****
#include "mbed.h"
/* *****      Definitionen      *****
DigitalOut sconhi(P0_2);          // P0.2=Ausgang (Umschalter für USB-UART)
Serial pc(P0_19, P0_18);         // TxD, RxD Definition der Portleitung

/* *****      Prototypendeklarationen      *****
void init_mcboard();
void zeitstellen(void);
void zeit_ausgabe_individuell(void);

/* *****      Variablendeklarationen      *****
struct tm t;
/* *****      Hauptprogramm      *****
int main()
{
    init_mcboard();          // Initialisierung Microboard
    zeitstellen();          // Einstellung Datum und Uhrzeit

    while(1)
    {
        time_t seconds = time(NULL); // Zuweisung Struktur seconds
        pc.printf("Zeit seit 1.1.1970 = %d sec\n\r",seconds); //Ausgabe
            // Ausgabe normierter Zeichenkette
        pc.printf("Date&Time:   %s \n\r",ctime(&seconds));
        pc.printf("Individuelle Ausgabe: *****\n");
        zeit_ausgabe_individuell();
        wait(1);            // Warte 1 Sekunde
    }
}
/* *****      Funktionen      *****
void init_mcboard()
{
    pc.baud(9600);          // Datenübertragungsgeschw. 9600 Bit/sec
    sconhi=1;              // UART an USB verbinden
    wait(5);               // Warte 5 Sekunden
}
void zeitstellen(void)
{
    // get the current time from the terminal
    pc.printf("Geben Sie das aktuelle Datum und Uhrzeit ein:\n");
    pc.printf("YYYY MM DD HH MM SS[enter]\n");
    pc scanf("%d %d %d %d %d %d", &t.tm_year, &t.tm_mon, &t.tm_mday, &t.tm_hour,
&t.tm_min, &t.tm_sec);

            // adjust for tm structure required values
    t.tm_year = t.tm_year - 1900;
    t.tm_mon = t.tm_mon - 1;
    set_time(mktime(&t)); // Setzen von Datum und Uhrzeit
}
void zeit_ausgabe_individuell(void)
{
    pc.printf(" Jahr:= %d \n", t.tm_year+1900);
    pc.printf(" Monat:= %d \n", t.tm_mon+1);
    pc.printf(" Tag:= %d \n", t.tm_mday);
    pc.printf(" Zeit:= %2d:%2d:%2d \n", t.tm_hour,t.tm_min,t.tm_sec);
}

```

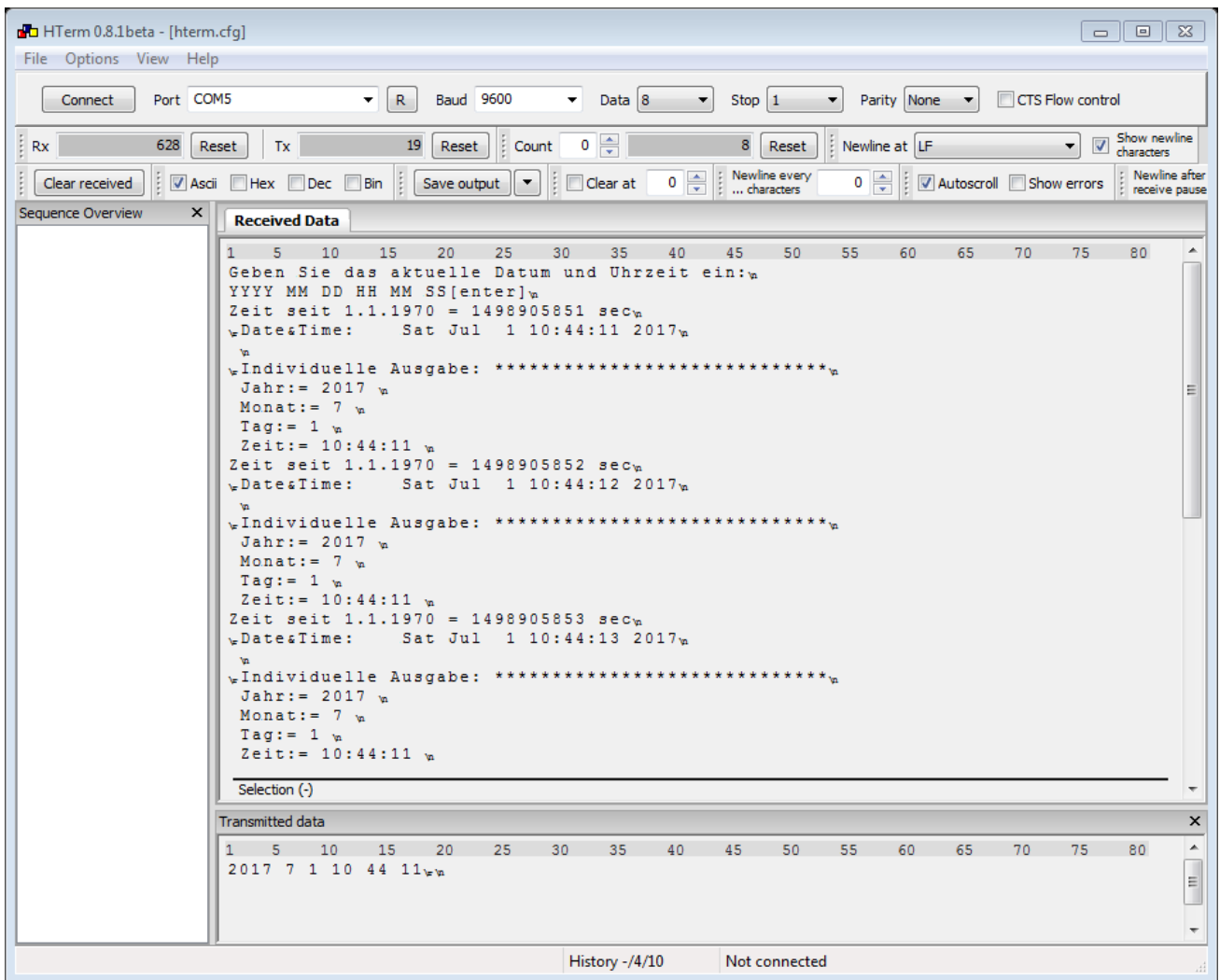


Abbildung 45: Anzeige nach der individuellen Ein- und Ausgabe

2.16 Timer

2.16.1 Allgemeines

In der mbed-Bibliothek sind zahlreiche Timer-Funktionen integriert. Es können mehrere Timer verwendet werden. Die Definition erfolgt mit:

Timer <name>

Ähnlich einer Stoppuhr können folgende Funktionen verwendet werden:

Befehl	Beschreibung
<name>.start()	Starte Timer <name>
<name>.reset()	Reset Timer <name> auf 0
<name>.stop()	Stoppe Timer <name>
<name>.read_us()	Lese die Zeit von Timer in Mikrosekunden
<name>.read_ms()	Lese die Zeit von Timer in Millisekunden
<name>.read_s()	Lese die Zeit von Timer in Sekunden

Tabelle 5: Befehle der mbed-Bibliothek „Timer“

2.16.2 Messung der Dauer einer UART Ausgabe

2.16.2.1 Programm

In diesem Beispiel wird die Dauer der Ausgabe „Hello World“ in Millisekunden gemessen.

```

/* ***** */
/* *****      BULME GRAZ, Zeitmessung einer Ausgabe      ***** */
/* *****      Abteilung Elektronik und Technische Informatik / Humer      ***** */
#include "mbed.h"

/* *****      Definitionen      ***** */
DigitalOut sconhi(P0_2);          // P0.2=Ausgang (Umschalter für USB-UART)
Serial pc(P0_19, P0_18);         // TxD, RxD Definition der Portleitung
Timer minki;

/* *****      Funktionen      ***** */
void init_mcboard();

/* *****      Variablendeklarationen      ***** */

/* *****      Hauptprogramm      ***** */
int main()
{
    init_mcboard();              // Initialisierung Microboard
    minki.reset();               // Timer rücksetzen
    minki.start();               // Timer start
    pc.printf("Hello World\n\r");
    minki.stop();
    pc.printf(" Die Ausgabezeitdauer betrug: %d ms\n\r",minki.read_ms());

    while(1);
}

/* *****      Funktionen      ***** */
void init_mcboard()
{
    pc.baud(9600);                // Datenübertragungsgeschw. 9600 Bit/sec
    sconhi=1;                     // UART an USB verbinden
    wait(5);                       // Warte 5 Sekunden
}

```

Bei der Baudrate von 9600 bit/Sekunde ist die Dauer der Ausgabe 11 ms. Bei einer entsprechend höheren Baudrate verkürzt sich die Ausgabezeit.

2.16.2.2 Ausgabe

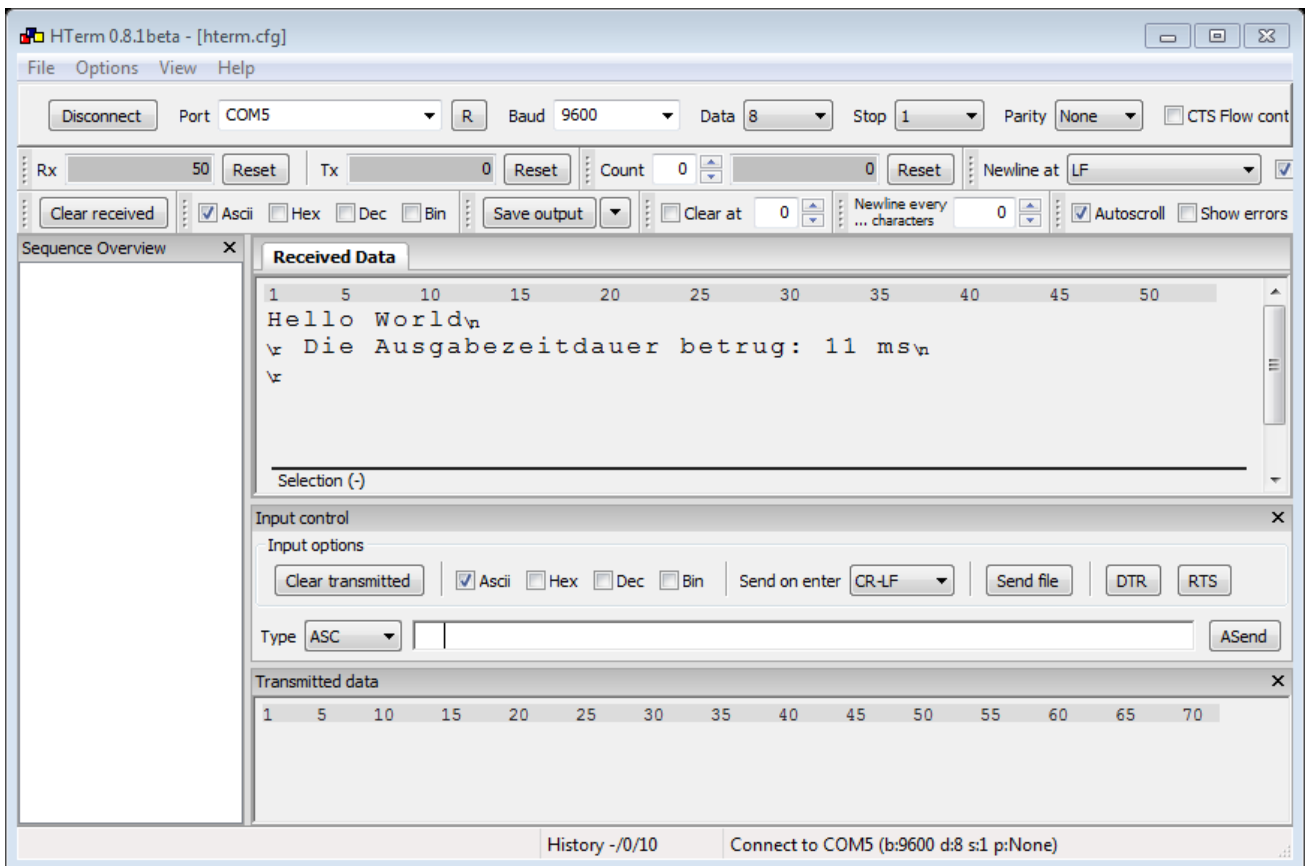


Abbildung 46: Anzeige der Zeitdauer für die Ausgabe

2.16.3 Messung einer PWM Spannung

2.16.3.1 Signalgenerierung

In diesem Beispiel wird eine PWM Spannung mit einer Frequenz von 100 Hz und einem dc von 33 % gemessen. Die Periode ist dabei entsprechend der 100 Hz, 10 ms. Die zu messende Spannung wird in einem Funktionsgenerator eingestellt.

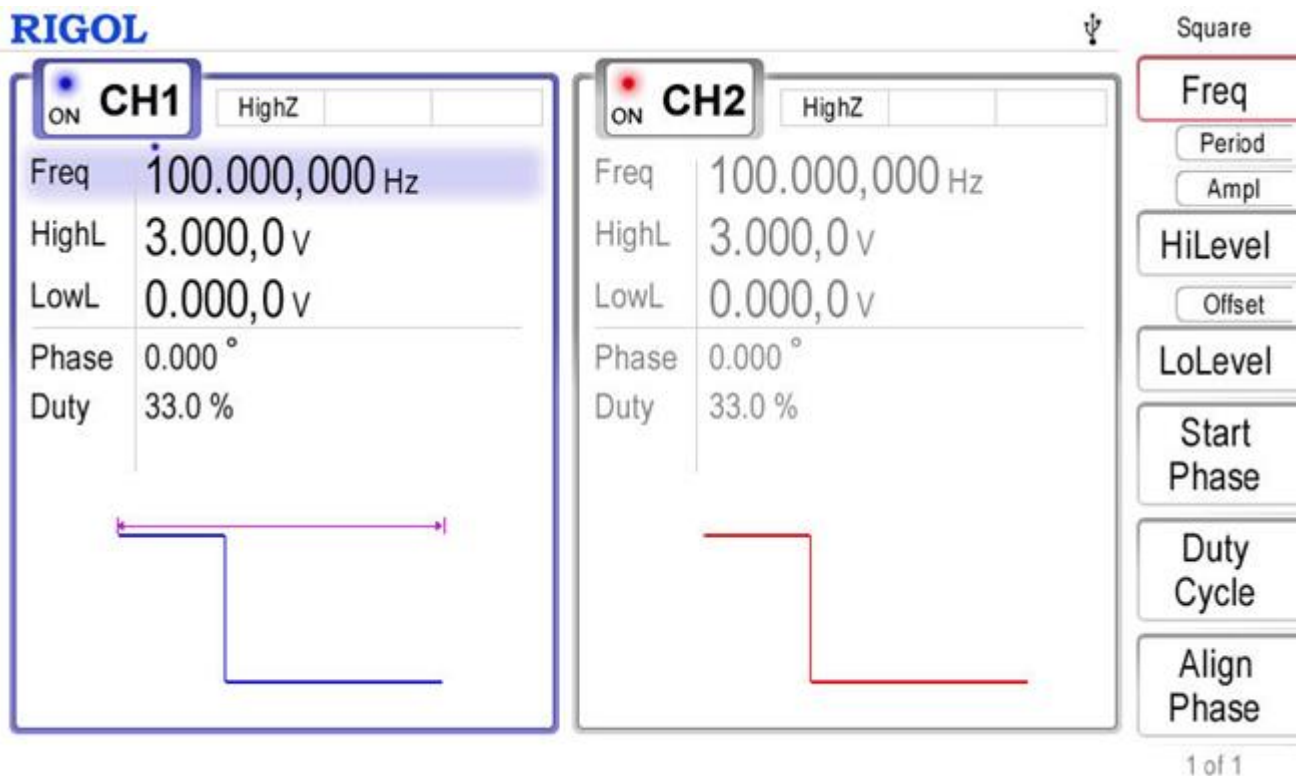


Abbildung 47: Signalgenerierung des gewünschten PWM-Signals am Funktionsgenerator

Die Rechteckspannung hat dabei folgende Eckdaten:

- High-Level = 3 V
- Periode = 10 ms
- Frequenz = 100 Hz
- Hightime = $dc^{16} = 33 \%$

¹⁶ dc = duty cycle

2.16.3.2 Kontrolle des Eingangssignals

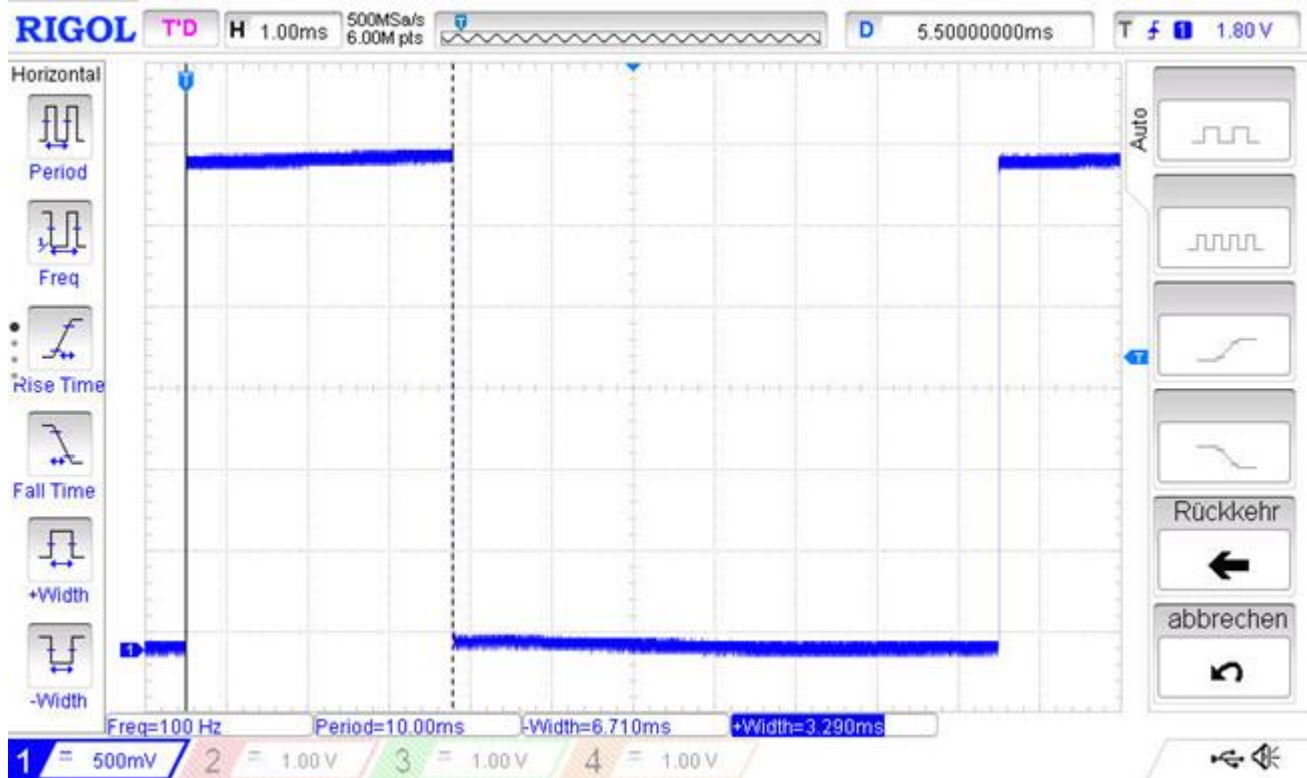


Abbildung 48: Messung des PWM Signals mit dem Oszilloskop

Das PWM-Signal wird mit dem Port P1.13 (Pin-Nummer 30 de Microboards) verbunden und als Interrupt-Eingang definiert. Im nachfolgenden Beispiel wird die Periode und die High-Time in Microsekunden gemessen und die gemessenen Werte über UART an den PC geschickt. Die Baudrate wird auf 115200 baud erhöht.

2.16.3.3 Programm

```

/* *****
/* *****      BULME GRAZ, Messung einer PWM Eingangsgrösse      *****
/* *****  Abteilung Elektronik und Technische Informatik / Humer *****
#include "mbed.h"

/* *****      Definitionen      *****
DigitalOut sconhi(P0_2);          // P0.2=Ausgang (Umschalter für USB-UART)
Serial pc(P0_19, P0_18);         // TxD, RxD Definition der Portleitung
InterruptIn tom(P1_13);          // PWM-Eingangssignal als Interrupt
Timer minki;                     // Timerfunktion mbed-Library
/* *****      Funktionen      *****
void init_mcboard();             // Initialisierung des Microboards

/* *****      Variablendeklarationen      *****
long hightime;                   // Variable für die hightime
long period;                     // Variable für die Periode des Eingangssignals
long counter, counter_old;       // Zwischenvariablen

/* *****      Interruptfunktion      *****
void trigger()
{
    counter = minki.read_us();    // Momentaner Zählerstand
    period=counter-counter_old;   // Ermittlung der Periode
    while(tom==1);               // Warte hightime ab
    hightime=minki.read_us()-counter; // Ermittlung der hightime
    counter_old=counter;         // Speichern des Zählerstandes für nächste
                                // Messung
    printf("PWM, periode=%d us, ontime=%d us\n\r",period,hightime); // Ausgabe
}

/* *****      Hauptprogramm      *****
main()
{
    init_mcboard();              // Initialisierung Microboard

    while(1);
}

/* *****      Funktionen      *****
void init_mcboard()
{
    pc.baud(115200);             // Datenübertragungsgeschw. 9600 Bit/sec
    sconhi=1;                    // UART an USB verbinden
    wait(5);                     // Warte 5 Sekunden
    minki.reset();               // Timer rücksetzen
    minki.start();               // Timer start
    tom.rise(&trigger);          // Interrupt steigende Flanke
}

```

2.16.3.4 Anzeige der Messdaten

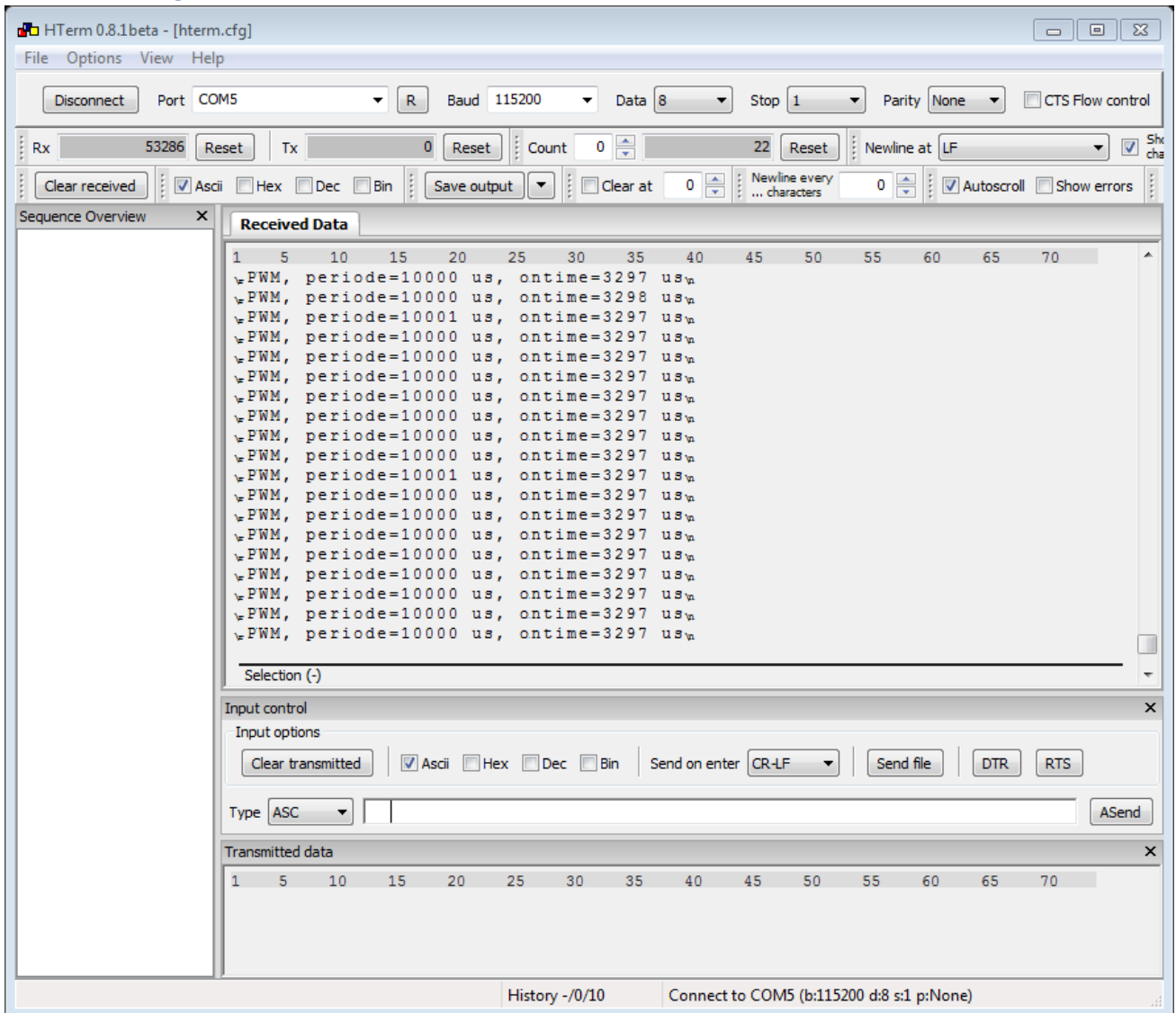


Abbildung 49: Anzeige der PWM Messung 100 Hz / dc=33 %

3 Anhang

3.1 Abbildungsverzeichnis

Abbildung 1: Oberseite des Microboards © Foto Schönauer (SHO)	4
Abbildung 2: Unterseite des Microboards © Foto Schönauer (SHO)	4
Abbildung 3: Anschlussbelegung des Microboards	5
Abbildung 4: Der Microcontroller LPC11U68-48PIN mit Anschlussdefinitionen ...	6
Abbildung 5: Microcontroller LPC11U68 - LQFP48 pinning	7
Abbildung 6: Taktversorgung für den Microcontroller und der Echtzeituhr (RTC)	8
Abbildung 7: Pin-Belegung des USB-Mini Anschlusses	8
Abbildung 8: USB-UART-Controller der Firma FTDI	9
Abbildung 9: Umschaltung UART-USB durch den PortPin P0.2	9
Abbildung 10: Schaltung für RESET und BOOT (für Programmdownload)	10
Abbildung 11: Tastenfunktion für RESET und BOOT (für Programmdownload)	10
Abbildung 12: Schaltung für den 3,3 V Regler	11
Abbildung 13: Schaltung für das Lademanagement einer LIPO-Zelle.	11
Abbildung 14: Anschlussbelegung des DIL40 – Connectors	12
Abbildung 15: Bestückungsplan oben und unten (nicht maßstabsgetreu)	13
Abbildung 16: Darstellung des Layouts oben und unten, ohne Masseflächen (nicht maßstäblich)	14
Abbildung 17: Schaltplan für den Aufbau auf einem Protoboard, R1 und D1 werden laut Schaltplan verdrahtet.	16
Abbildung 18: Darstellung des Ausgangssignals P2.2 an einem Oszilloskop	17
Abbildung 19: Aufbau eines 4 Bit Laufflichtes am Protoboard	18
Abbildung 20: Aufbau eines 4 Bit Zählers am Protoboard	19
Abbildung 21: Aufbau der Schaltung am Protoboard.....	20
Abbildung 22: Ausgangspegel (4 Bit) im 200ms Zeitraster:.....	21
Abbildung 23: Aufbau am Protoboard, Blinklicht 1 Hz	22
Abbildung 24: Visualisierung am PC mit Hilfe der Software HTerm	24
Abbildung 25: Visualisierung am PC mit HTerm	26
Abbildung 26: Visualisierung Ein/Ausgabe am PC.....	28
Abbildung 27: Visualisierung der Eingabe „Maus“ und deren Codes am PC	30
Abbildung 28: Schaltung für den elektronischen Würfel	31
Abbildung 29: Anzeige der Würfelergebnisse am Bildschirm mit einem Terminalprogramm	34
Abbildung 30: 3 Portleitungen für die interne PWM Einheiten.....	35

Abbildung 31: Visualisierung der Signalform am Ausgang eines Oszilloskops ..	36
Abbildung 32: Die Darstellung der 2 PWM Signale am Oszilloskop	37
Abbildung 33: Pin-Belegung Microboard.....	38
Abbildung 34: Versuchsaufbau für eine Spannungsmessung.....	40
Abbildung 35: Ergebnis der Spannungsmessung am Potentiometer.....	41
Abbildung 36: Aufbau der Schaltung mit dem IC LM235 auf dem Protoboard..	42
Abbildung 37: Ausgabe der Temperaturmessung	44
Abbildung 38: Zusammenhang zwischen Temperatur und Widerstand (EPCOS)	45
Abbildung 39: Schaltung für die Temperaturmessung mit einem NTC- Widerstand (Photo).....	45
Abbildung 40: Kennlinie des verwendeten NTC-Widerstands.....	47
Abbildung 41: Ausgabe der Sensordaten am Bildschirm mit dem Terminalprogramm RealTerm	50
Abbildung 42: Ausgabe Temperaturdaten NTC versus LM235	52
Abbildung 43: Ausgabe von Datum und Uhrzeit	54
Abbildung 44: Normierte Ausgabe nach einer individuellen Datum und Uhrzeit Eingabe	56
Abbildung 45: Anzeige nach der individuellen Ein- und Ausgabe	58
Abbildung 46: Anzeige der Zeitdauer für die Ausgabe.....	61
Abbildung 47: Signalgenerierung des gewünschten PWM-Signals am Funktionsgenerator	62
Abbildung 48: Messung des PWM Signals mit dem Oszilloskop.....	63
Abbildung 49: Anzeige der PWM Messung 100 Hz / dc=33 %	65

3.2 Tabellenverzeichnis

Tabelle 1: Lieferbare Typen des Microcontrollers.....	7
Tabelle 2: Stückliste, RS-Nr. bedeutet Bestell-Nr. der Firma RS-Components (BBG Lieferant)	15
Tabelle 3: Ausgabe für das entsprechende Würfelergebnis	32
Tabelle 4: Widerstandswerte in Abhängigkeit von der Temperatur und NTC-Typ.	46
Tabelle 5: Befehle der mbed-Bibliothek „Timer“	59

3.3 Literaturverzeichnis

3.3.1 Internetquellen

- a) Entwicklungsseite: www.mbed.org
- b) Handbook: <https://developer.mbed.org/handbook>
- c) Microcontroller: www.nxp.com
- d) Datenblatt: http://cache.nxp.com/documents/data_sheet/LPC11U6X.pdf

3.3.2 Bücher

- a) Embedded systems, Bert van Dam, ARM Mikrocontroller, elektor-Verlag, Band 1, 35 Einsteiger-Projekte in C mit dem mbed Board LPC1768 von NXP, ISBN 987-3-89576-262-8, 2012
- b) Embedded systems, Bert van Dam, ARM Mikrocontroller, elektor-Verlag, Band 2, 30 Projekte in C für Fortgeschrittene mit dem mbed Board LPC1768 von NXP, ISBN 987-3-89576-271-0, 2013