

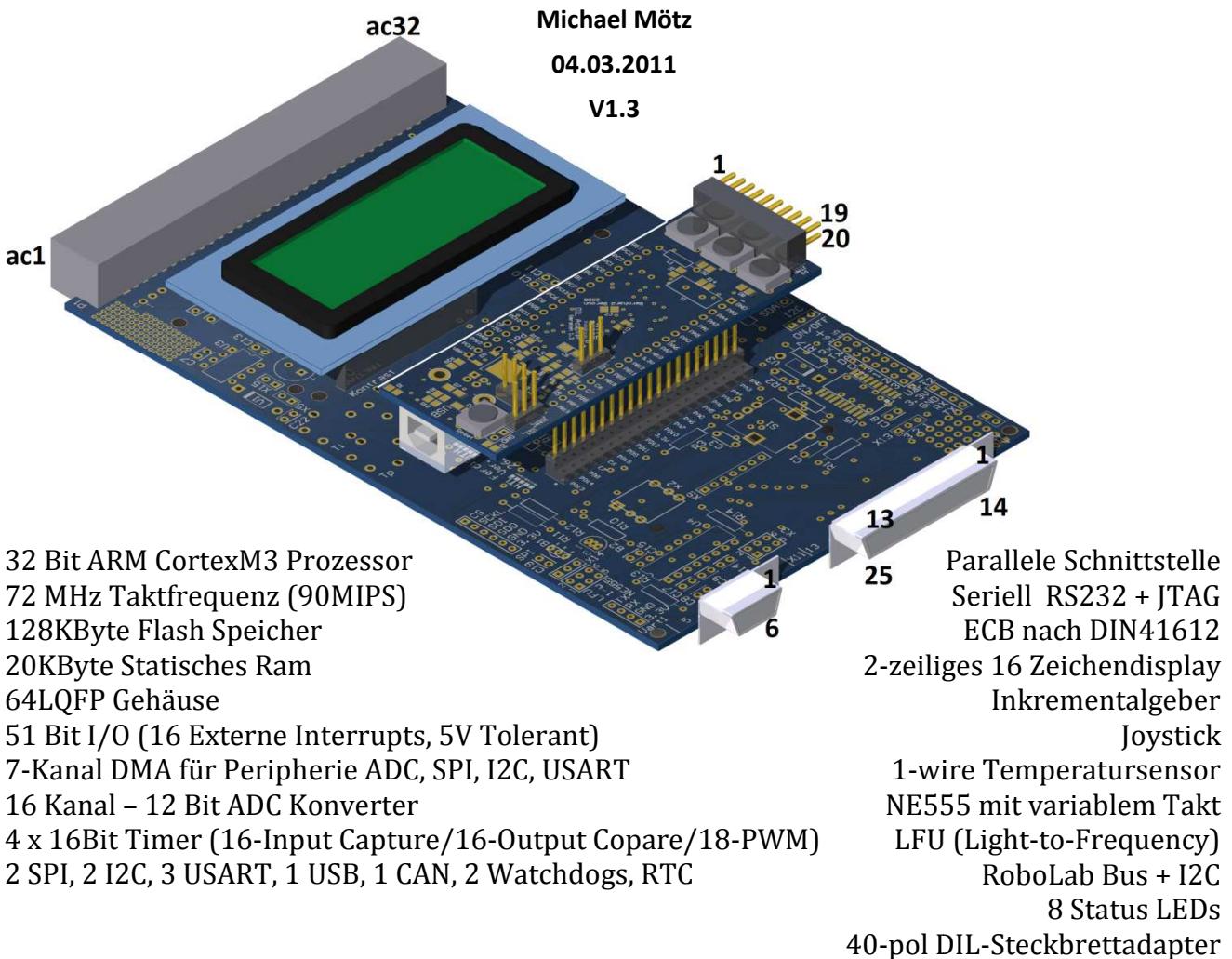
Cortex-M - Success in Market



STM32
ARM Cortex-M3

CM3 Arbeitsunterlagen

STM32F103RBT6 Entwicklungsumgebung



0. Inhaltverzeichnis

0. Inhaltverzeichnis.....	2
1. STM32F103RB	3
1.1 Blockschaltbild aller Platinen	3
1.2 Foto aller Platinen	4
1.3 Blockschaltbild STM32F103RB	5
1.4 Pinning.....	6
1.5 Pin-Belegung.....	7
1.6 GPIOs	9
1.7 Struktur der Dokumente	10
2. Fertigung.....	11
2.1 DIL-Adapter	11
2.1.1 DIL - Stromlaufplan.....	11
2.1.2 DIL - Bestückungsplan	12
2.1.3 DIL - Stückliste	13
2.1.4 DIL - Schaltungsbeschreibung	14
2.2 Europlatine	17
2.2.1 Euro - Stromlaufplan	17
2.2.2 Euro - Bestückungsplan	18
2.2.3 Euro - Stückliste	19
2.2.4 Euro - Schaltungsbeschreibung	20
2.3 LED/Schalter-Platine.....	27
2.3.1 L/S - Stromlaufplan.....	27
2.3.2 L/S - Bestückungsplan.....	28
2.3.4 L/S - Schaltungsbeschreibung.....	29
2.4 V24-Modul.....	30
2.4.1 V24 - Stromlaufplan.....	30
2.4.2 V24 - Bestückungsplan	30
2.4.3 V24 - Stückliste	30
2.4.4 V24 - Schaltungsbeschreibung	31
3. Inbetriebnahme.....	32
3.1 Installation uVision4	32
3.2 uVision4-Projekt erstellen	35
3.3 HTL-Platinen testen	43
3.4 Funktionen in Library.....	49
3.5 Debuggen eines C-Programms	50
3.6 Über UART flashen	55
3.7 Über USB flashen.....	59
3.8 Hex-Datei flashen	61
4. FAQ	63

Danksagung:

Dieses 32-bit Lehr- und Übungssystem ist entstanden durch die Mithilfe von:

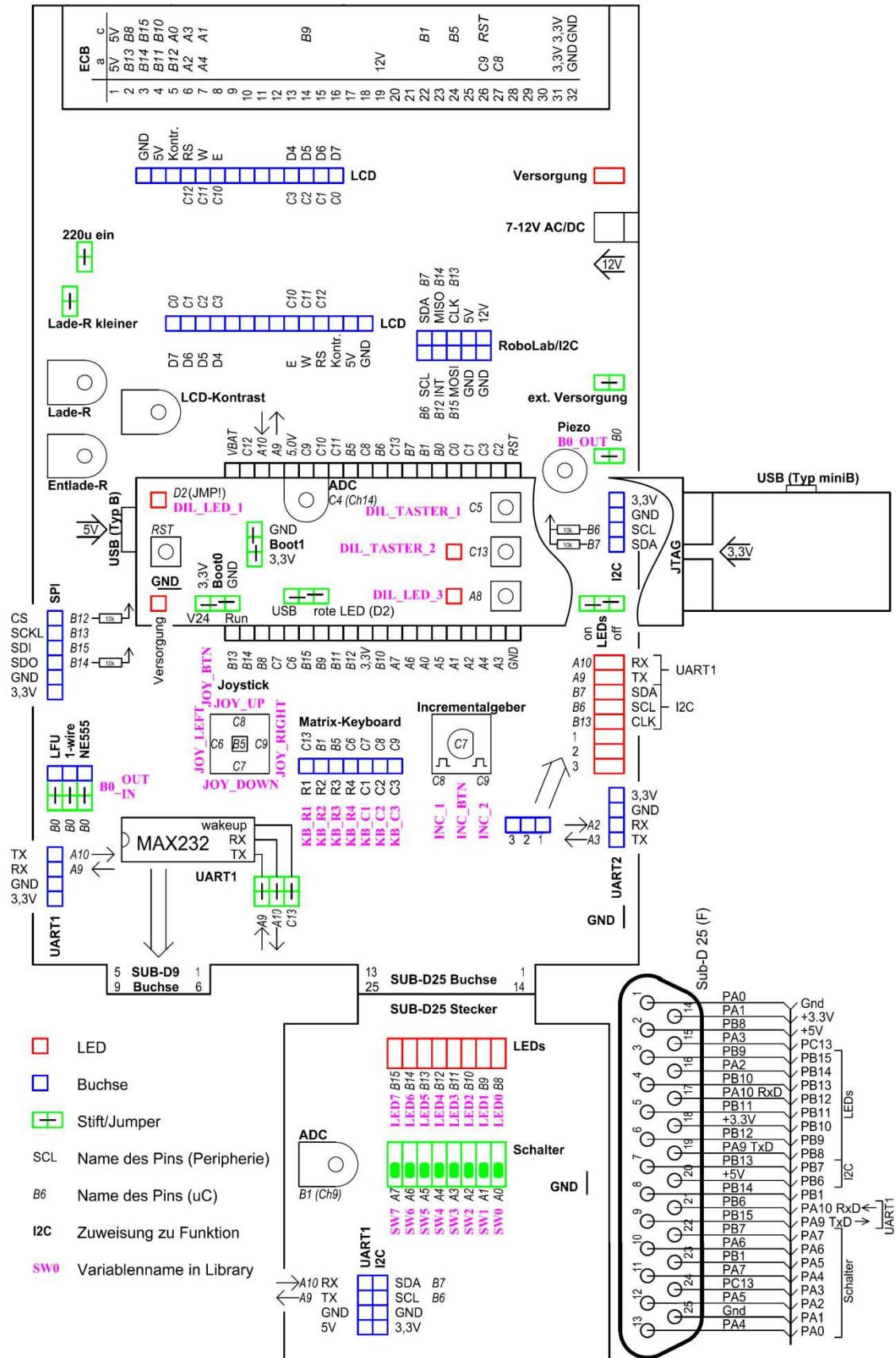
Bernhard Beroun dem im Zivildienst "sooo faaad" war, dass er den Steckbrett DIL Adapter entwickelt hat, **Ferdinand Wimmer** der neben seinem Internetradio noch die Basis-Europakarte im Altium Designer realisiert hat, und **Daniel Höfenstock** der, obwohl er eine Kransteuerung machen wollte, noch zusätzlich für die LED/Schalter-Platine und die Fertigungsanleitung verantwortlich war.

Manfred Resel im Mai 2010

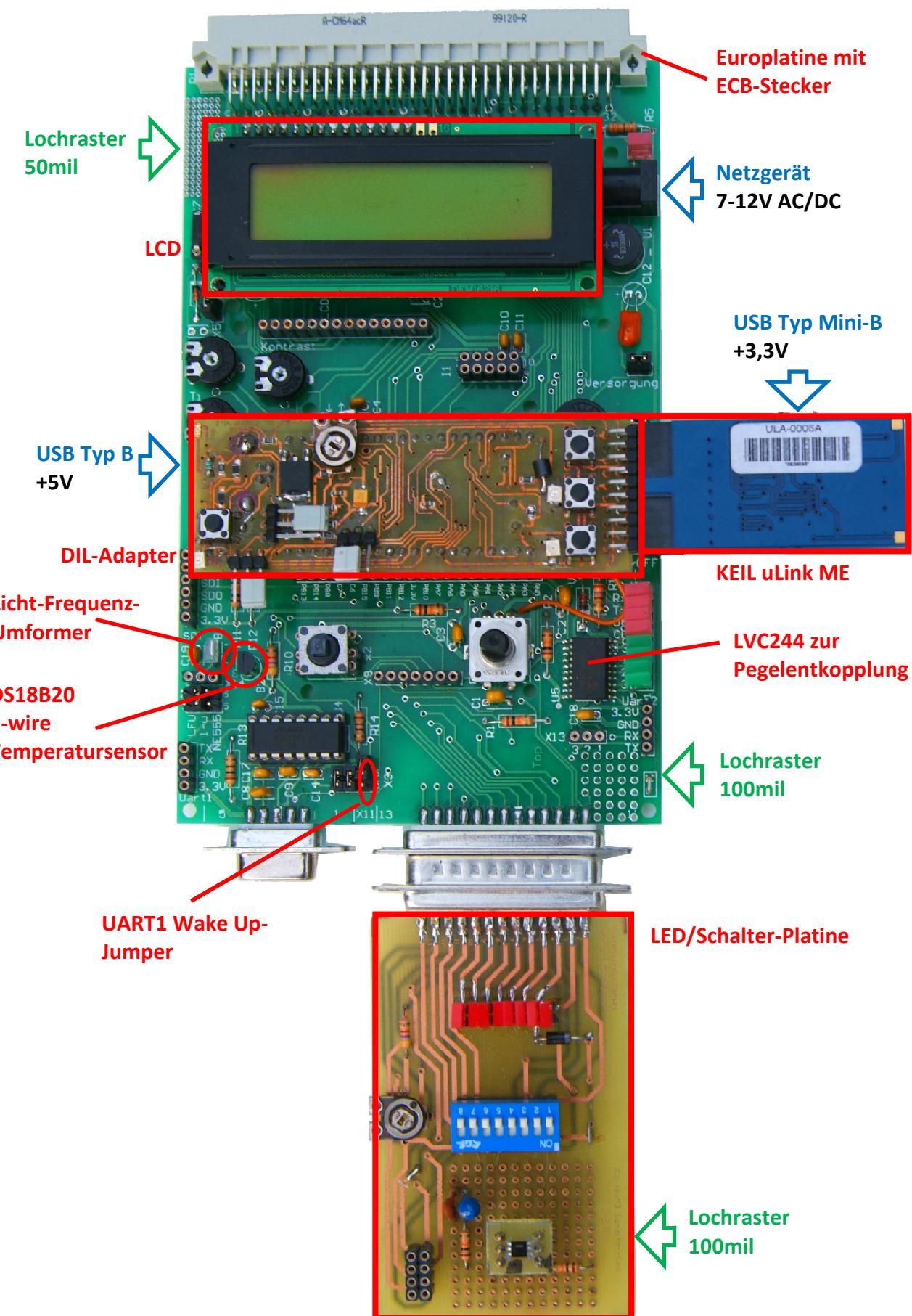
p.s. Ganz besonders bedanken möchte ich mich bei **Wolfgang Kauer** der für die CM3 Fertigung die SMD Werkstätte erweitert hat und bei **Thomas Stiedl** der mit akribischer Genauigkeit die Stücklisten durchforstet, und immer beim günstigsten Lieferanten bestellt hat.

1. STM32F103RB

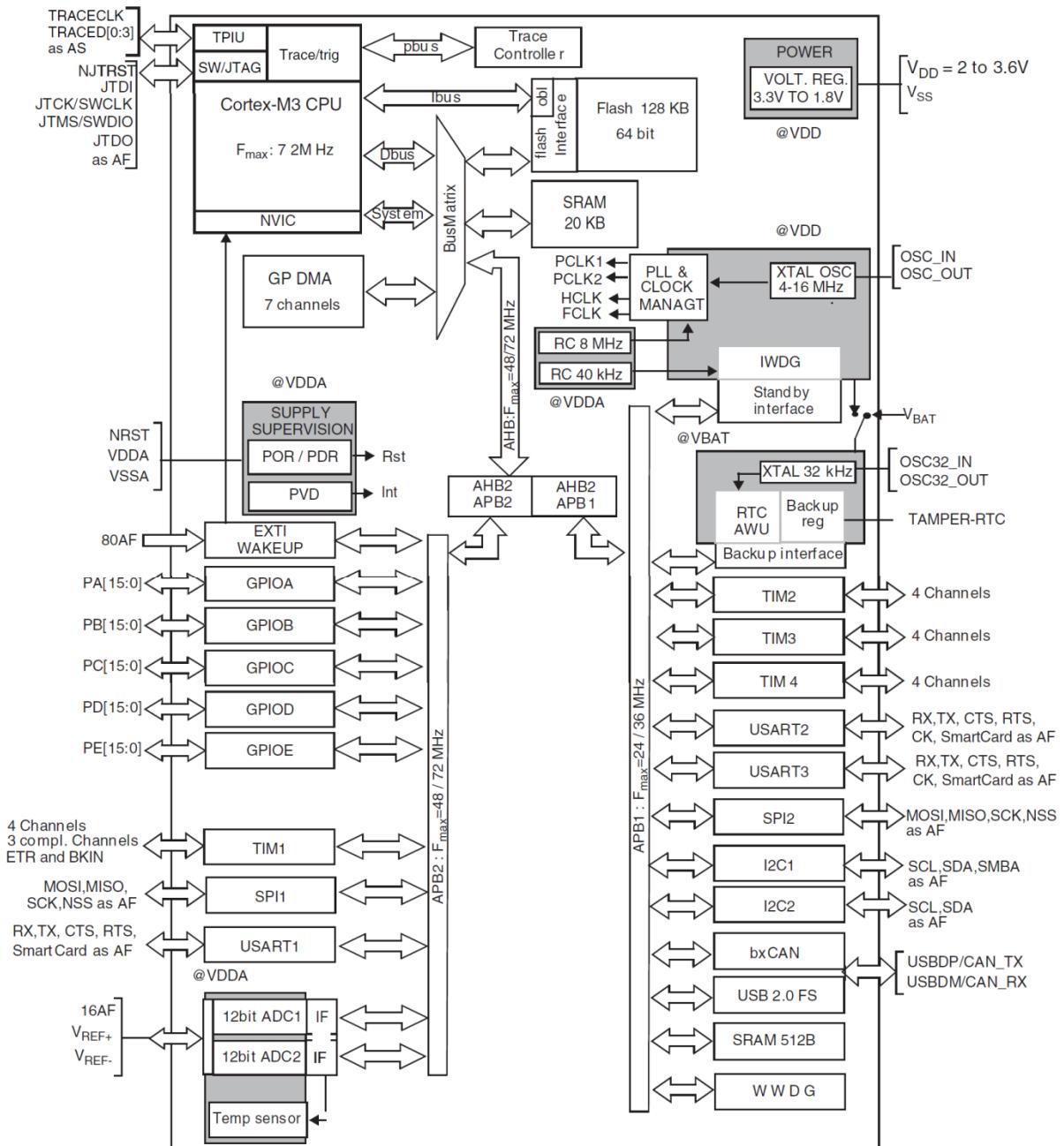
1.1 Blockschaltbild aller Platinen



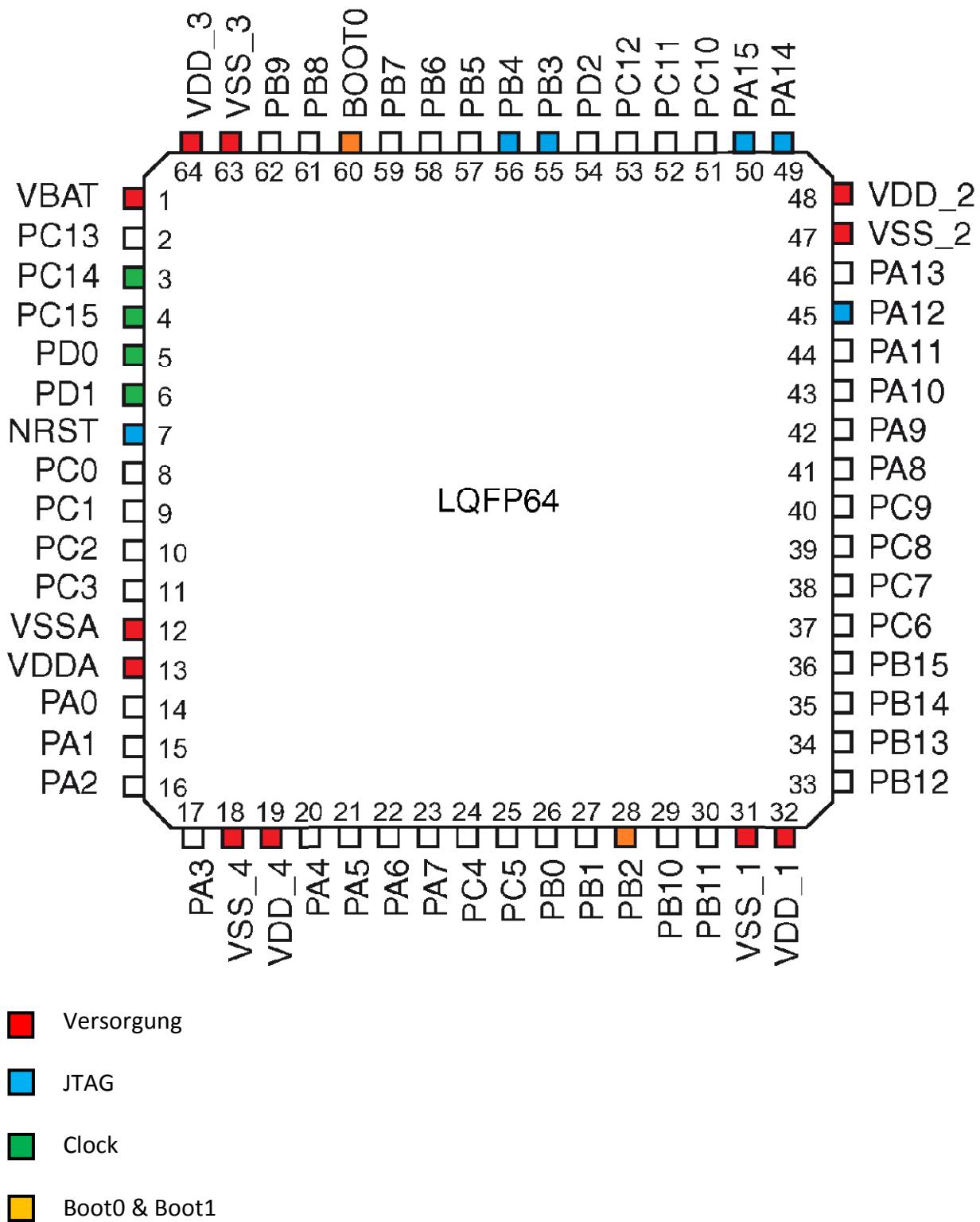
1.2 Foto aller Platinen



1.3 Blockschaltbild STM32F103RB



1.4 Pinning



1.5 Pin-Belegung

Pin	Pin-Name	Typ	5V Tolerant	Funktion auf HTL-Platinen <i>Default function</i>
1	V_{BAT}	V		Versorgung für Low Power-Modes
2	PC13	I/O		DIL-LED/Taster; Matrix-Keyboard (R1); UART1 (wakeup); TAMPER-RTC
3	PC14	I/O		Clock; OSC32_IN
4	PC15	I/O		Clock; OSC32_OUT
5	PD0	I		Clock; OSC_IN
6	PD1	O		Clock; OSC_OUT
7	NRST	I/O		JTAG (NRESET/nSRST)
8	PC0	I/O		LCD (D7); ADC12_IN10
9	PC1	I/O		LCD (D6); ADC12_IN11
10	PC2	I/O		LCD (D5); ADC12_IN12
11	PC3	I/O		LCD (D4); ADC12_IN13
12	V_{SSA}	V		Versorgung
13	V_{DDA}	V		Versorgung
14	PA0	I/O		L/S-Schalter (SW0); WKUP / USART2_CTS / ADC12_IN0 / TIM2_CH1_ETR
15	PA1	I/O		L/S-Schalter (SW1); USART2_RTS / ADC12_IN1 / TIM2_CH2
16	PA2	I/O		L/S-Schalter (SW2); UART2 (TxD); ADC12_IN2 / TIM2_CH3
17	PA3	I/O		L/S-Schalter (SW3); UART2 (RxD); ADC12_IN3 / TIM2_CH4
18	V_{SS_4}	V		Versorgung
19	V_{DD_4}	V		Versorgung
20	PA4	I/O		L/S-Schalter (SW4); SPII_NSS / USART2_CK / ADC12_IN4
21	PA5	I/O		L/S-Schalter (SW5); SPII_SCK / ADC12_IN5
22	PA6	I/O		L/S-Schalter (SW6); SPII_MISO / ADC12_IN6 / TIM3_CH1
23	PA7	I/O		L/S-Schalter (SW7); SPII_MOSI / ADC12_IN7 / TIM3_CH2
24	PC4	I/O		DIL-ADC (Ch14)
25	PC5	I/O		DIL-LED; ADC12_IN15
26	PB0	I/O		LFU; 1-wire; NE555; Piezo; ADC12_IN8 / TIM3_CH3
27	PB1	I/O		L/S-ADC (Ch9); Matrix-Keyboard (R2); ADC12_IN9 / TIM3_CH4
28	PB2	I/O	X	Boot1
29	PB10	I/O	X	L/S-LED (LED2); I2C2_SCL / USART3_TX
30	PB11	I/O	X	L/S-LED (LED3); I2C2_SDA / USART3_RX
31	V_{SS_1}	V		Versorgung
32	V_{DD_1}	V		Versorgung
33	PB12	I/O	X	L/S-LED (LED4); SPI_CS; RoboLab_INT; I2C2_SMBA / USART3_CK / TIM1_BKIN
34	PB13	I/O	X	L/S-LED (LED5); SPI_SCKL; RoboLab_CLK; USART3_CTS / TIM1_CHIN
35	PB14	I/O	X	L/S-LED (LED6); SPI_SDO; RoboLab_MISO; USART3_RTS / TIM1_CH2N
36	PB15	I/O	X	L/S-LED (LED7); SPI_SDI; RoboLab_MOSI; TIM1_CH3N
37	PC6	I/O	X	Joystick (Left); Matrix-Keyboard (R4)
38	PC7	I/O	X	Joystick (Down); Matrix-Keyboard (C1); Inkrementalgeber (Taste)

39	PC8	I/O	X	Joystick (Up); Matrix-Keyboard (C2); Inkrementalgeber (INC_1)
40	PC9	I/O	X	Joystick (Right); Matrix-Keyboard (C3); Inkrementalgeber (INC_2)
41	PA8	I/O	X	DIL-LED/Taster; <i>USART1_CK / TIM1_CH1 / MCO</i>
42	PA9	I/O	X	UART1 (TxD); <i>TIM1_CH2</i>
43	PA10	I/O	X	UART1 (RxD); <i>TIM1_CH3</i>
44	PA11	I/O	X	USB (-); <i>USART1_CTS / CANRX / TIM1_CH4</i>
45	PA12	I/O	X	USB (+); <i>USART1_RTS / CANTX / TIM1_ETR</i>
46	PA13	I/O	X	JTAG (TMS)
47	V _{SS_2}	V		Versorgung
48	V _{DD_2}	V		Versorgung
49	PA14	I/O	X	JTAG (TCK)
50	PA15	I/O	X	JTAG (TDI)
51	PC10	I/O	X	LCD (E)
52	PC11	I/O	X	LCD (W)
53	PC12	I/O	X	LCD (RS)
54	PD2	I/O	X	USB/DIL-LED; <i>TIM3_ETR</i>
55	PB3	I/O	X	JTAG (TDO)
56	PB4	I/O	X	JTAG (nTRST)
57	PB5	I/O		Joystick (Taster); Matrix-Keyboard (R3); <i>I2C1_SMBAI</i>
58	PB6	I/O	X	I2C (SCL); <i>TIM4_CH1</i>
59	PB7	I/O	X	I2C (SDA); <i>TIM4_CH2</i>
60	Boot0	I		Boot0
61	PB8	I/O	X	L/S-LED (LEDO); CAN (RxD); <i>TIM4_CH3</i>
62	PB9	I/O	X	L/S-LED (LED1); CAN (TxD); <i>TIM4_CH4</i>
63	V _{SS_3}	V		Versorgung
64	V _{DD_3}	V		Versorgung

A0-A7 L/S-Schalter **B0** Piezo, NE555/LFU/1-wire
A2, A3 UART2 **B1** Matrix-Keyboard, SUB-D25

A8 DIL-LED/Taster oben **B2** Boot1

A9, A10 UART1 **B3, B4** JTAG

A11, A12 USB **B5** MOSI, Matrix-Keyboard

A13-A15 JTAG **B6, B7** SUBD-25 I2C

B8-B15 L/S-LEDs

B12-B15 SPI, RoboLab

C0-C3 LCD

C4 Poti oben

C5 DIL-LED/Taster oben

C6-C9 Matrix-Keyboard, Joystick, Inkrementalgeber

C10-C12 LCD

C13 Tamper, DIL-LED/Taster oben, Matrix-Keyboard, UART-Wakeup, LVC-LED

C14, C15 32kHz-Quarz

D0, D1 8MHz-Quarz

D2 USB-Enable, DIL-rote LED

1.6 GPIOs

Config-Register: GPIOx_CRy

x = A...D (Port) **y = H/L (Bits 15-8/Bits 7-0)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNF7[1:0]		MODE7[1:0]		CNF6[1:0]		MODE6[1:0]		CNF5[1:0]		MODE5[1:0]		CNF4[1:0]		MODE4[1:0]	
rw	rw	rw	rw												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNF3[1:0]		MODE3[1:0]		CNF2[1:0]		MODE2[1:0]		CNF1[1:0]		MODE1[1:0]		CNF0[1:0]		MODE0[1:0]	
rw	rw	rw	rw												

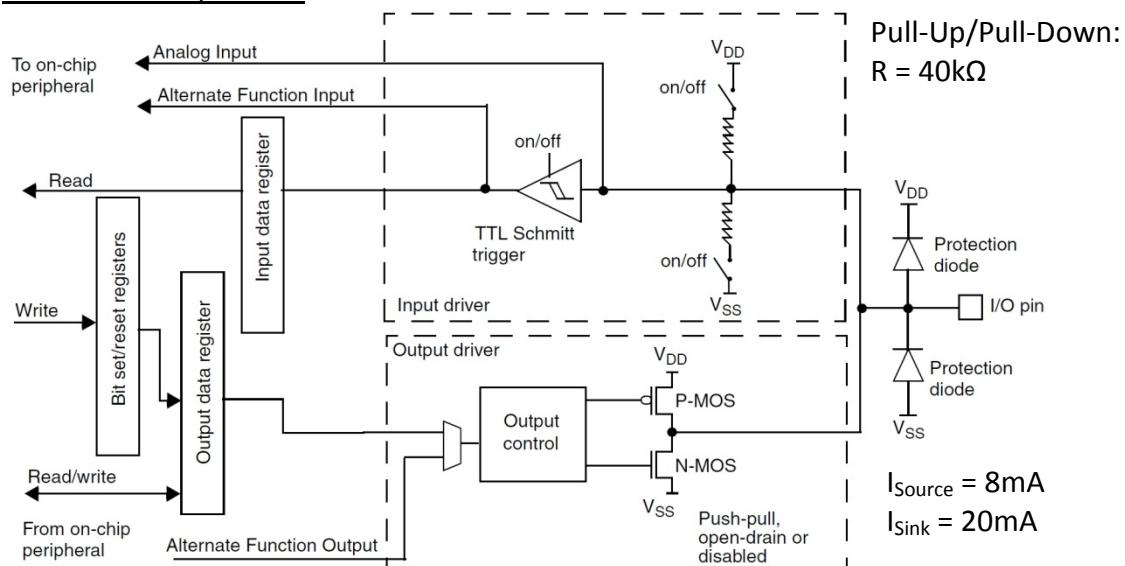
Reset-Wert: 0x4444 4444

Beispiel:

```
GPIOA->CRH &= 0xFFFFFFFF0; //PA8 Konfigurationsbit löschen
GPIOA->CRH |= 0x00000003; //PA8 als Push Pull Output
                           //mit 50Mhz max.Frequenz definieren
```

Configuration mode		CNF1	CNF0	MODE1	MODE0	PxODR register
General purpose output	Push-pull	0	0	MODE [1:0]	Max. Output Speed	0 or 1
	Open-drain		1	01	10 MHz	0 or 1
Alternate Function output	Push-pull	1	0	10	2 MHz	don't care
	Open-drain		1	11	50 MHz	don't care
Input	Analog	0	0	00		
	Input floating		1			
	Input pull-down	1	0	00		
	Input pull-up		0	0		

Struktur eines I/O-Pins:



Über das Register AFIO_MAPR können die Portpins der Timer, der UARTs, des I2C1 und des SPI1 auf andere Pins verlegt werden (z.B. von PA4 auf PA15). Weiters können die Trigger-Eingänge der ADCs konfiguriert werden.

siehe RM0008 S. 170 (8.4.2 AF remap and debug I/O configuration register)

1.7 Struktur der Dokumente

CD-Aufbau:

E:\Dienstag\Doku\CM3-Arbeitsunterlagen_v1.3.pdf
 \Datenblätter\STM32F103xx_Reference_Manual_(ST_RM0008)(2010-04-23).pdf
 \Datenblätter\ STM32F103xB_Datasheet_(ST)(2009-09-22).pdf
 \Datenblätter\STM32F103xx_User_Manual_(ST_UM0427)(Rev4_2008-06-13).pdf
 \Datenblätter\STM32F10xxx_Programming_Manual_(ST_PM005)(2009-10-26).pdf
 \Datenblätter\STM32F10xxx_Flash_Loader_Demonstrator_(ST_UM0462).pdf
 \Datenblätter\uv4_Getting_Started_(Keil).pdf

E:\Dienstag\Programme\MDK411.exe (KEIL uVision4 IDE)
 \Flash_Loader_Demonstrator_v2.2.0_Setup.exe (Flashen über UART)
 \PuTTY_0.6.exe (Terminal-Programm)
 \hypertrm.exe (Hyperterminal 5.1.26)
 \USB-Treiber\DFU\ (Flashen über USB)
 \VCOM\ (Virtueller UART)

E:\Dienstag\Software\VCOM-Test_v1.0\
 \cm3-test_v1.3\

E:\Dienstag\info.txt (Erklärung des CD-Aufbaus)

Viele hilfreiche PDFs und Programmbeispiele gibt es auf der Homepage des Microcontroller-Herstellers:

<http://www.st.com>

- Micros & Memories
- Microcontrollers
- STM32 – 32-bit ARM Cortex MCUs
- STM32F 32-bit MCUs
- STM32F103RB

Unter dem Reiter “Design support” stehen Datenblätter, Dokumentationen, Programmbeispiele, u.v.m. zum Download zur Verfügung.

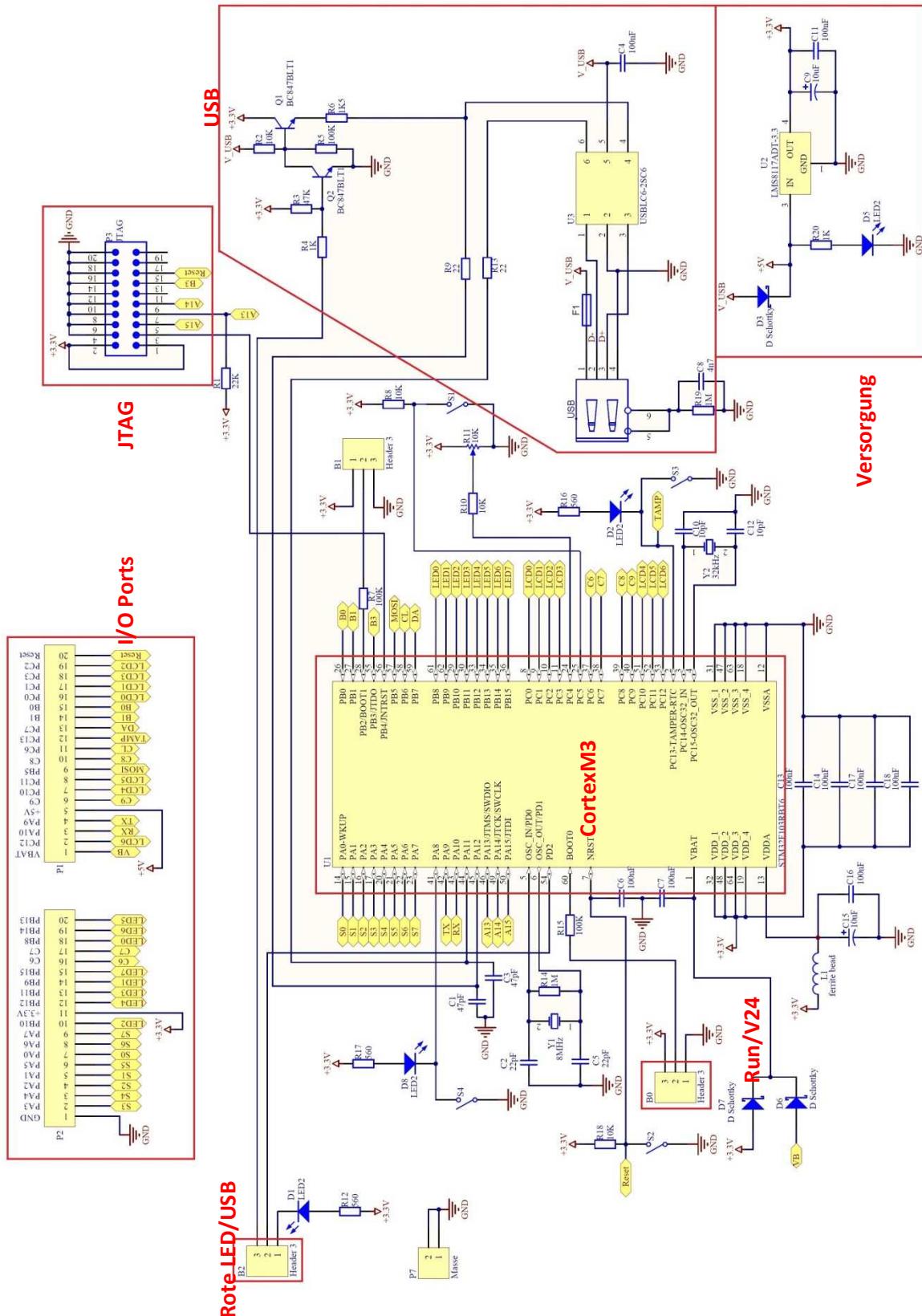
Description	Version	Size
Medium-density performance line ARM-based 32-bit MCU with 64 or 128 KB Flash, USB, CAN, 7 timers, 2 ADCs, 9 communication interfaces	12.0	1431KB

Description	Version	Size
AN2812: Vocoder demonstration using a Speex audio codec on STM32F101xx and STM32F103xx	2.3	275KB

2. Fertigung

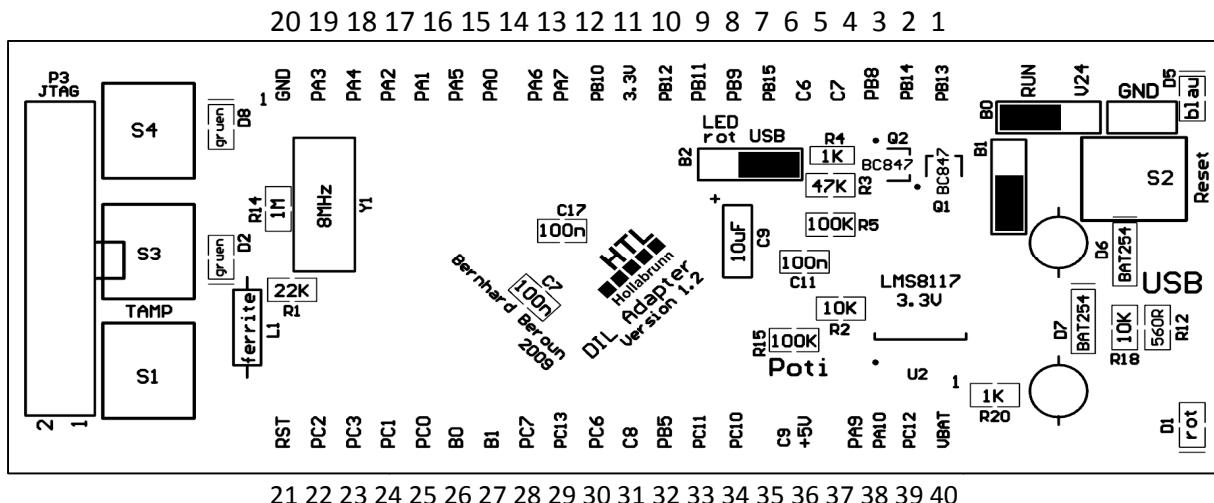
2.1 DIL-Adapter

2.1.1 DIL - Stromlaufplan

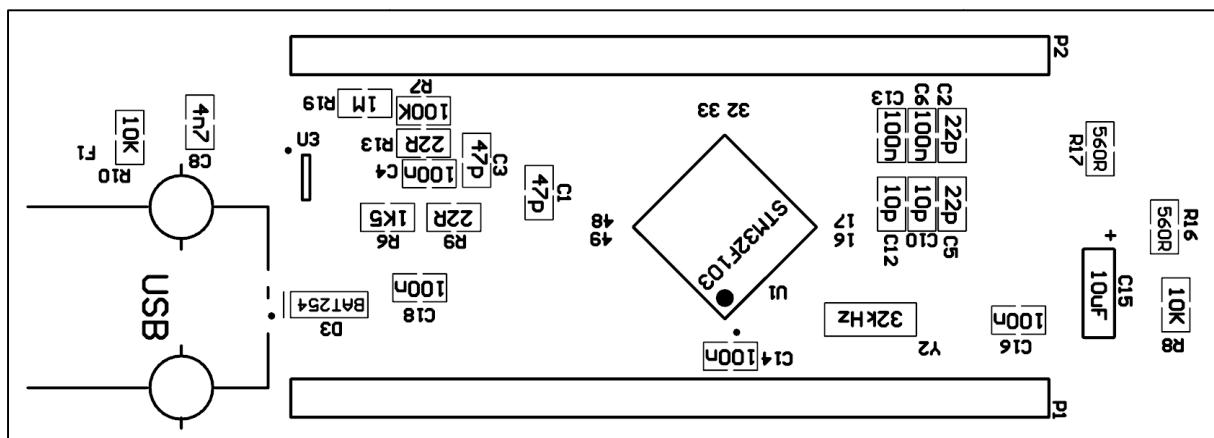


2.1.2 DIL - Bestückungsplan

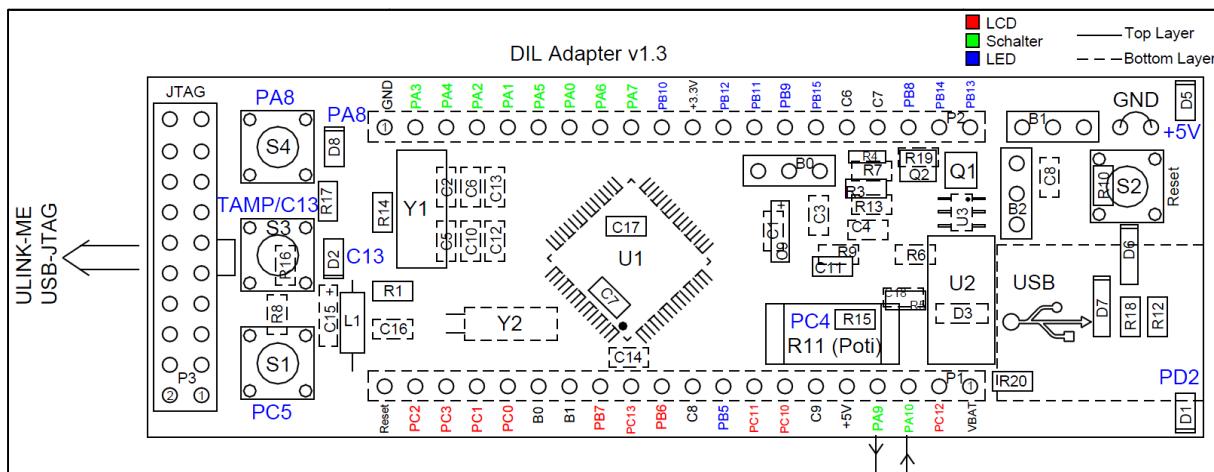
Top:



Bottom:



Gesamt:

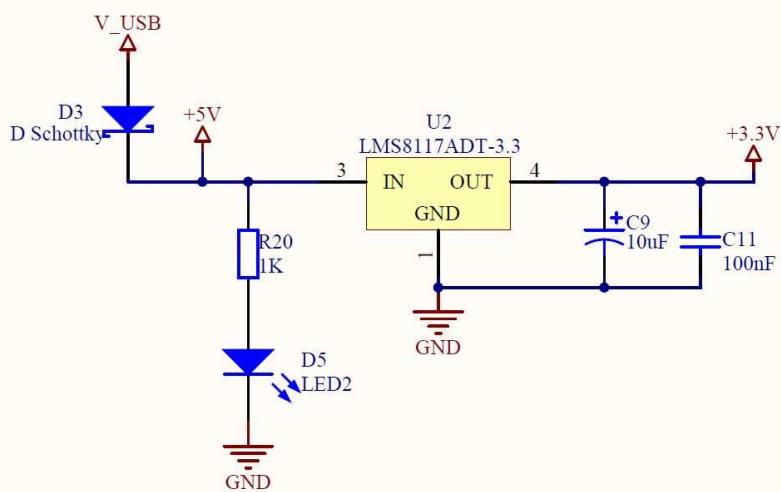


2.1.3 DIL - Stückliste

Anzahl	BMKZ	Benennung	Wert	Gehäuse	Bemerkung
1	Bestueckung_DIL_Adapter.PcbDoc				PCB-Pool
1	U1	Mikrocontroller	STM32F103RBT6	LQFP64	
2	C2, C5	Keramikkondensator	22pF	SMD 0805	
2	C10, C12	Keramikkondensator	10pF	SMD 0805	
2	C1, C3	Keramikkondensator	47pF	SMD 0805	
1	C8	Keramikkondensator	4,7nF	SMD 0805	
9	C4, C6, C7, C11, C13, C14, C16, C17, C18	Keramikkondensator	100nF	SMD 0805	
2	C9, C15	Tantalkondensator	10uF	SMD BF B	
3	R12, R16, R17	Widerstand	560Ω	SMD 0805	
2	R14, R19	Widerstand	1MΩ	SMD 0805	
3	R5, R7, R15	Widerstand	100kΩ	SMD 0805	
2	R4, R20	Widerstand	1kΩ	SMD 0805	
1	R3	Widerstand	47kΩ	SMD 0805	
1	R1	Widerstand	22kΩ	SMD 0805	
4	R2, R8, R10, R18	Widerstand	10kΩ	SMD 0805	
1	R6	Widerstand	1,5kΩ	SMD 0805	
2	R9, R13	Widerstand	22Ω	SMD 0805	
2	Q1, Q2	NPN Transistor	BC847	SMD SOT-23	
2	D2, D8	LED grün		SMD PLCC2	
1	D1	LED rot		SMD PLCC2	
1	D5	LED blau		SMD PLCC2	
1	U2	3,3V Fixspannungsregler	LMS8117ADT3.3	SMD	
3	D3, D6, D7	Schottky-Diode	DO214A	SMD	
1	U3	USB-ESD-Schutz	USBLC6-25C6	SOT23	
1	Y1	Quarz	8,0000MHz		
1	Y2	Quarz	32,768kHz		
1	USB	USB Buchse	Typ B, gewinkelt		
1	F1	USB-Sicherung	750mA		
1	R11	Potentiometer	10kΩ	10mm	liegend
1	L1	Ferritperle	1,3mm innen		
4	S1, S2, S3, S4	Printtaster		h=4,2mm	
3	B0, B1, B2	3er Stiftleiste + Jumper			
1	P3	JTAG Stiftleiste	20-pol	2-reihig, gewinkelt	
2	P1, P2	DIL-Reihe-Buchse + Pfostenleiste 4mm hoch	20-pol	1-reihig	

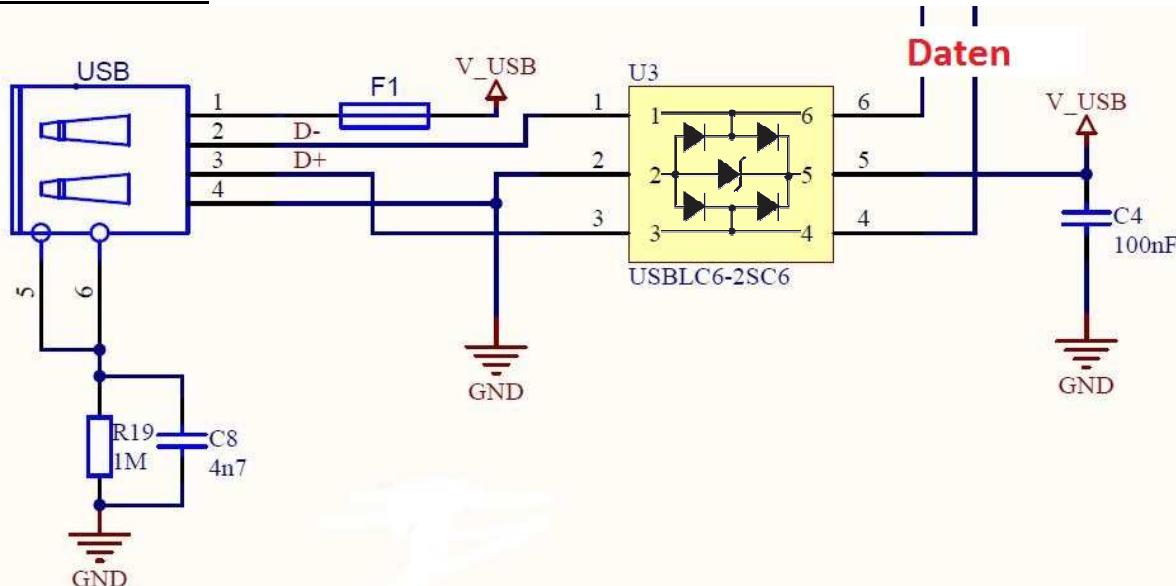
2.1.4 DIL - Schaltungsbeschreibung

Versorgung:

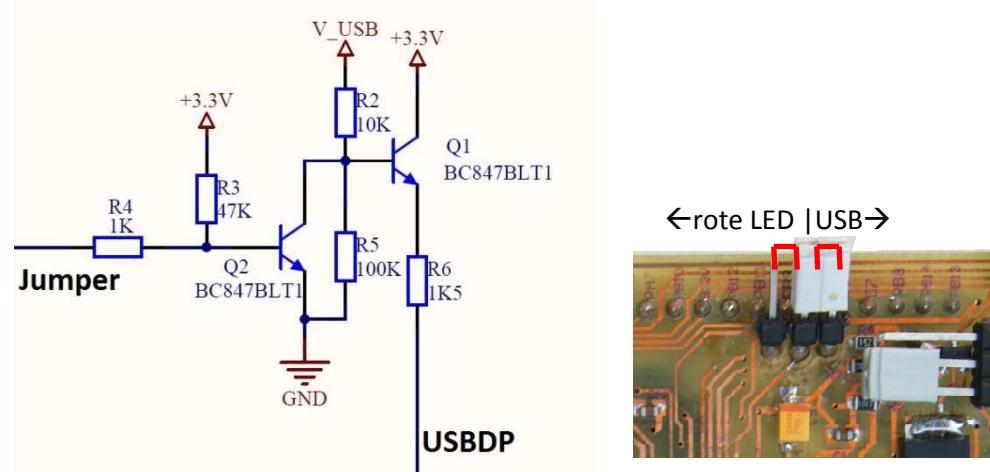


Die Versorgung der DIL-Platine kann entweder über die **+3,3V** der JTAG Schnittstelle oder über die **+5V** der USB Schnittstelle erfolgen. Das bedeutet sobald man eine Hardware anschließt, welche **+5V** benötigt (z.B. LCD), muss man die DIL über die USB Schnittstelle versorgen.

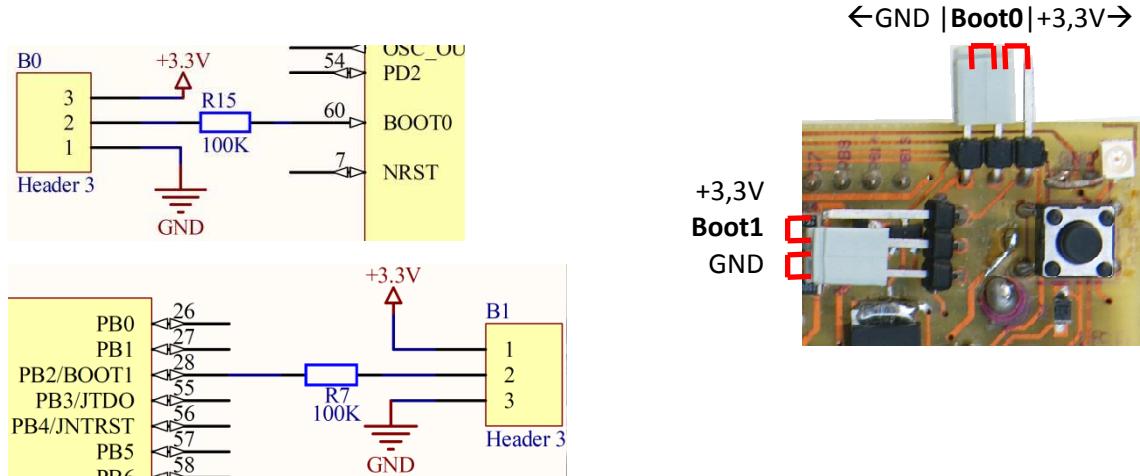
USB-Schnittstelle:



Die **USB** Schnittstelle kann als direkte Verbindung zu einem Computer oder als Stromversorgung verwendet werden. Um im Falle eines Kurzschlusses den **USB** Port des Computers zu schützen wurde eine Sicherung (**F1**) auf die **+5V** Leitung gesetzt. Die DIL-Platine besitzt außerdem einen **ESD** (Electrostatic discharge) Schutz (**USBLC6-2SC6**) um den Mikrocontroller vor statischen Entladungen über die **USB** Buchse zu schützen.

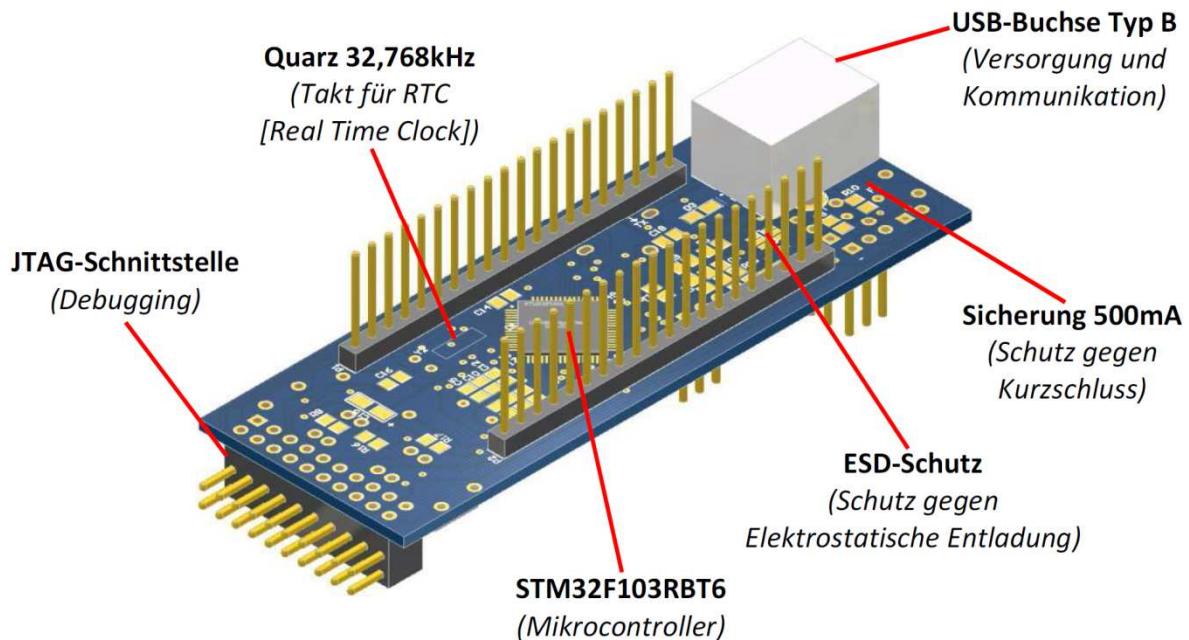
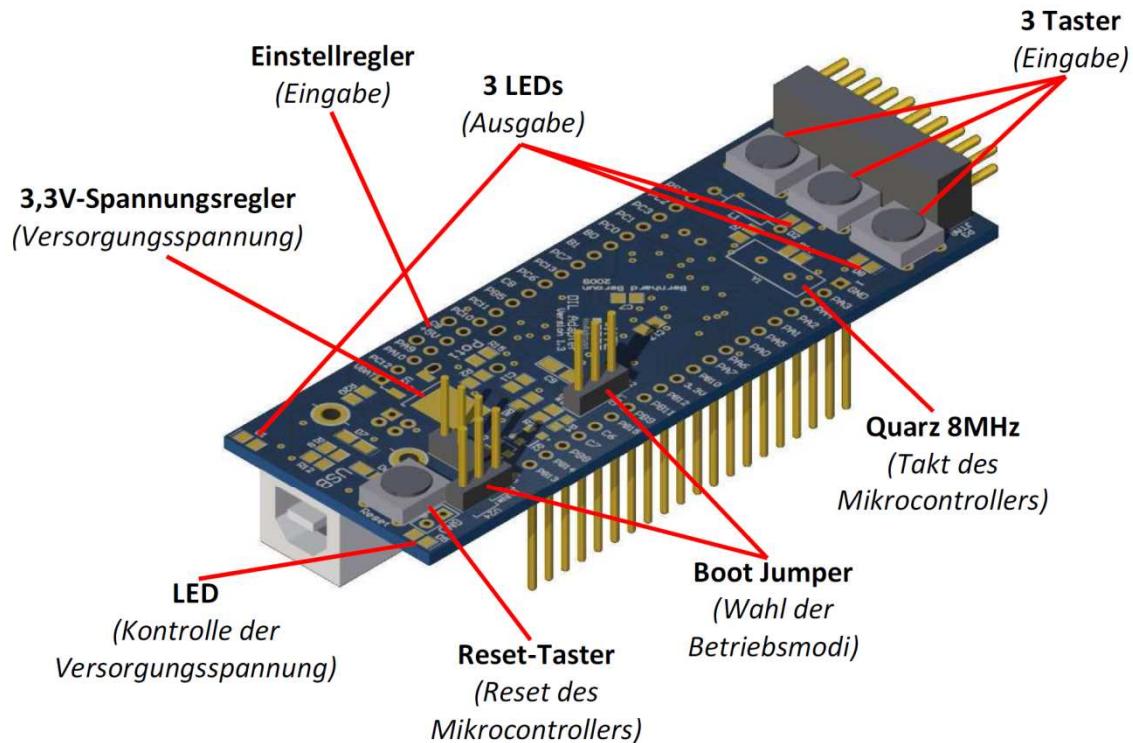
Jumper:**USB/rote LED:**

Mit dem Jumper „rote Led/USB“ kann ausgewählt werden ob entweder die rote Led in Verwendung ist und die USB Verbindung deaktiviert ist (USBDP [USB Daten +] über einen Pull-Up auf +3,3V geschalten) oder die rote Led ohne Funktion, dafür aber die USB Verbindung aktiviert ist (+3,3V Pull-Up deaktiviert).

Boot 0 & Boot 1:

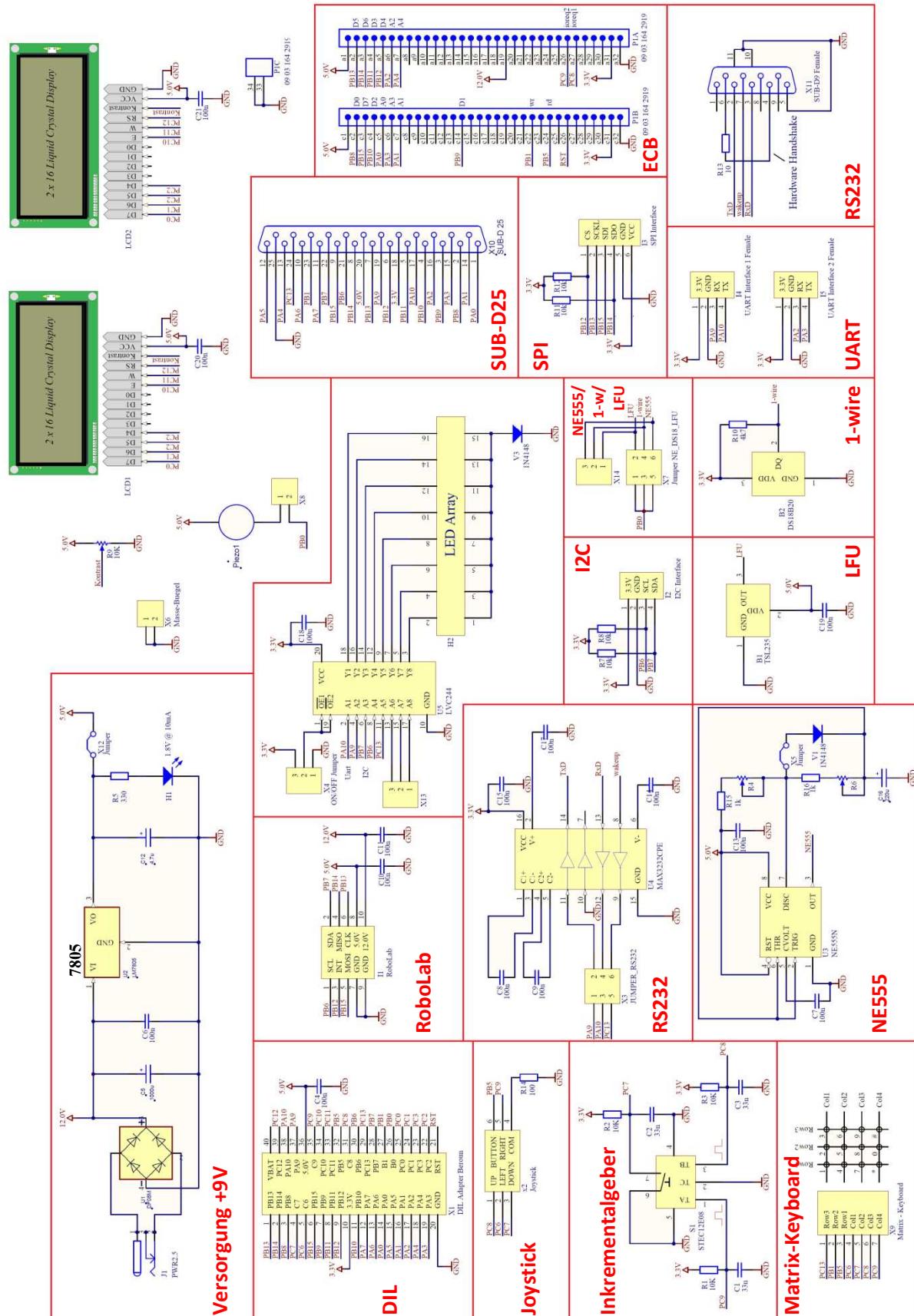
Die Jumper „Boot0“ und „Boot1“ entscheiden aus welchem Speicher der CortexM3 startet:

Boot1	Boot0	Funktion
X	GND	Von Flash-Speicher starten (run Application)
GND	+3,3V	Von Systemspeicher starten (UART)
+3,3V	+3,3V	Von SRAM starten

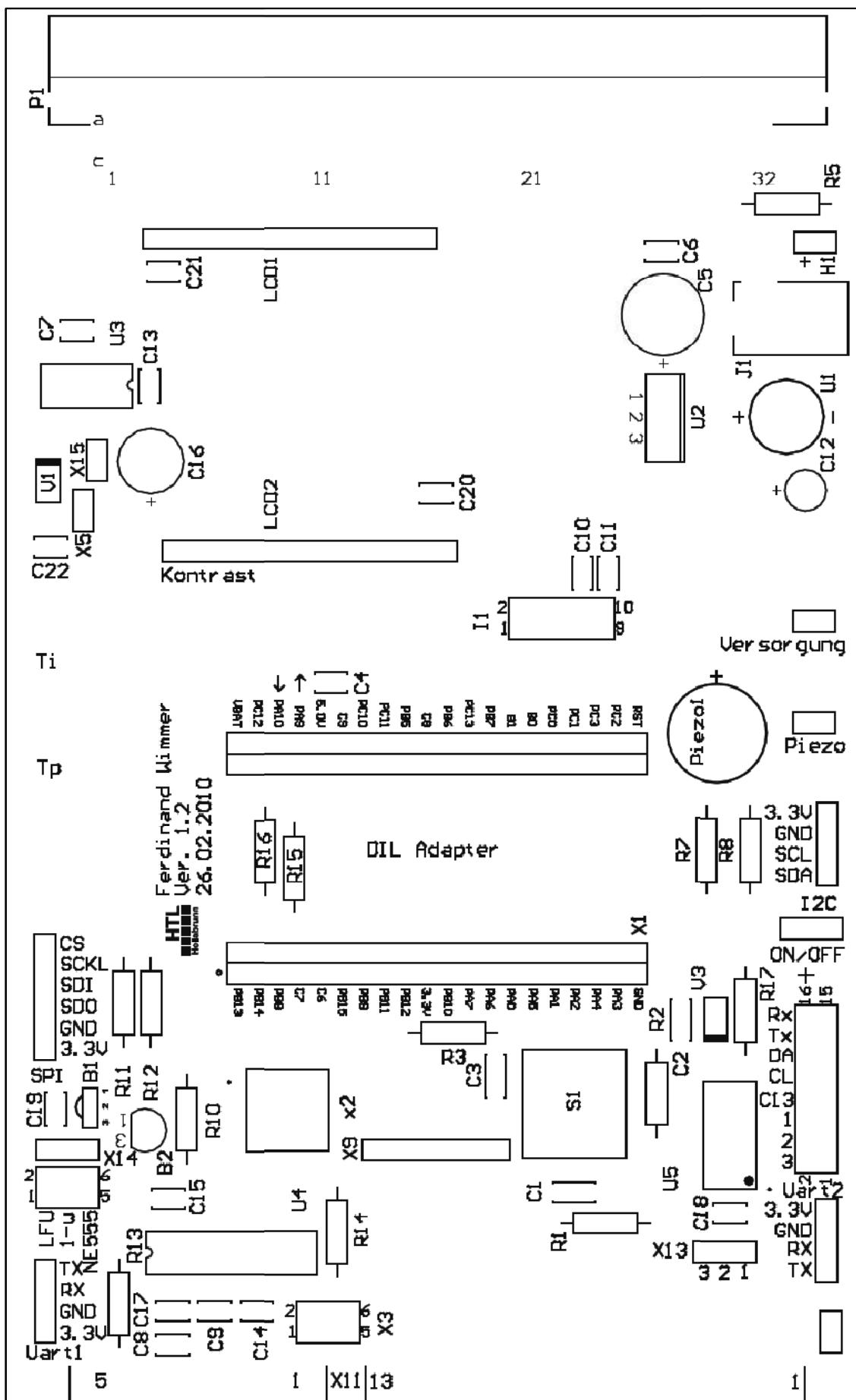
Ansichten des DIL-Adapters:

2.2 Europlatine

2.2.1 Euro - Stromlaufplan



2.2.2 Euro - Bestückungsplan

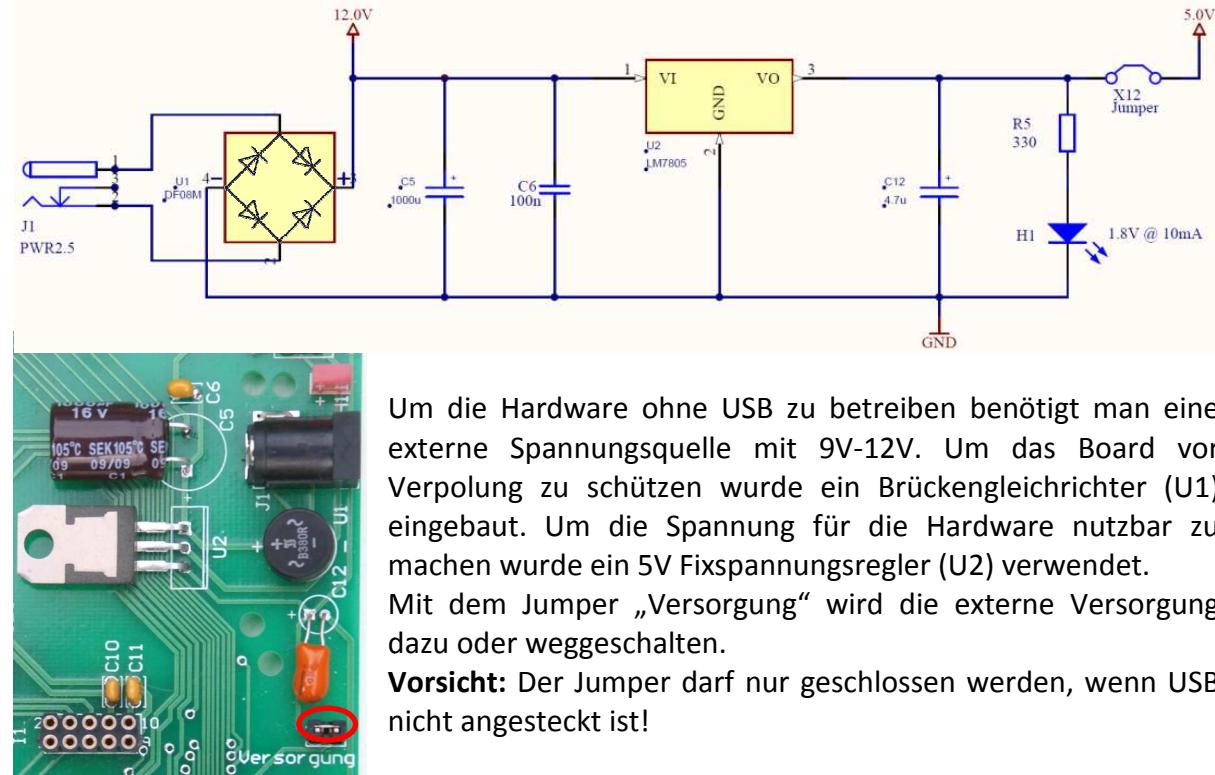


2.2.3 Euro - Stückliste

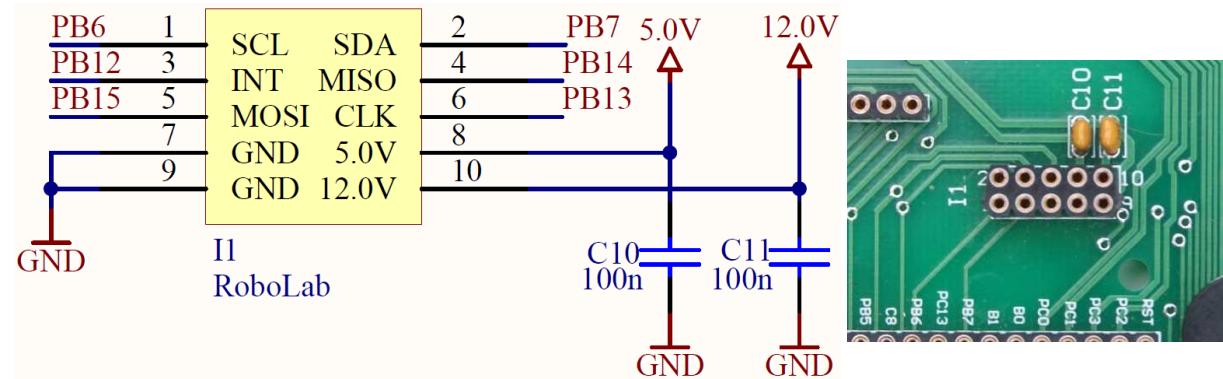
Anzahl	BMKZ	Benennung	Wert	Gehäuse	Bemerkung
1	MainBoard_CortexM3_Ver.1.2.PcbDoc				PCBPool
1	U5	Tristate-Buffer	SN74ALVC244DWG4	SMD	
3	C1, C2, C3	Keramikkondensator	33nF		
16	C4, C6, C7, C8, C9, C10, C11, C13, C14, C15, C17, C18, C19, C20, C21, C22	Keramikkondensator	100nF		
		Keramikkondensator			
1	C5	Alu-Elektrolytkondensator	1000µF/16V		Liegend
1	C12	Tantal-Kondensator	4.7µF/6,3V		
1	C16	Alu-Elektrolytkondensator	220µF/6,3V		liegend
7	R1, R2, R3, R7, R8, R11, R12	Widerstand	10kΩ		
1	R5	Widerstand	330Ω		
1	R10	Widerstand	4,7kΩ		
1	R13	Widerstand	10Ω		
1	R14	Widerstand	100Ω		
2	R15,R16	Widerstand	1kΩ		
1	R17	Widerstand	270Ω		
3	R4,R6,R9	Potentiometer	10kΩ	10mm	liegend
2	X1a	IC Sockelstreifen	20-polig		
2	X1b	Buchsenleiste	20-polig, hoch		DIL-Adapter
1	X2a	Joystick	SKQUCAA010		
2	X2b	Buchsenleiste	2- & 1-polig		Sockel f. Joystick
1	X6	Massebügel	u-förmig gebogener Widerstandsdräht		
5	X5a, X12, X15a,X8	Stiftleiste 7mm	2-polig		
9	X3b, X4b, X5b, X12, X15b, Piezo	Jumper			
5	X3a, X4a, X7, X13	Stiftleiste 7mm	3-polig		
1	X9	Buchsenleiste	7-polig, hoch		Keyboard
1	X10	SUB-D25 Buchse			L/S-Platine
1	X11	SUB-D9 Buchse			RS-232
2	X13, X14	IC Sockelstreifen	3-Polig		
1	B1	Licht-Frequenz-Wandler	LFU TSL235R		
1	B2	1-Wire Digital-Thermometer	DS18B20		
5	H1,H2c	LED rot	5mm, 20mA		
2	H2a	IC Sockelstreifen	8-polig		
4	H2b	LED grün	5mm, 20mA		
2	I1	Buchsenleiste	5-polig, hoch		I2C
3	I2,I4,I5	Buchsenleiste	4-polig, hoch		UART
1	I3	Buchsenleiste	6-polig, hoch		SPI
1	J1	Netzgerätebuchse	PWR2.5		
2	LCD1,LCD2	Buchsenleiste hoch	14-polig		
1	P1	VG-Leiste Din41612	Stecker 2 * 32 polig		a,c bestückt
1	Piezo1	Piezo-Summer	8V, 85dB		
1	S1	Inkrementalgeber	STEC 12E08		
1	U1	Brückengleichrichter	30V/1A		
1	U2	5V Fixspannungsregler	7805		liegend, Massefläche oben
1	U3a	NE555N	Multivibrator		
1	U3b	IC-Sockel Präzision	DIL08		NE555
1	U4a	3Volt RS232 Pegelwandler	MAX3232CPE		
1	U4b	IC-Sockel Präzision	DIL16		MAX232
2	V1, V3	Diode	1N4148		

2.2.4 Euro - Schaltungsbeschreibung

Versorgung:

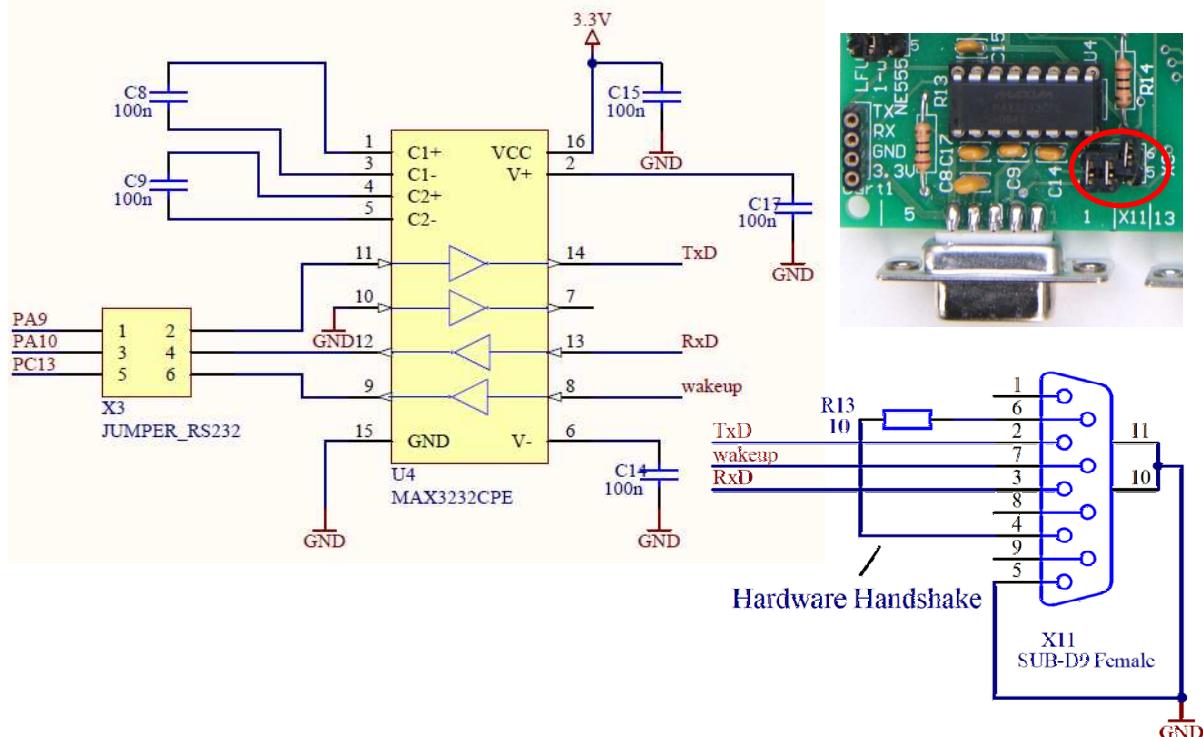


RoboLab:

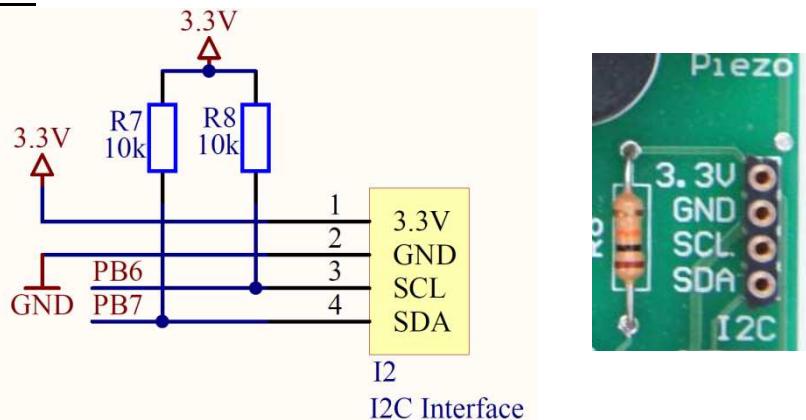


Damit bereits vorhandene RoboLab¹ Hardware mit diesem Board betrieben werden kann, ist dafür ein Steckplatz integriert. Mit diesem kann auch das LCD über den I2C-Bus angesteuert werden.

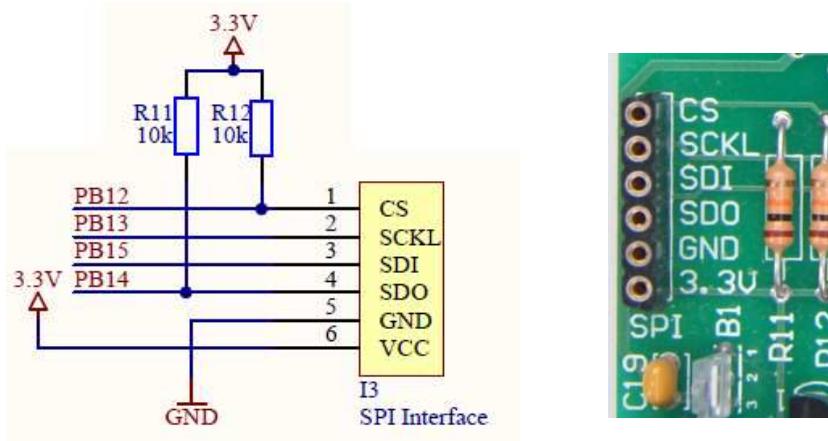
¹ RoboLab ist ein von Schülern an der HTBL-Hollabrunn entwickeltes Bussystem, welches zur raschen Realisierung der Elektronik eines Robotersystems eingesetzt werden kann.

RS232:

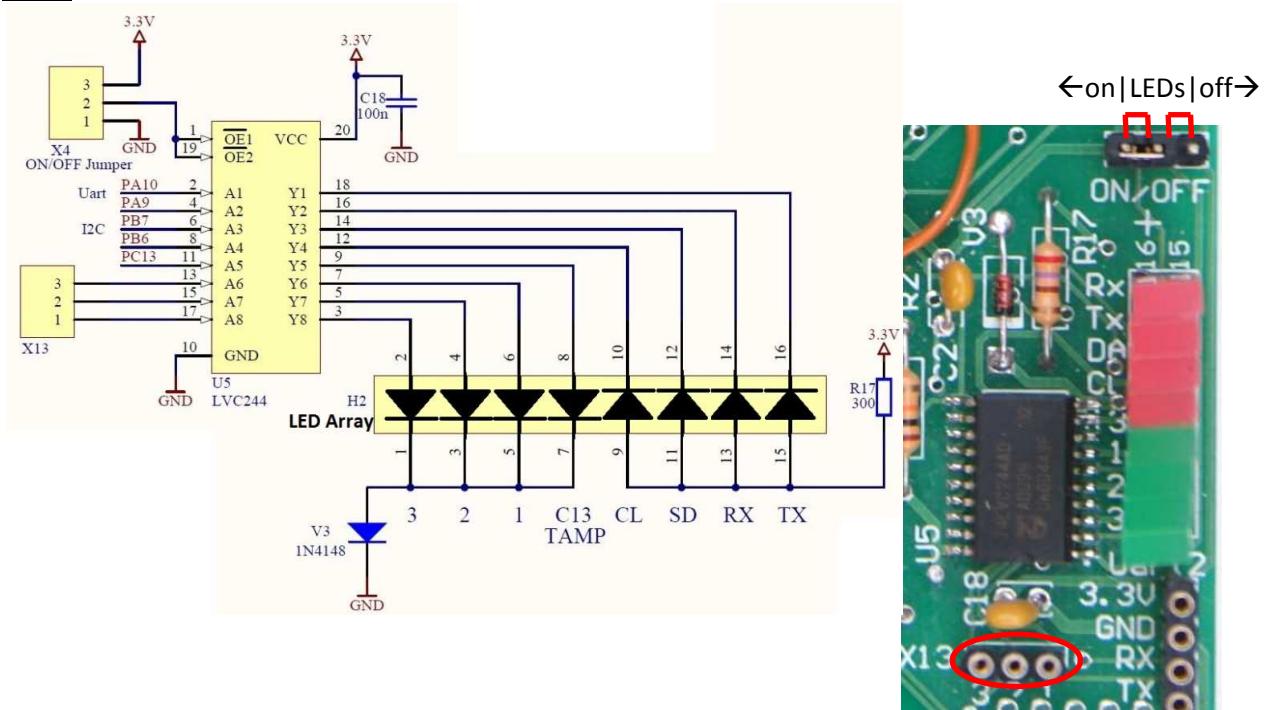
Für die Kommunikation über RS232 gibt es den Treiberbaustein MAX2323 (3,3V-Version des MAX232), der die Signale des Mikrocontrollers (3,3V) auf das Level von RS232 bring (+-12V). Die drei Jumper für RxD, TxD und wakeup verbinden die jeweiligen Leitungen mit dem MAX2323.

I2C:

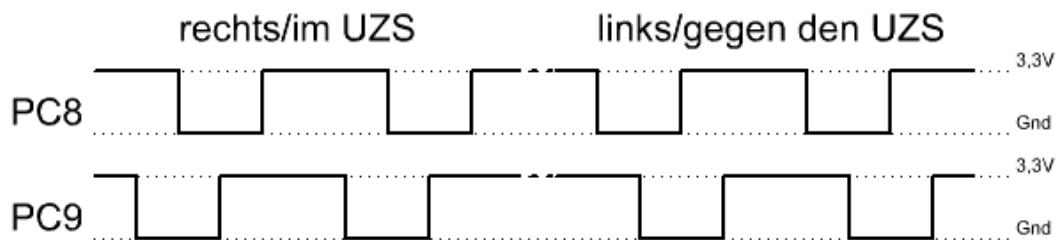
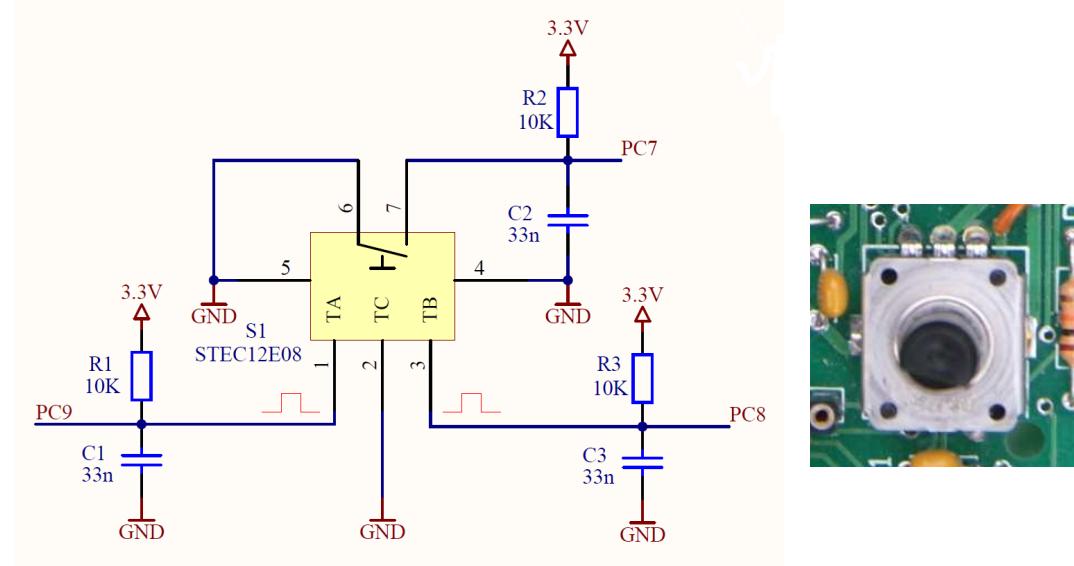
Der I2C Bus ist ein auf dem Master-Slave Prinzip basierender serieller Bus und wird zum Ansteuern von verschiedenster Hardware verwendet.

SPI:

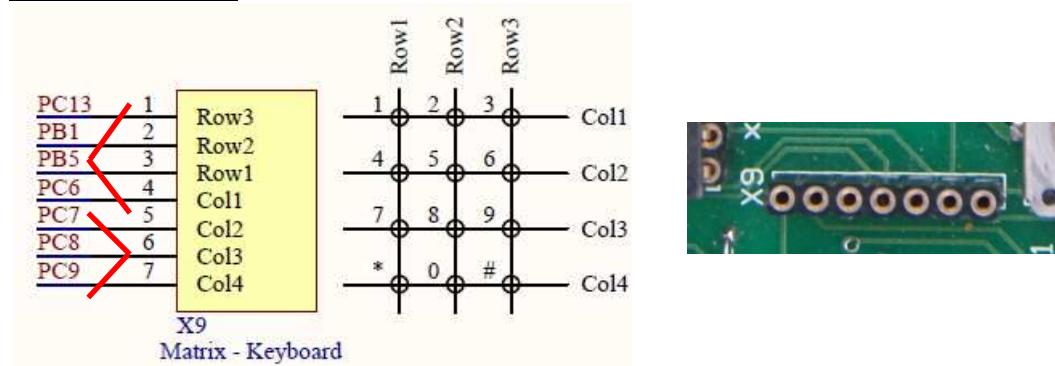
Das Serial Peripheral Interface (kurz: SPI) ist ein von Motorola entwickeltes Bus-System mit einem sehr lockeren Standard für einen synchronen seriellen Datenbus, mit dem digitale Schaltungen nach dem Master-Slave-Prinzip miteinander verbunden werden können.

LEDs:

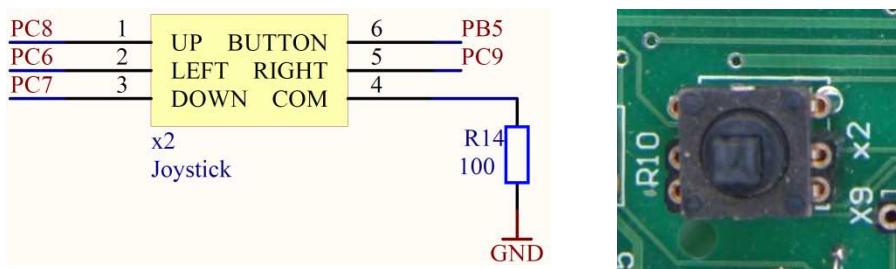
Damit man auf einen Blick erkennen kann ob Daten über die RS232 oder den I2C Bus gesendet werden, gibt es 4 rote und 4 grüne LEDs. 3 dieser LEDs sind frei verfügbar und können über die Buchsenleiste mit jedem beliebigem Pin verbunden werden. Bei offenen Eingängen der 3 Pins ist nicht sichergestellt, ob die dazugehörige LED leuchtet oder nicht und kann flackern (nicht definierter Eingangszustand des LVC244). Da die Datenleitungen nicht belastet werden dürfen, um eine gute Übertragung sicherzustellen, wird ein Buffer (U5) verwendet. Mit dem On/Off Jumper kann der Buffer aktiviert und deaktiviert werden.

Inkrementalgeber:

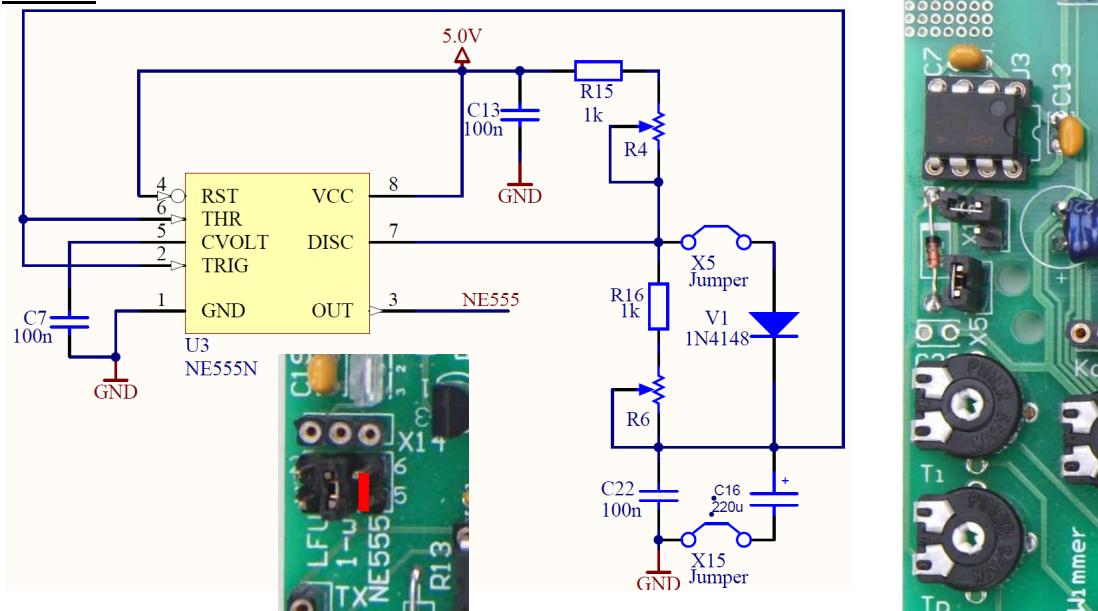
Der Inkrementalgeber (STEC12E08) gibt während er gedreht wird ein Rechtecksignal auf seinen zwei Pins aus. Weiters hat er auch einen Taster.

Matrix-Keyboard:

Um ein Matrix-Keyboard mit 12 Zeichen („0“-„9“; „*“ und „#“) ansteuern zu können benötigt man 7 Portleitungen von denen 3 die Reihen und 4 die Spalten des Keyboards darstellen.

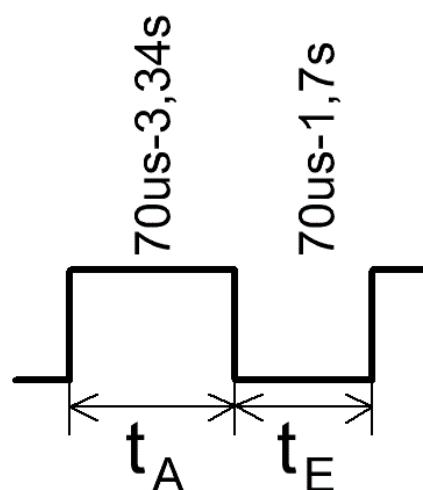
Joystick:

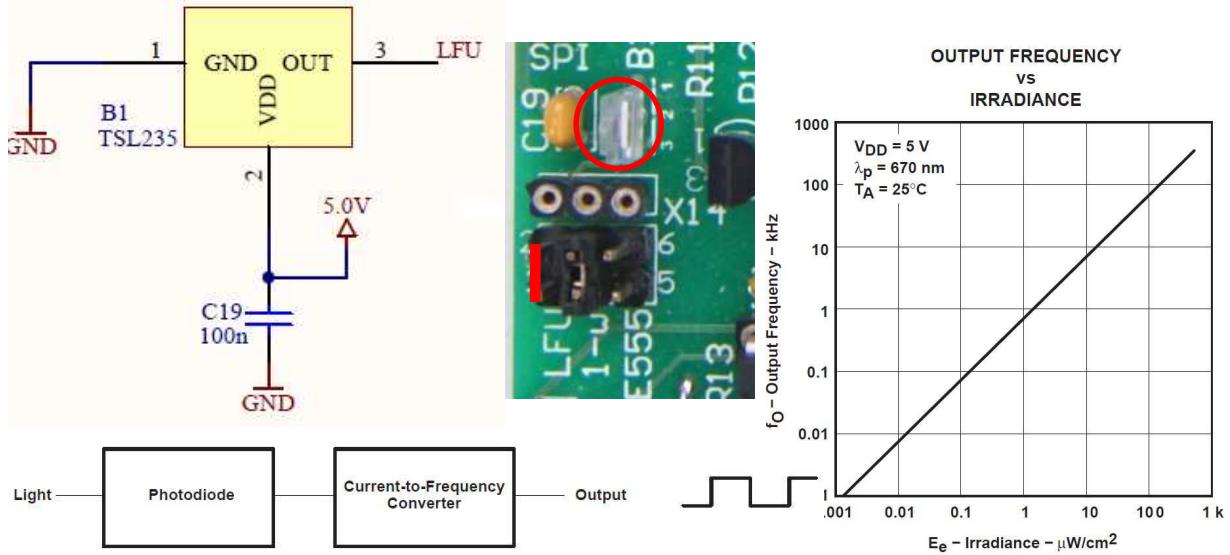
Der eingebaute Joystick verfügt über einen Taster und vier Richtungspositionen. Es können auch zwei Richtungen gleichzeitig gedrückt werden (diagonale Richtungen).

NE555:

Der NE555 kann als externer Taktgenerator verwendet werden, durch die beiden Potentiometer (R4, R6) können die Taktfrequenz und das Tastverhältnis eingestellt werden. Um den Taktgenerator mit dem Port B1 des CM3 zu verbinden muss der dazugehörige Jumper auf die rechten Pins gesteckt werden.

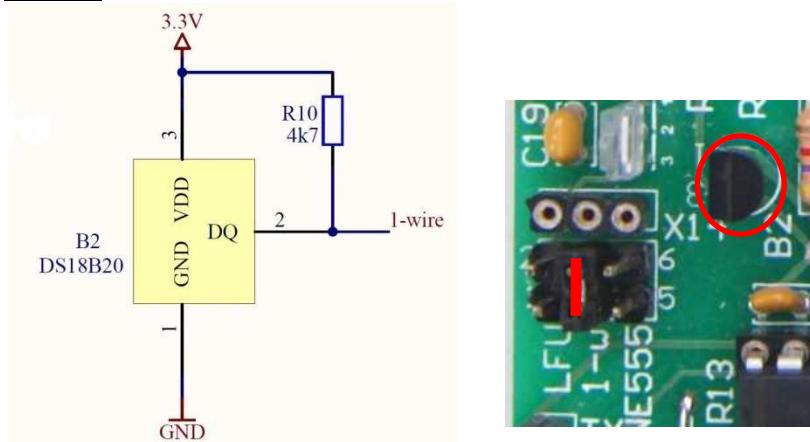
R4+R15	R6+R16	C	t_A	t_E
	1kΩ	100nF	$X15$ zu $X5$ offen	70us
1kΩ	11kΩ	100nF		70us
11kΩ	1kΩ	100nF		770us
11kΩ	11kΩ	100nF		770us
1kΩ	1kΩ	220uF	$X15$ zu $X5$ zu	154ms
1kΩ	11kΩ	220uF		154ms
11kΩ	1kΩ	220uF		1,694s
11kΩ	11kΩ	220uF		1,694s



LFU:

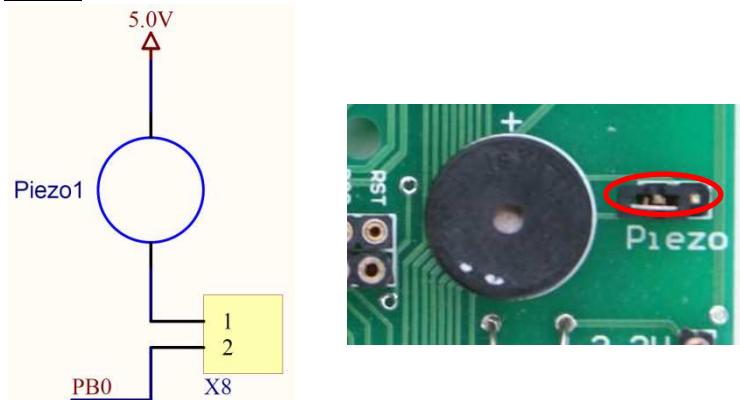
Auf dem Mainboard ist auch der Einsatz eines LFU (Light to Frequency Converter) vorgesehen. Je nach Lichtstärke gibt der TSL235 eine bestimmte Frequenz am Output aus, damit kann man ohne großen Aufwand die Lichtstärke messen.

Um die LFU mit dem Port B1 des CM3 zu verbinden muss der dazugehörige Jumper auf die linken Pins gesteckt werden.

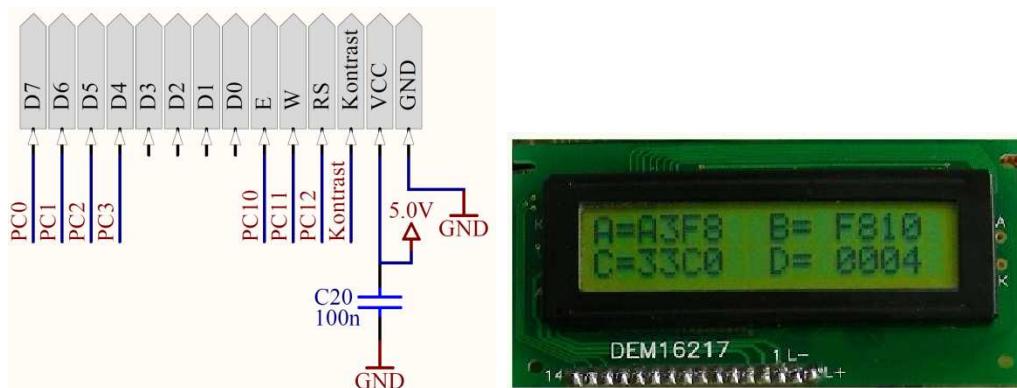
1-Wire:

Der 1-Wire Bus ist eine serielle Schnittstelle welche nur 1 Leitung für Versorgung und Daten gemeinsam verwendet. Der 1-Wire hat aber auch eine GND Leitung, dadurch hat man insgesamt 2 Leitungen. Die Übertragung ist asynchron, das heißt ohne übertragenem Takt, seriell und halbduplex. Das Bussystem basiert auf dem One-Master/Multiple-Slave System, das bedeutet dass es nur einen Master geben darf, aber bis zu 100 Slaves. Der 1-Wire Sensor wird über einen integrierten Kondensator betrieben, wenn dieser die 64bit-Datenblöcke verschickt.

Um den 1wire-Temperatursensor mit dem Port B1 des CM3 zu verbinden muss der dazugehörige Jumper auf die mittleren Pins gesteckt werden.

Piezo:

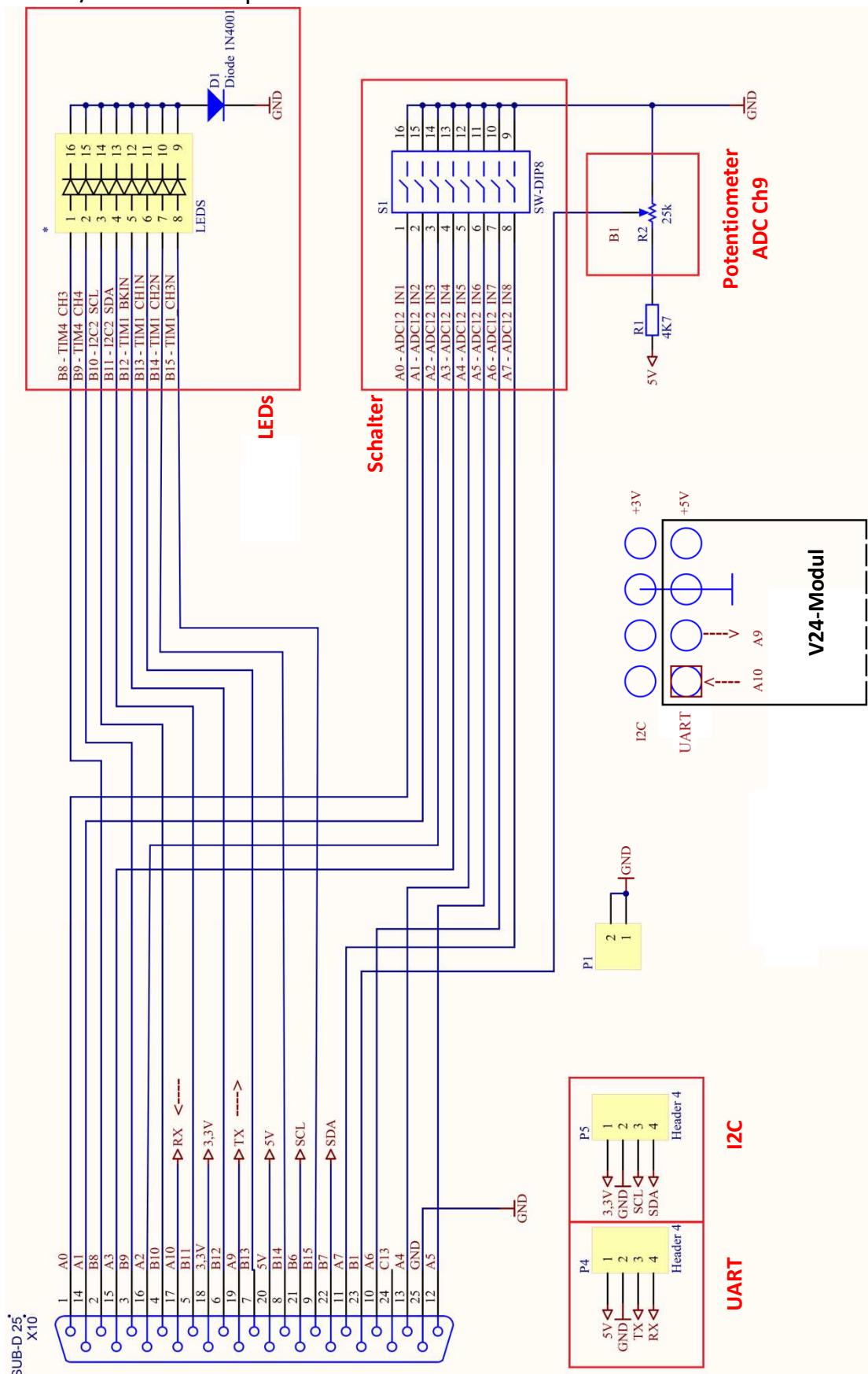
Der Piezo wird mit dem zugehörigen Jumper mit dem Pin B1 des CM3 verbunden.
Wenn der 1-wire oder die LFU eingelesen werden sollen, muss der Jumper geöffnet sein.

LCD:

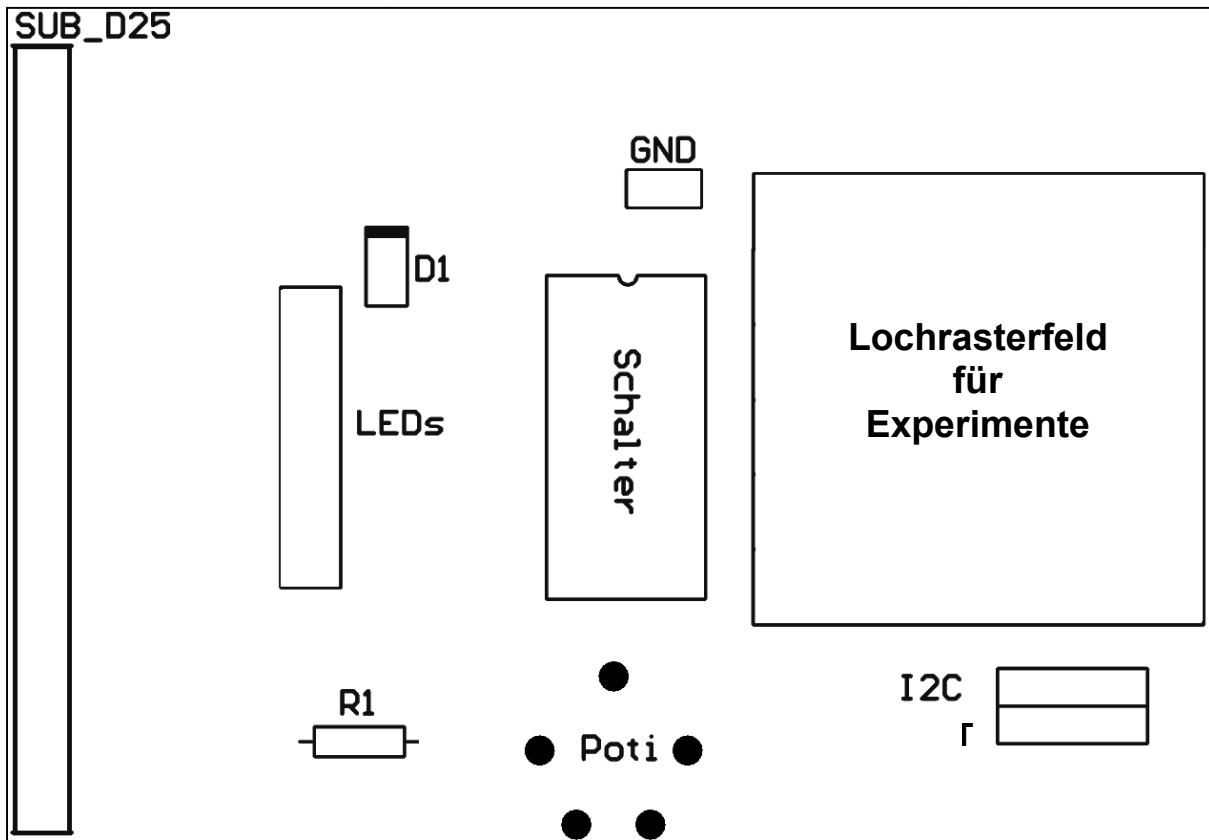
Das verwendete Display besitzt zwei Zeilen mit jeweils 16 Zeichen und wird im 4-bit Modus über 7 Portleitungen (PC0, PC1, PC2, PC3, PC10, PC11, PC12) der CortexM3 DIL-Platine angesteuert. Die weiteren Pins sind Kontrast, VCC (+5V) und GND. Die Pins (L-, L+) sind nicht verwendet, da unser Display keine Beleuchtung besitzt.

2.3 LED/Schalter-Platine

2.3.1 L/S - Stromlaufplan



2.3.2 L/S - Bestückungsplan

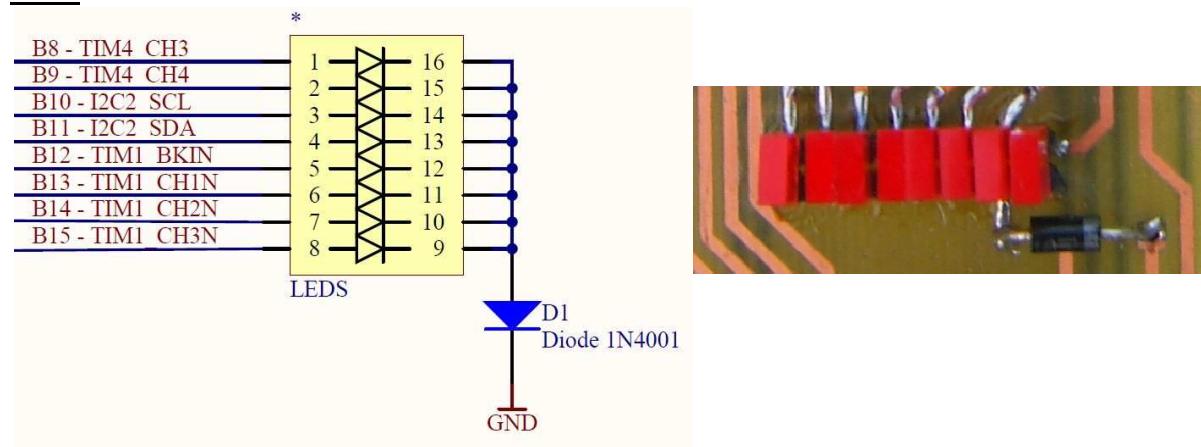


2.3.3 L/S - Stückliste

Anzahl	BMKZ	Benennung	Wert	Gehäuse	Bemerkung
1	Platine – Maurer Nr. 625				
1	SUB_D25	SUB-D25 Stecker			
1	Schalter	IC Sockel Präzision	DIL16		DIP-Schalter
1	Schalter	DIP-Schalter	8-polig, stehend		
1	LEDs	IC-Sockelstreifen	8-polig, h=2,54mm		LEDs
4	LEDs1	LED rot	Rechteckig, 5mm, 20mA		
4	LEDs2	LED grün	Rechteckig, 5mm, 20mA		
1	D1	Diode	1N4148		
2	I2C, Hard	Buchsenleiste	4-polig, h=7mm		
1	R1	Metallschichtwiderstand	4,7kΩ		
1	Poti	Potentiometer	25kΩ	10mm	liegend

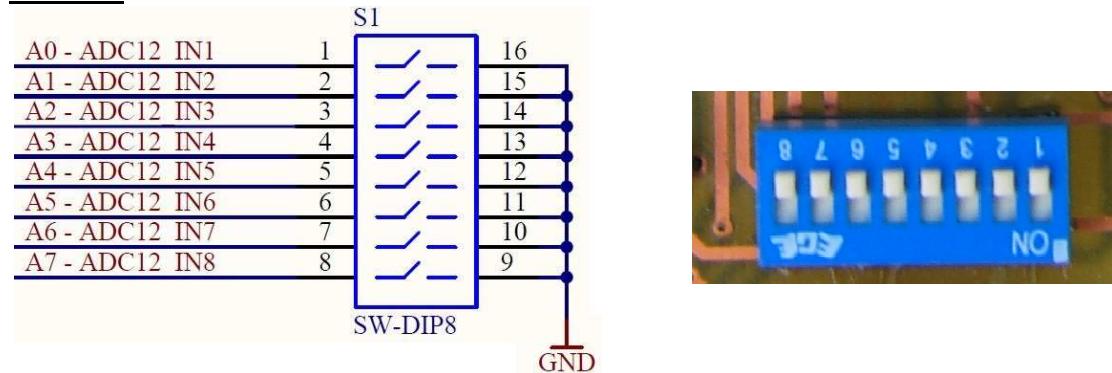
2.3.4 L/S - Schaltungsbeschreibung

LEDs:



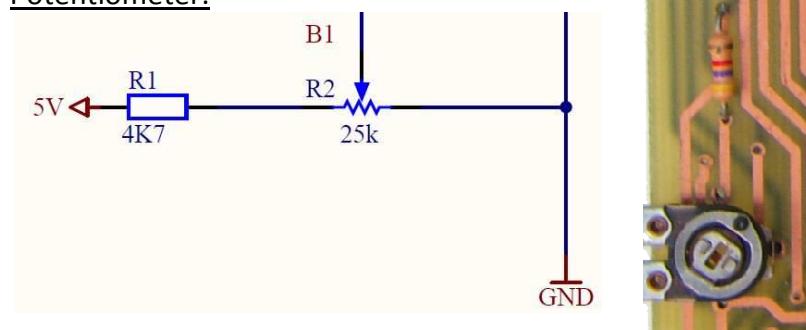
Die LEDs hängen an den Pins B8 bis B15 der CortexM3 DIL. Die Diode (D1) bildet mit den LEDs einen Spannungsteiler, damit benötigen wir hier keinen Widerstand zur Strombegrenzung.

Schalter:



Die Schalter der Platine hängen an den Pins A0 bis A7 der CortexM3 DIL. Diese Pins sind auch ADC-Eingänge, dadurch kann man damit auch verschiedene ADC Werte einstellen.

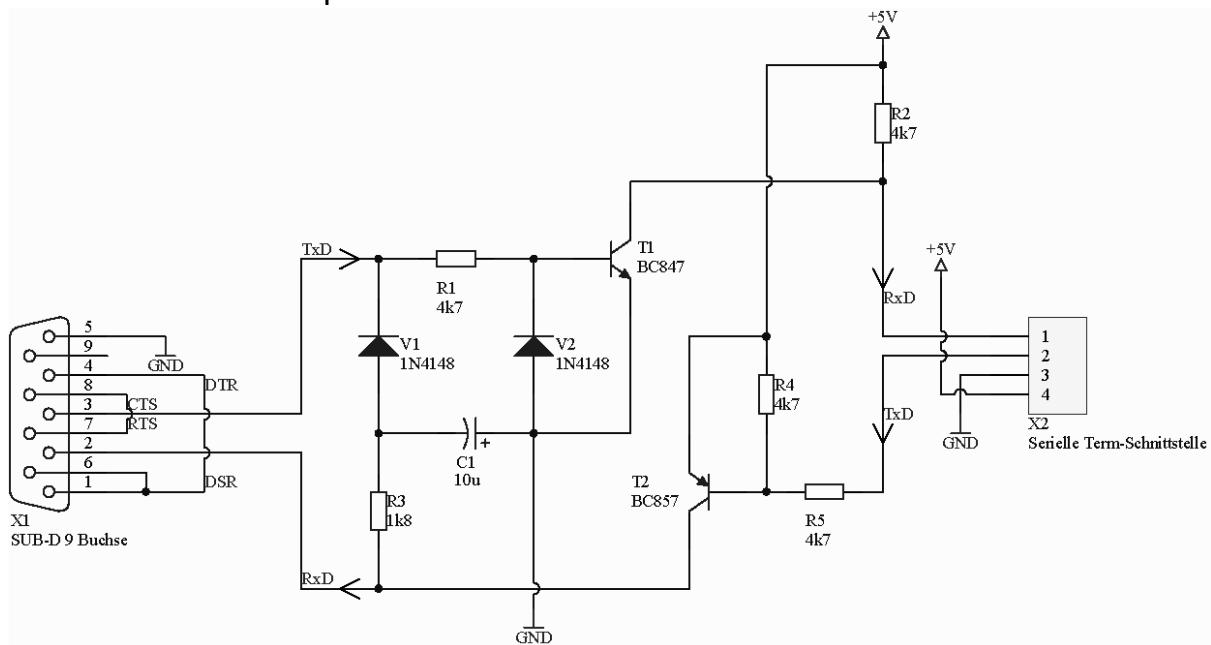
Potentiometer:



Das Potentiometer der Schalter/Led Platine hängt am Port B1 (Ch9) vom CortexM3 am ADC1.

2.4 V24-Modul

2.4.1 V24 - Stromlaufplan



2.4.2 V24 - Bestückungsplan



2.4.3 V24 - Stückliste

Anzahl	BMKZ	Benennung	Wert	Gehäuse	Bemerkung
1	Platine	Lochraster	2*5cm		
1	X1	SUB-D9 Buchse	9-polig		
1	X2	Stiftleiste	4-polig, 1-reihig		
2	V1, V2	Diode	1N4148		
4	R1, R2, R4, R5	Kohleschichtwiderstand	4,7kΩ		
1	R3	Kohleschichtwiderstand	1,8kΩ		
1	T1	NPN Transistor	BC847		
1	T2	PNP Transistor	BC857		
1	C1	Elektrolytkondensator	10uF/63V		

2.4.4 V24 - Schaltungsbeschreibung

Die V24 Schnittstelle liefert auf der TxD Leitung im Ruhezustand ca. -10 Volt. Dies entspricht einer logischen „1“. Wird nun ein Startbit gesendet, geht die TxD Leitung auf logisch „0“ also auf +10 Volt. Möchte man nun diese +10V Spannung auf TTL Pegel umformen, so ist neben der Spannungsanpassung das Signal noch zu invertieren. Dies kann ganz einfach mit einem Transistor erfolgen. Eine positive Spannung (größer 0,7V) schaltet den npn-Transistor T1 durch und am Kollektor liegt logisch „0“. Liegt jedoch logisch „0“ oder eine negative Spannung an der Basis so sperrt der npn und am Kollektor liegen +5V. Die Diode V2 begrenzt auf -0.7V und schützt die Basis-Emitter-Diode vor zu hohen negativen Spannungen. Schwieriger ist die Umformung von +5V auf -10V. Herkömmliche V24-Transceiver wie z.B. der MAX232 generieren über eine Spannungsverdopplerschaltung (Ladungspumpe mit Tantalkondensatoren) Eine Hilfsspannung von ca. +-8V. Dieser Aufwand ist nicht gerade billig!!

Geht man nun davon aus, dass die TxD Leitung des PCs oder Terminals ja sowieso im Ruhezustand -12V liefert, so kann man über die Diode V2 einen Kondensator C1 auf -12V laden. Dieser Kondensator dient dann als -12V Hilfs-Versorgung. Für die erforderliche positive Spannung reichen die +5V. Liefert nun der uC auf seiner TxD Leitung logisch „0“, so wird der pnp-Transistor leitend (Basisspannung ist um 0,7 Volt negativer als +5V und am Widerstand fallen 4,3 V ab) und liefert +5V auf die RxD Leitung des Terminals. Schickt der uC jedoch +5V so sperrt der pnp T2 und sein Kollektor liegt auf -12V. Der Transistor invertiert also wieder.

Prüfvorschrift:

Es ist ratsam zuerst einmal das Terminal bzw. die Kabelverbindung bis zum SUB-D 9 Stecker zu überprüfen. Dies geschieht sehr einfach durch eine Echo-Loop zwischen 2 und 3.

Weiters ist eine Überprüfung von +5V und GND durchzuführen.

Wird nun das V24 Modul eingesteckt, so darf noch kein uC eingesetzt sein. Wird am VT100 eine Taste gedrückt, so muss am RxD Pin ein 5V !! Rechteck mittels Oszi messbar sein.

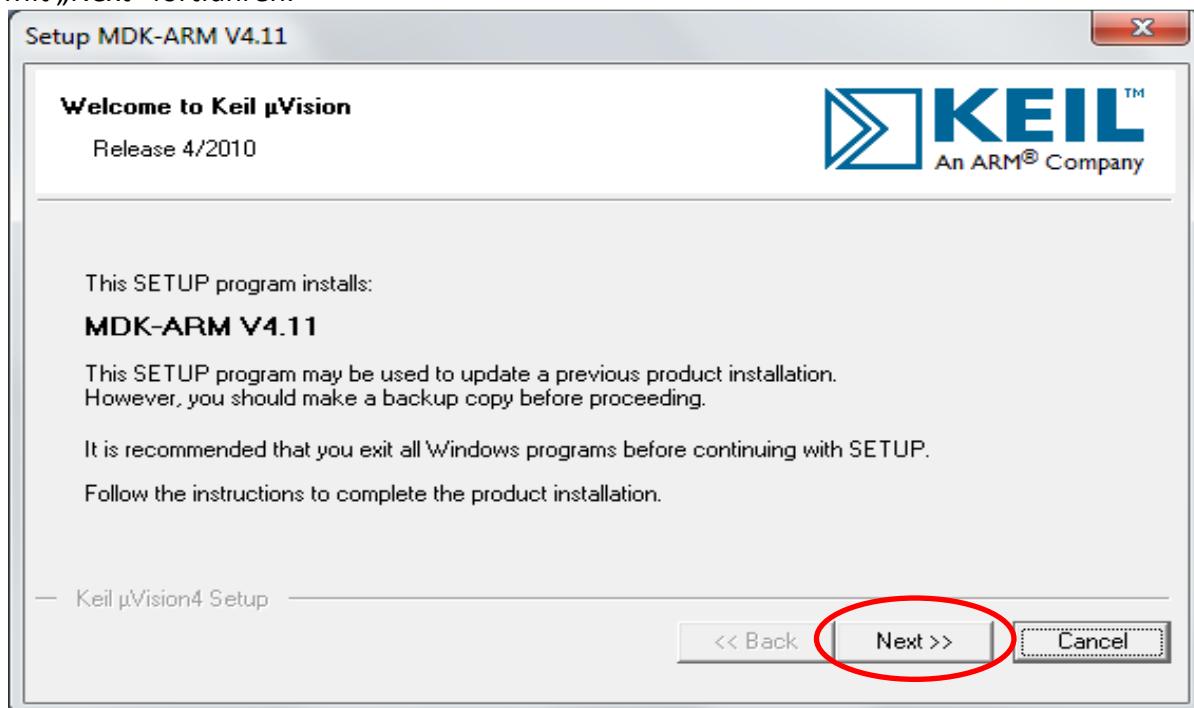
(+-10 Volt würden den uC zerstören!!) Am Terminalmonitor darf kein Echo erscheinen. Erst im nächsten Schritt wird anstelle des uC nun wieder eine Echo Verbindungsleitung zwischen RxD und TxD hergestellt. Funktioniert nun das Echo wird zuletzt der uC eingesetzt.

3. Inbetriebnahme

3.1 Installation uVision4

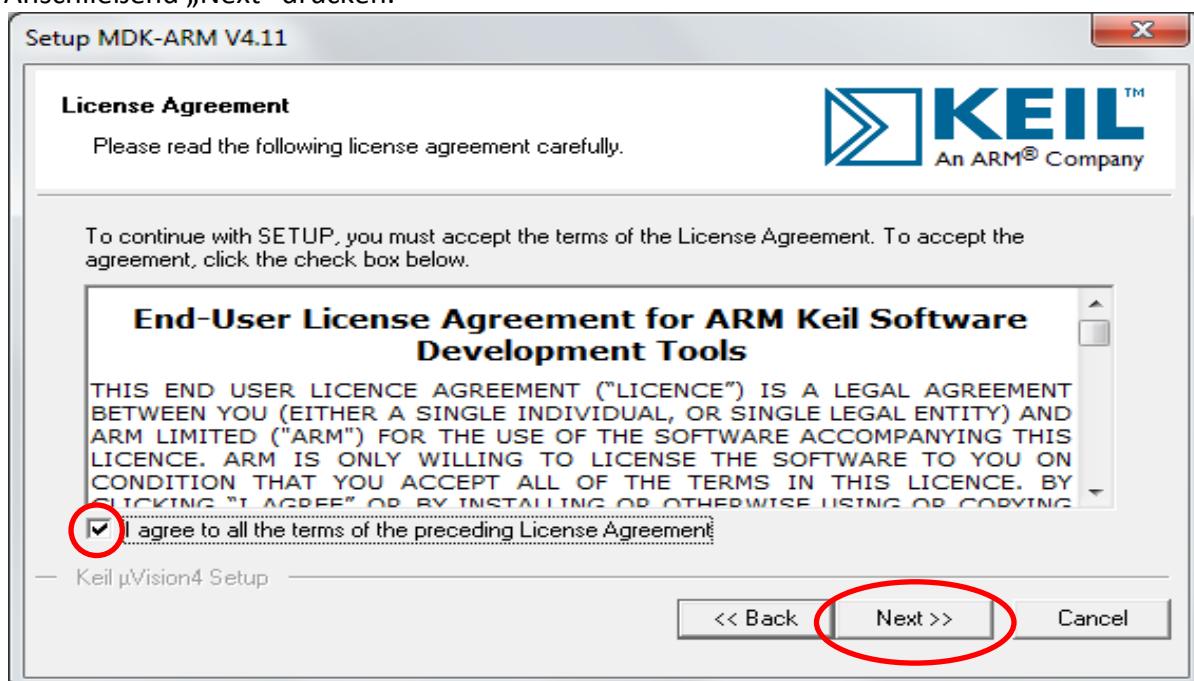
Datei „MDK411.exe“ öffnen (bei aktiver Benutzerkontensteuerung bei Windows Vista oder Windows 7 das Setup zulassen). Nun öffnet sich der Installations-Assistent.

Mit „Next“ fortfahren.

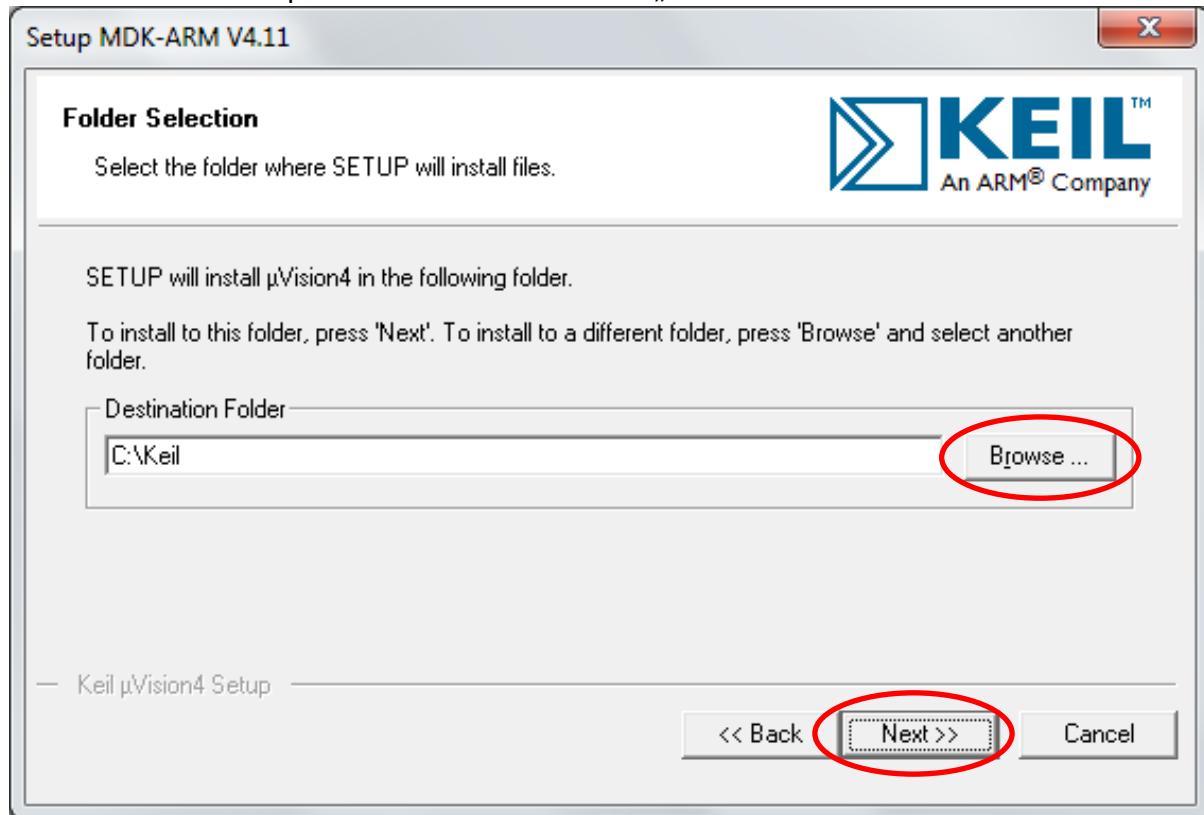


Die Lizenzvereinbarungen lesen ;-) und Häkchen setzen um zuzustimmen.

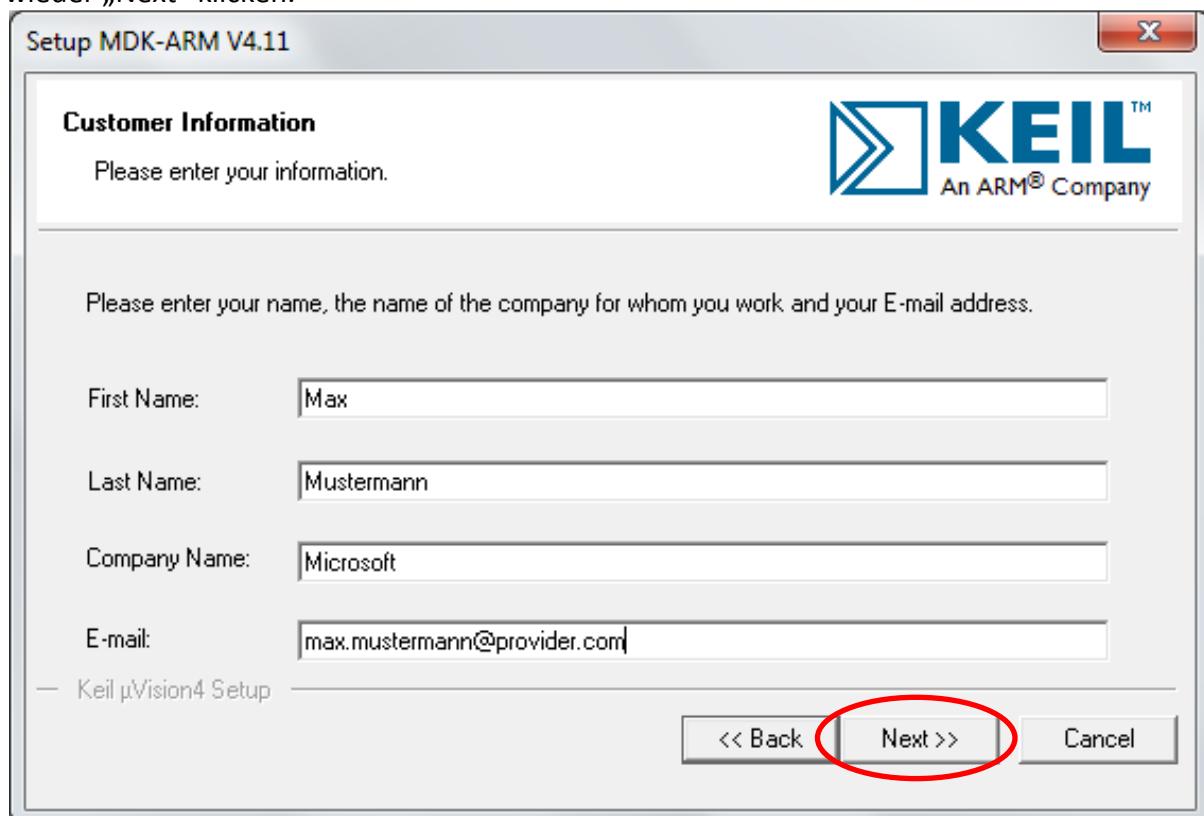
Anschließend „Next“ drücken.



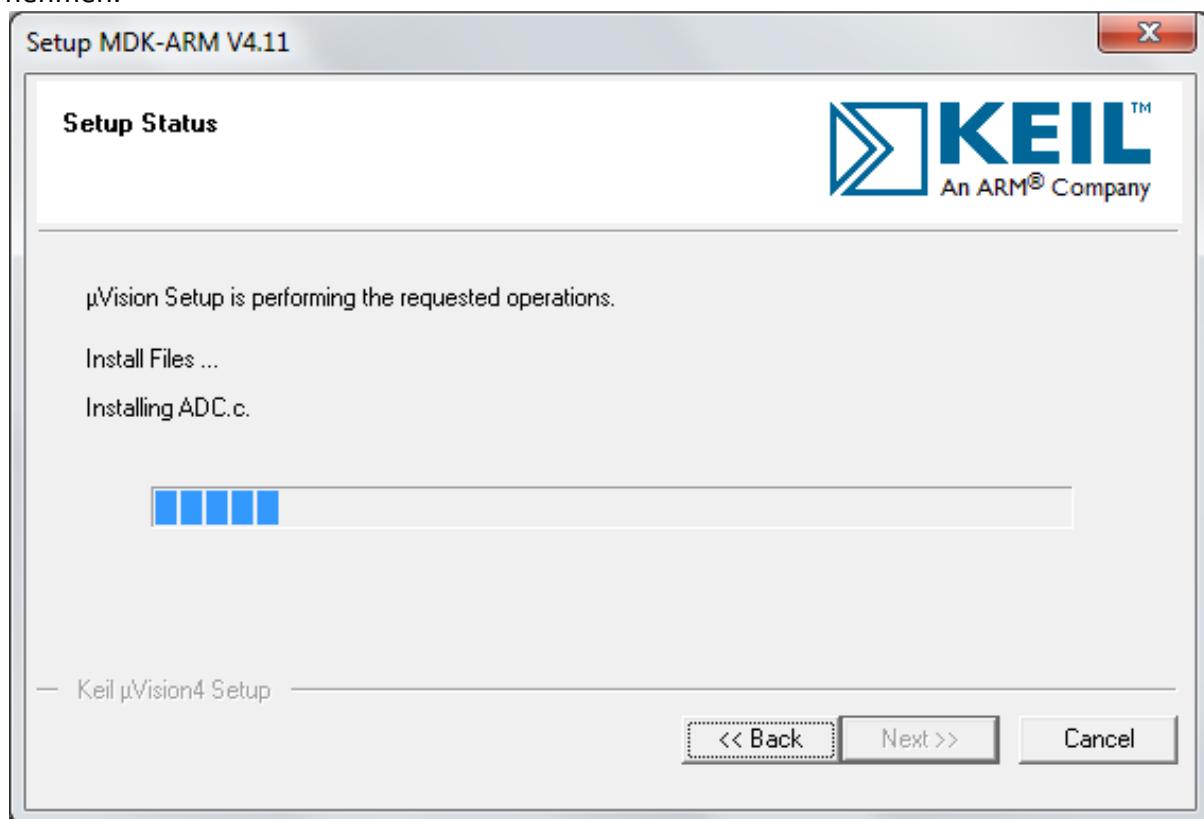
Jetzt den Installationspfad auswählen und wieder „Next“ klicken.



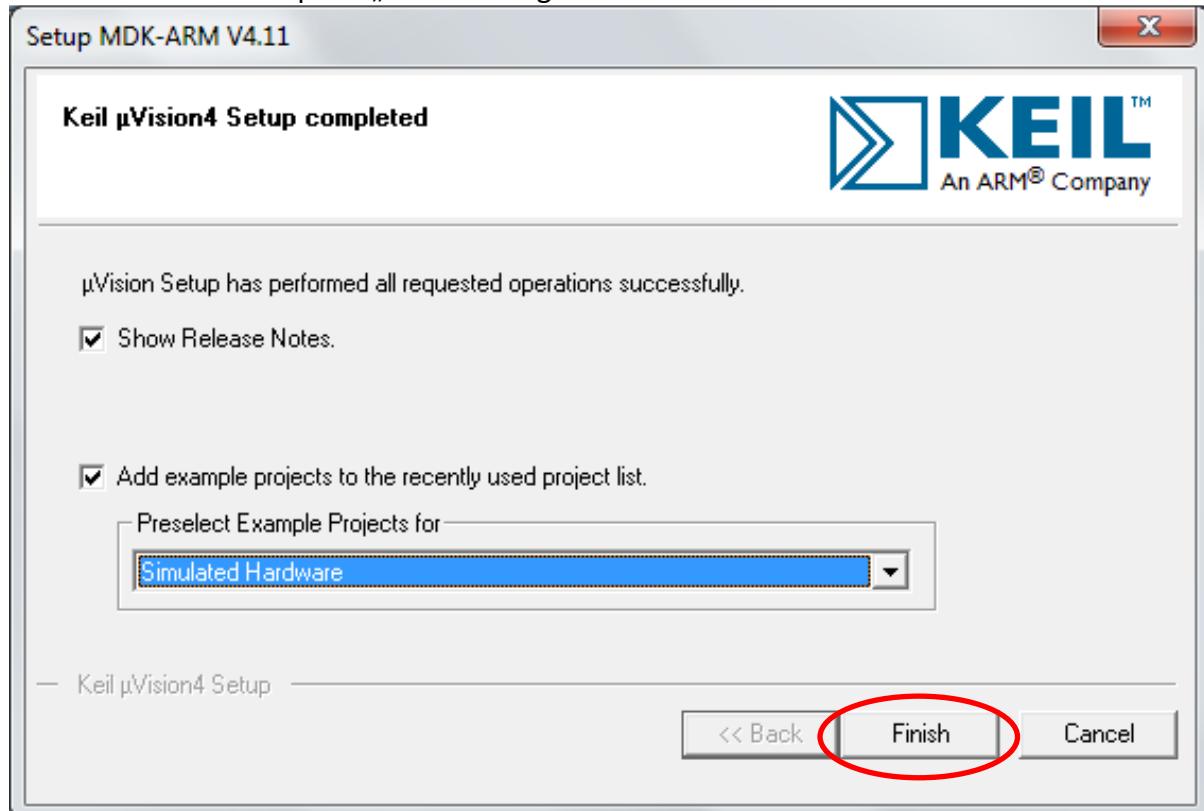
Auf dieser Seite die Felder mit Namen, Firma und E-Mail-Adresse ausfüllen und danach wieder „Next“ klicken.



Nun wird das Programm installiert. Dieser Vorgang wird einige Minuten in Anspruch nehmen.



Anschließend das Setup mit „Finish“ fertigstellen.

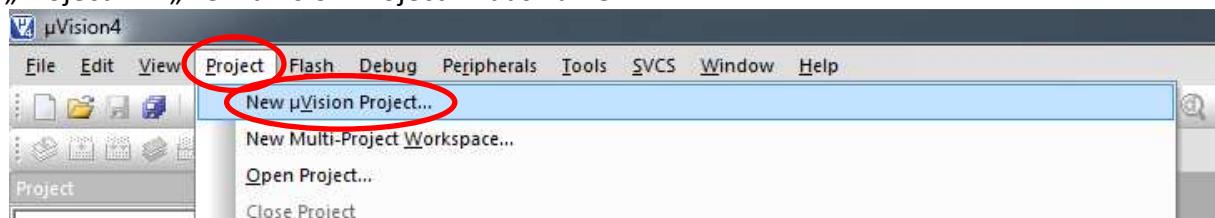


Das Programm kann jetzt auf dem Desktop mit folgendem Icon gestartet werden.

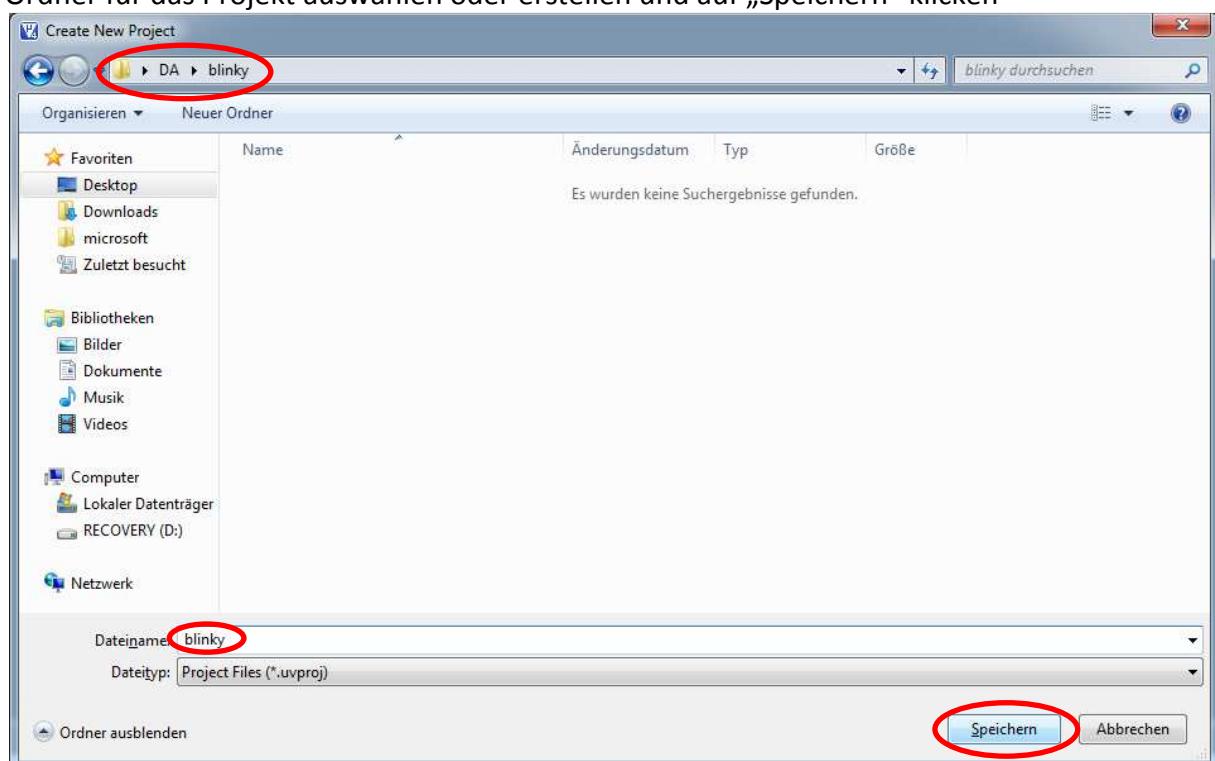


3.2 uVision4-Projekt erstellen

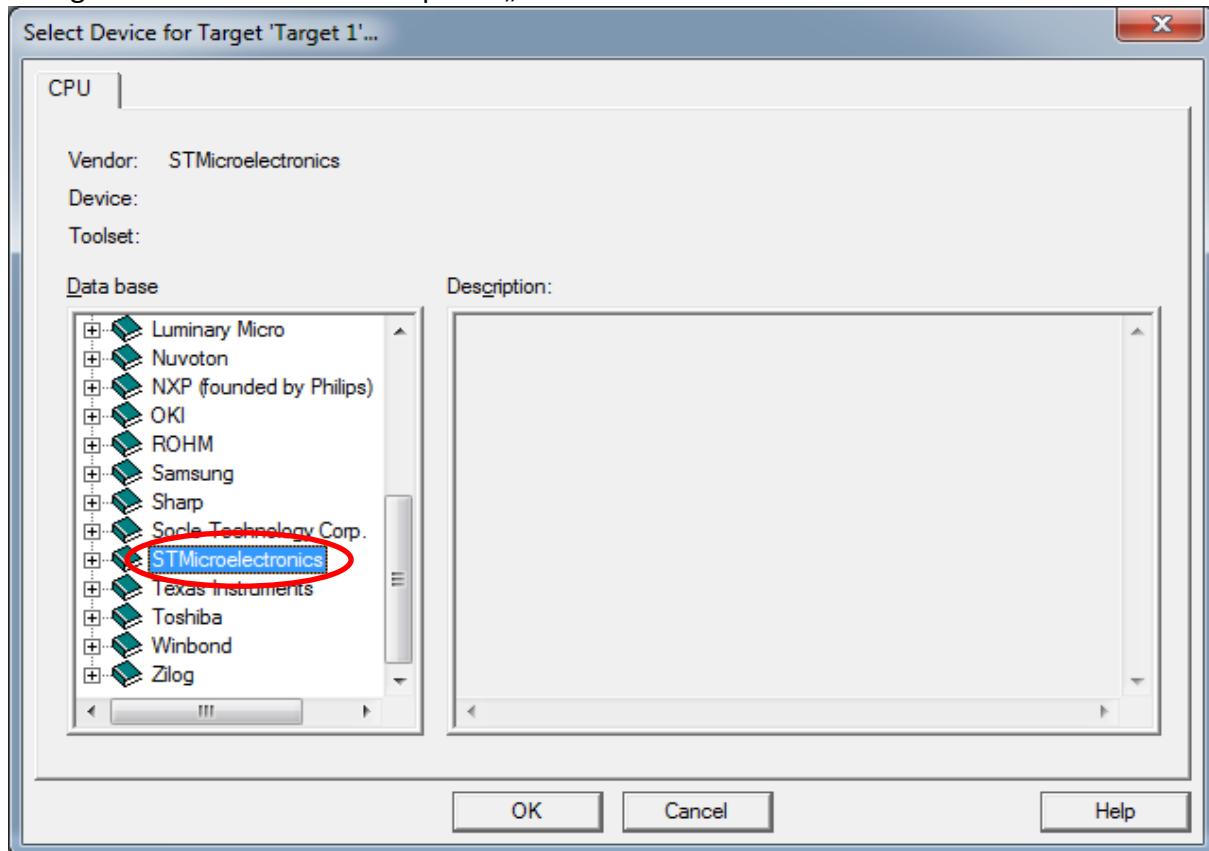
„Project“ → „New uVision Project...“ auswählen.



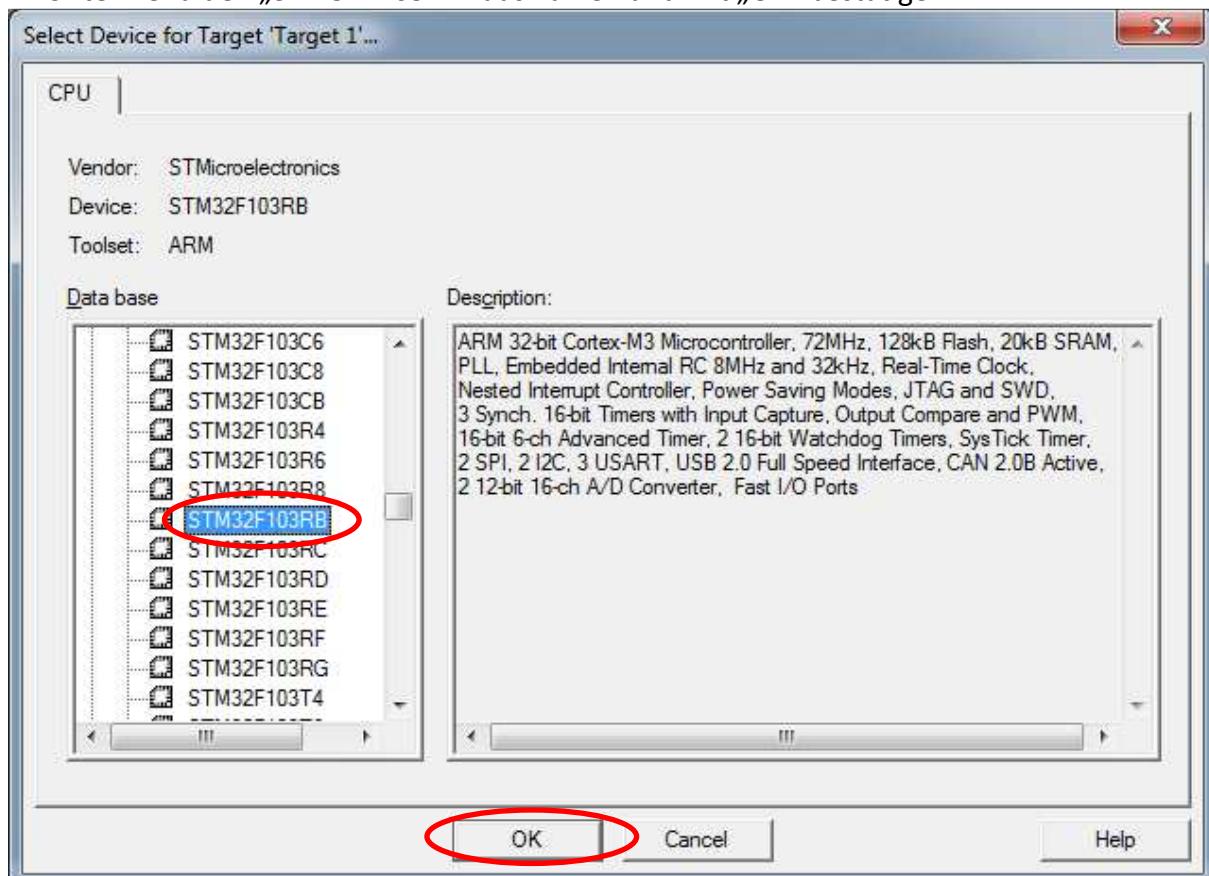
Ordner für das Projekt auswählen oder erstellen und auf „Speichern“ klicken



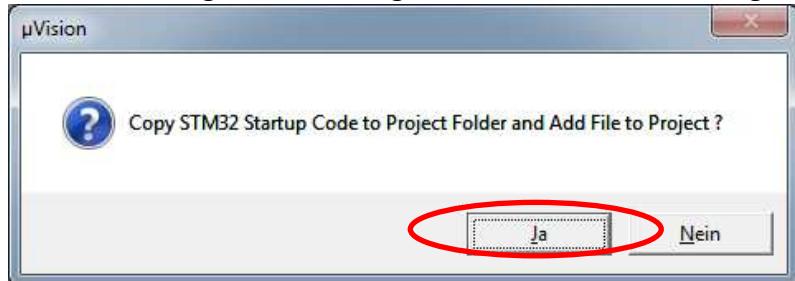
In folgendem Fenster den Menüpunkt „STMicroelectronics“ auswählen.



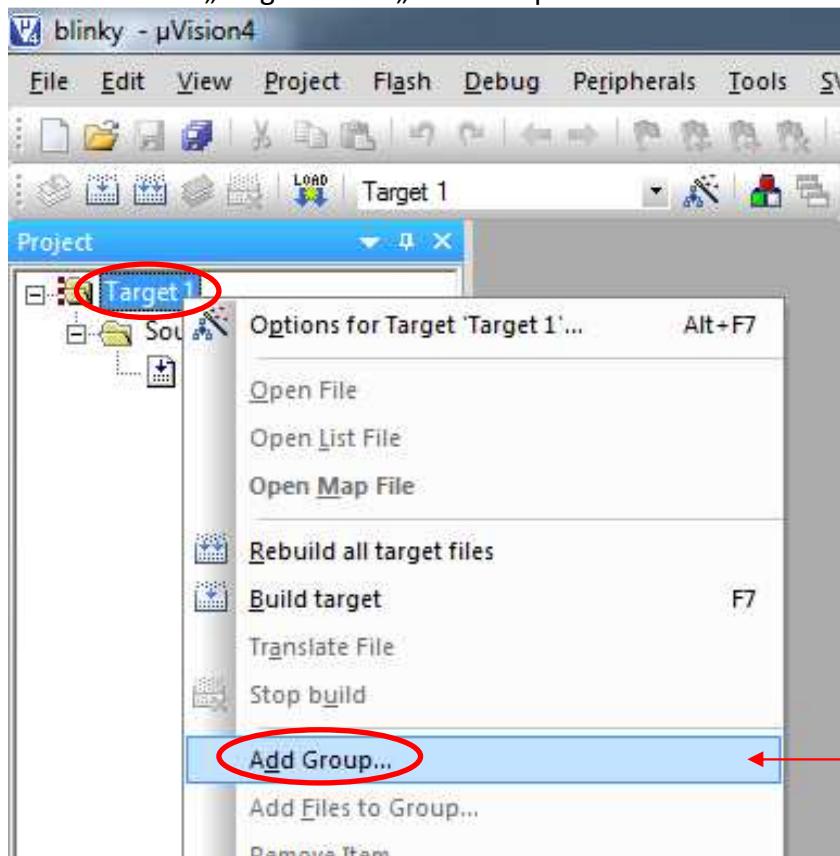
Im Untermenü den „STM32F103RB“ auswählen und mit „OK“ bestätigen.



Es erscheint folgende Meldung welche mit „Ja“ zu bestätigen ist!

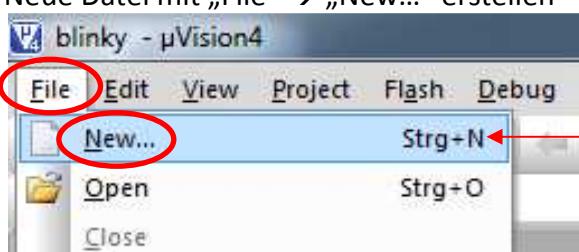


Rechtsklick auf „Target1“ und „Add Group...“ auswählen



Gruppen „Sources“ und „Libaries“ erstellen.

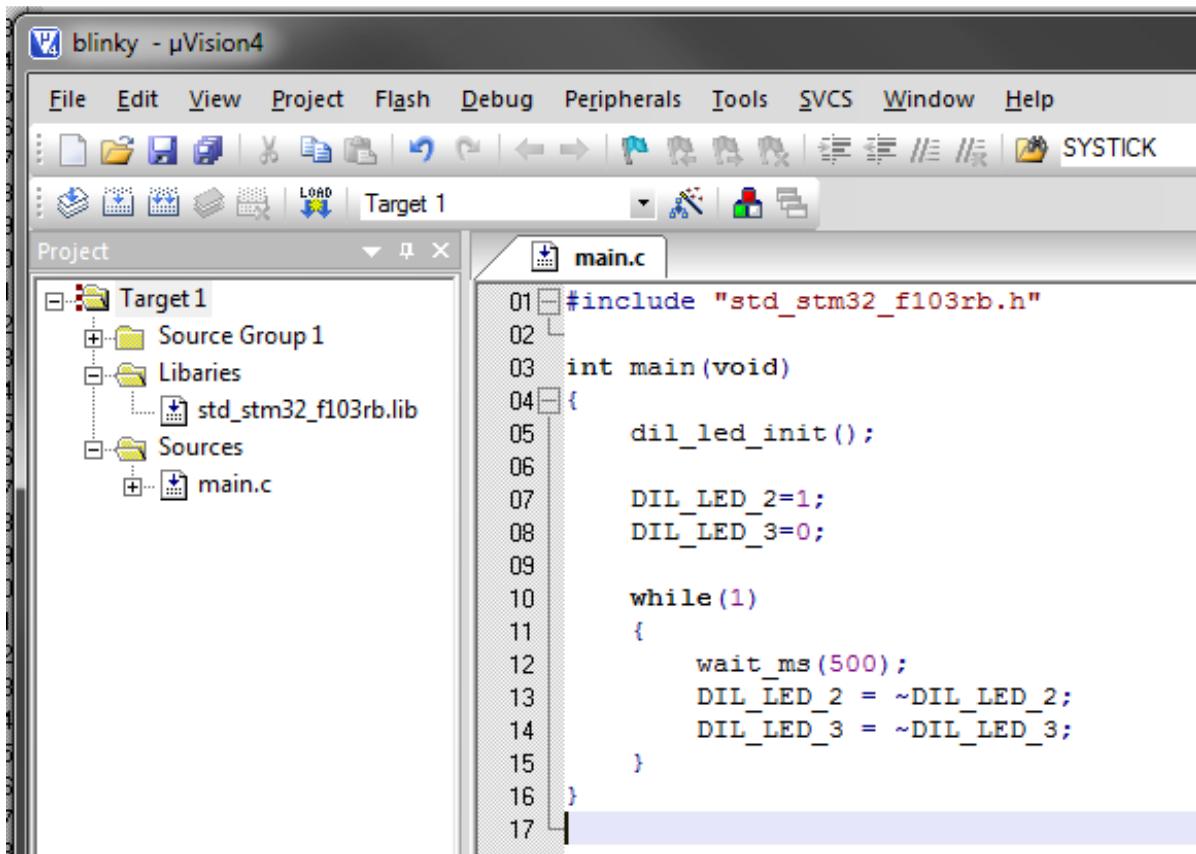
Neue Datei mit „File“ → „New...“ erstellen



Diese Datei wird der Source-Code.

#include „std-stm32-f103rb.h“ muss ganz am Beginn der Datei stehen!

Weiters muss die letzte Zeile der Datei leer sein!

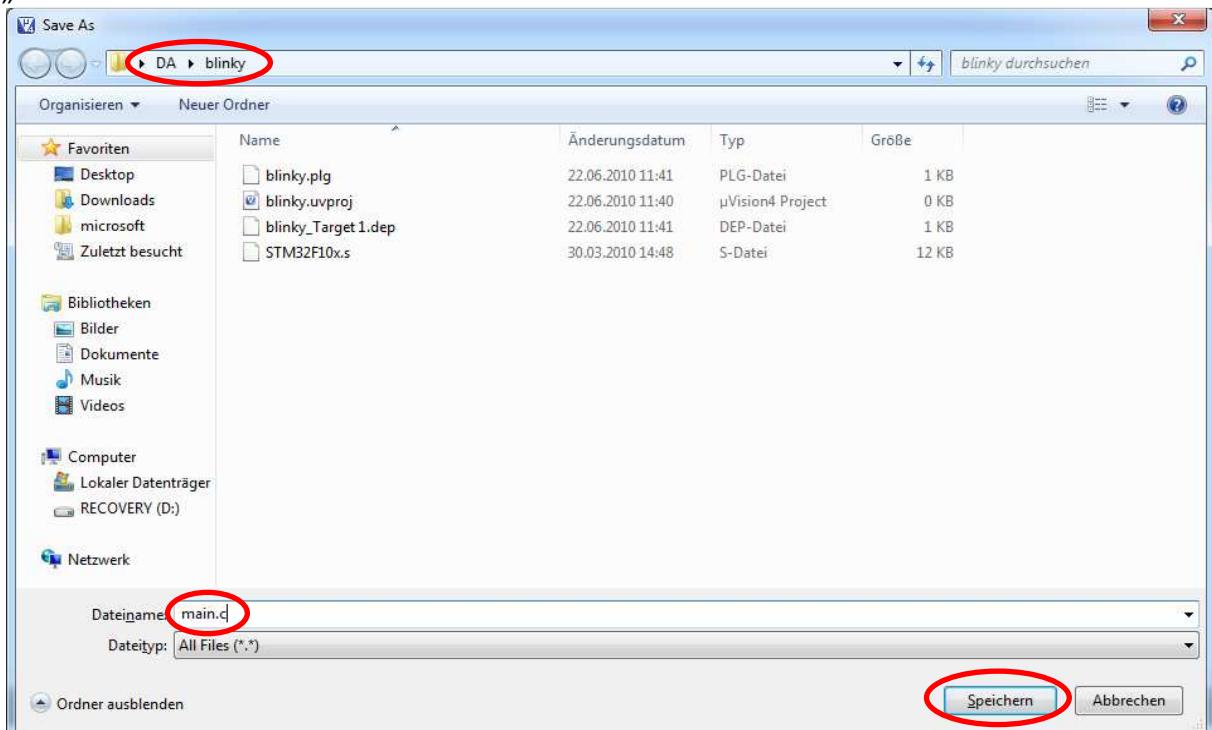


```

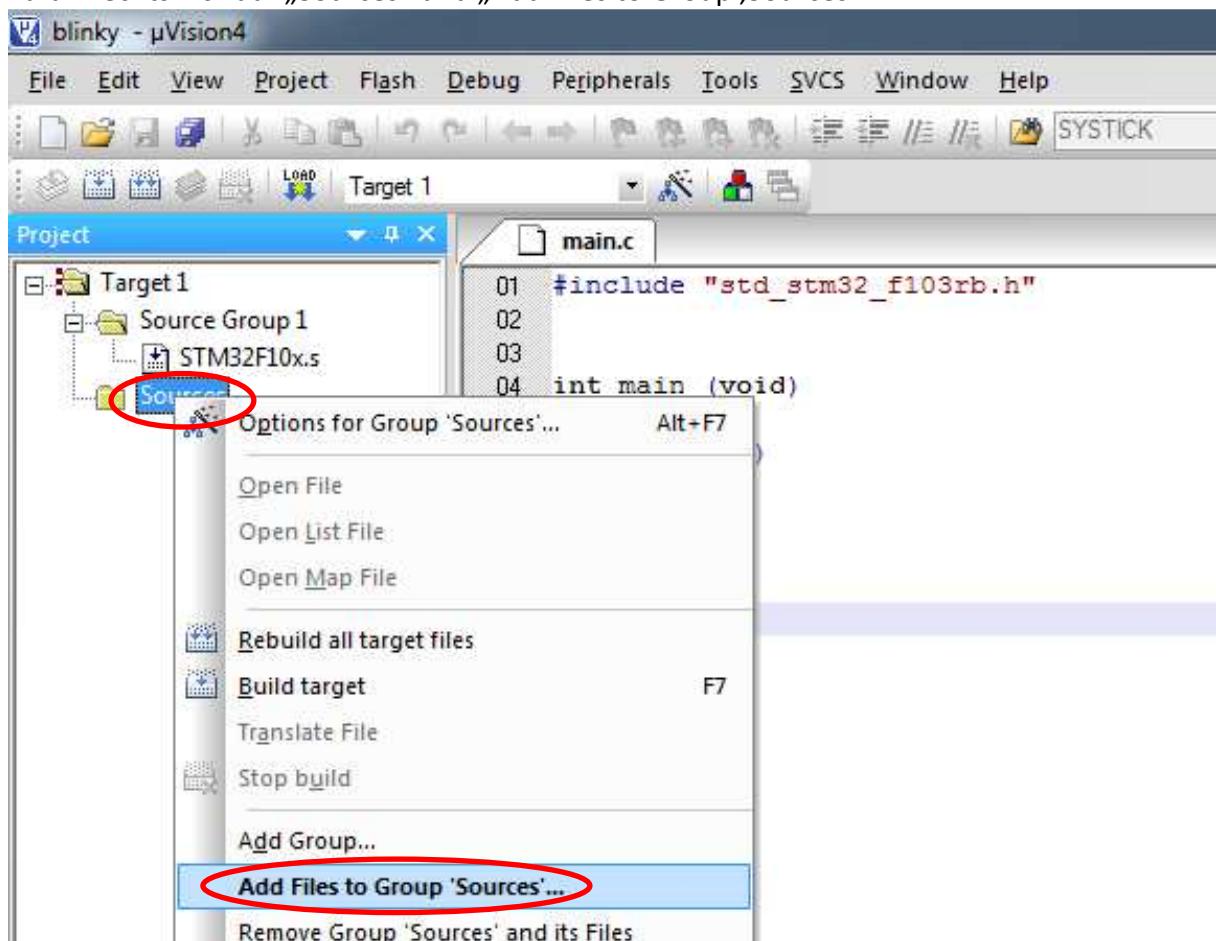
01 #include "std_stm32_f103rb.h"
02
03 int main(void)
04 {
05     dil_led_init();
06
07     DIL_LED_2=1;
08     DIL_LED_3=0;
09
10    while(1)
11    {
12        wait_ms(500);
13        DIL_LED_2 = ~DIL_LED_2;
14        DIL_LED_3 = ~DIL_LED_3;
15    }
16
17

```

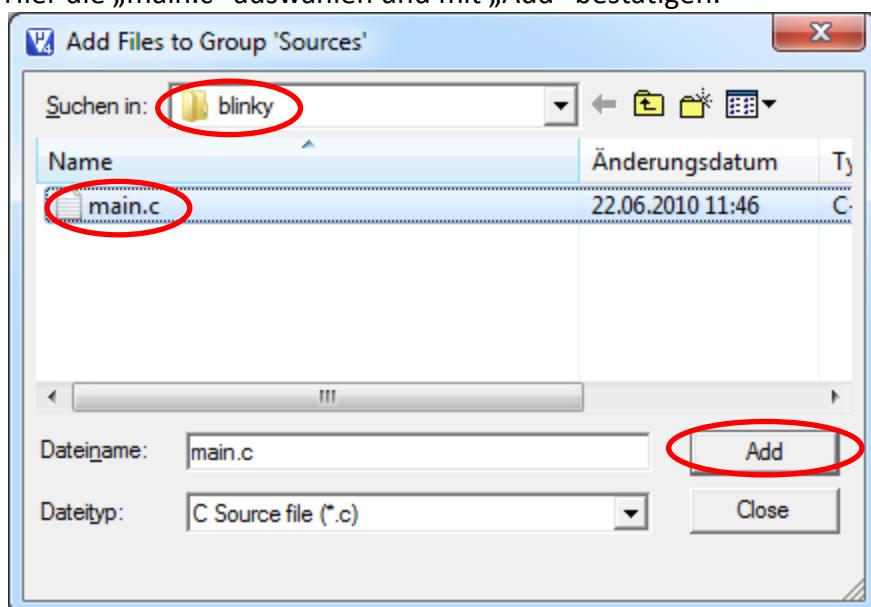
Speichern der Datei im selben Pfad wie das Projekt erstellt wurde unter dem Namen „main.c“.



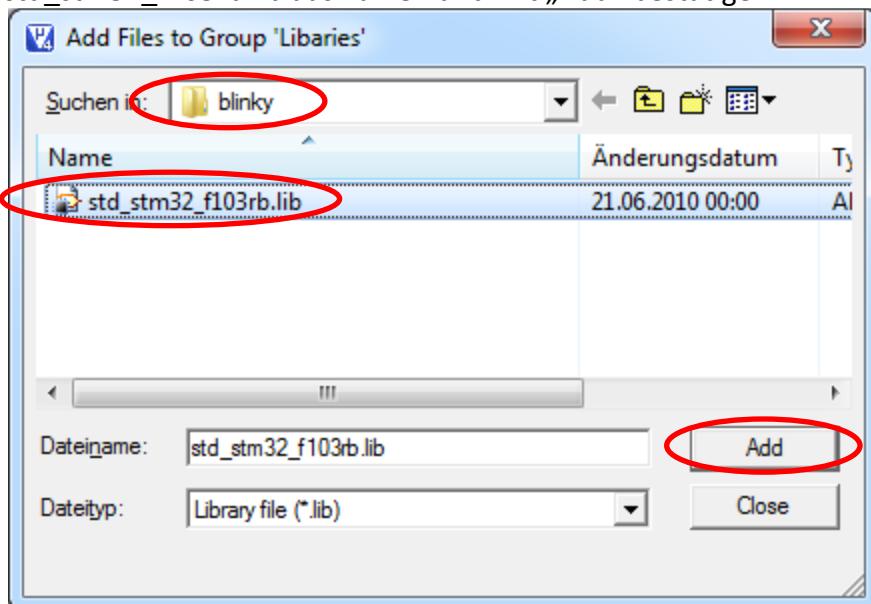
Jetzt muss das Source-File ins Projekt eingebunden werden.
Dafür Rechtsklick auf „Sources“ und „Add Files to Group ,Sources‘...“



Hier die „main.c“ auswählen und mit „Add“ bestätigen.

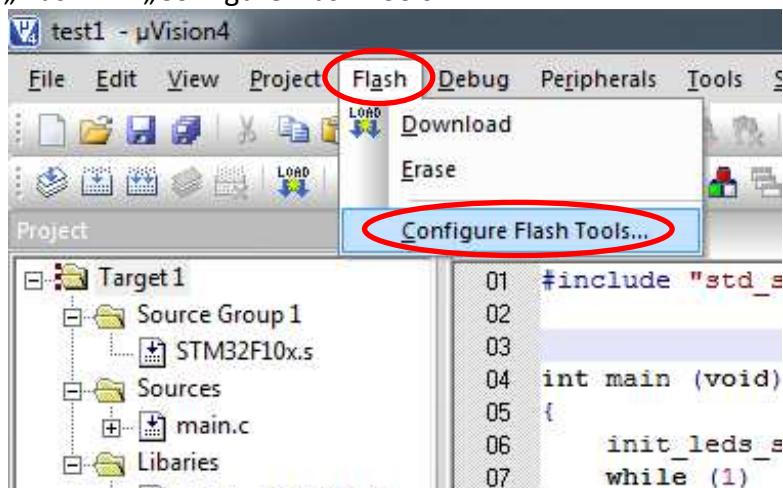


Jetzt muss nur noch die Library in das Projekt integriert werden.
 Rechtsklick auf „Libraries“ und „Add Files to Group „Libaries“...“ auswählen.
 std_stm32_f103rb.lib auswählen und mit „Add“ bestätigen.

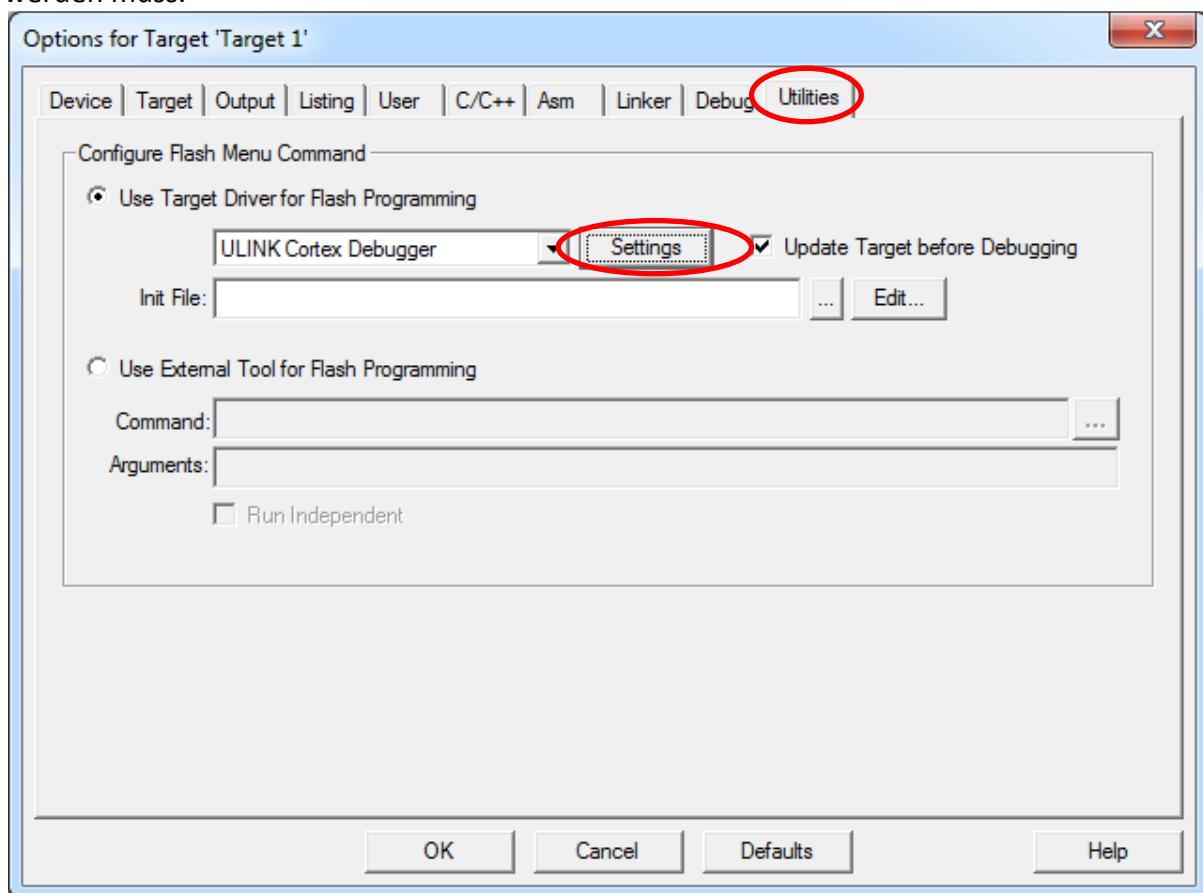


Empfehlenswert ist es auch einzustellen, dass nach dem Download eines Programmes ein automatischer Reset erfolgt. Man erspart sich dadurch das Drücken des Reset-Buttons nach jedem Download.

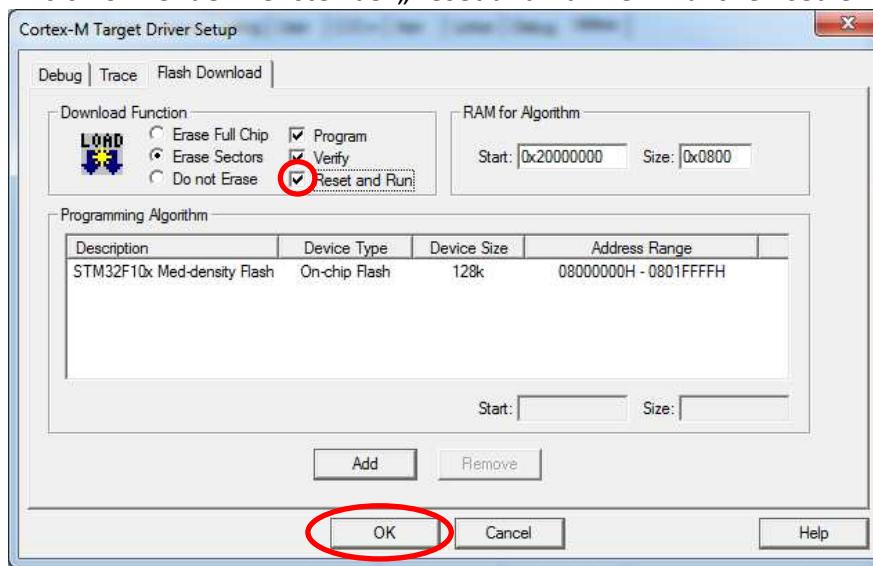
„Flash“ → „Configure Flash Tools...“



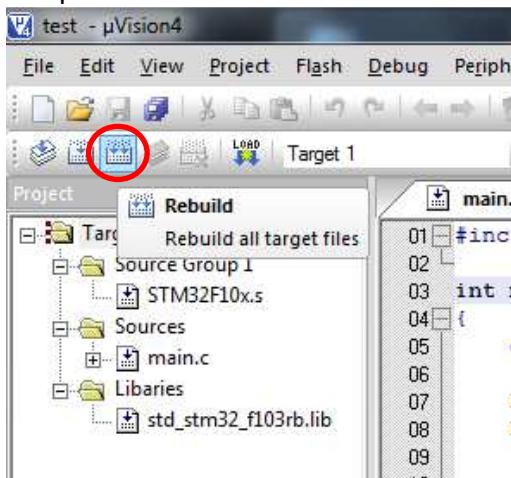
Es erscheint folgendes Fenster wo in der Registerkarte „Utilities“ „Settings“ ausgewählt werden muss.



Im sich öffnenden Fenster bei „Reset and Run“ ein Häkchen setzen und mit „OK“ bestätigen.



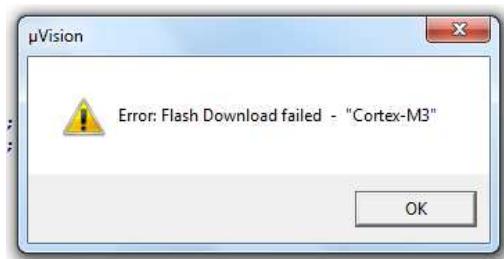
Zum Downloaden des Programms muss zuerst das Programm mithilfe des „Rebuild“-Buttons kompiliert werden.



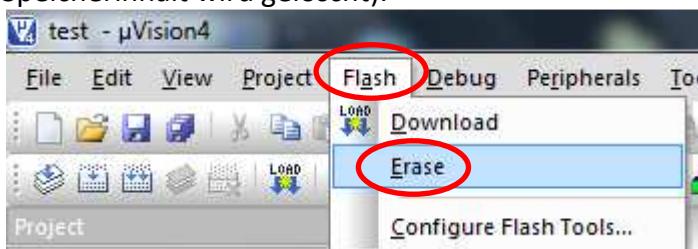
Wenn keine Fehler festgestellt wurden, kann das Programm mithilfe von „Load“ auf den CM3 gespielt werden.



Beim ersten Downloadversuch erscheint möglicherweise diese Fehlermeldung:



Dieser Fehler kann behoben werden indem man unter „Flash“ „Erase“ auswählt (kompletter Speicherinhalt wird gelöscht).



Wenn man nun nochmals „Load“ klickt sollte das Downloaden problemlos funktionieren und das Programm starten.

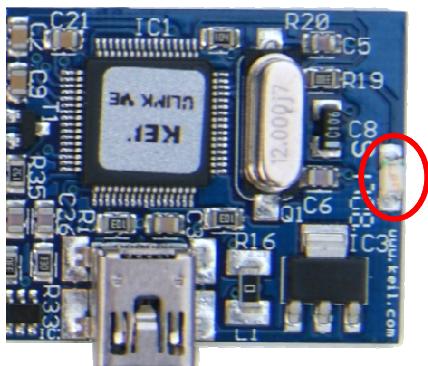
3.3 HTL-Platinen testen

JTAG-Debugger mit DIL-Adapter verbinden (Nase des JTAG-Debuggers muss oben sein/ICs beider Platinen müssen auf der selben Seite sein)

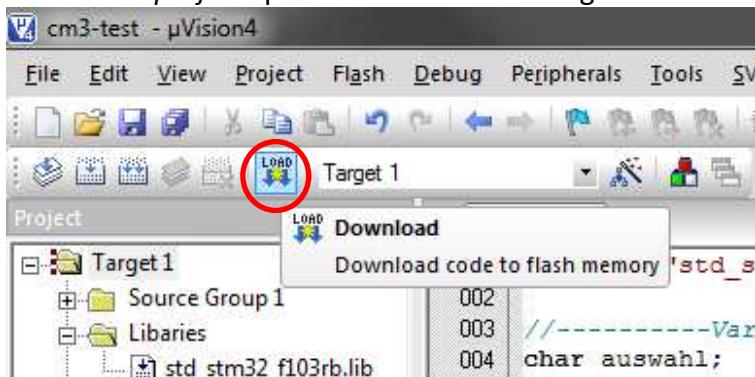


JTAG-Debugger über USB mit PC verbinden

wenn Treiber installiert wurde fadet die blaue LED am JTAG-Debugger



cm3-test.uvproj mit µVision4 öffnen und Programm downloaden



Das Programm zum Testen der HTL-Platten ist nun im Flash des CM3 gespeichert (auch nach einer beliebig langen Unterbrechung der Spannungsversorgung).

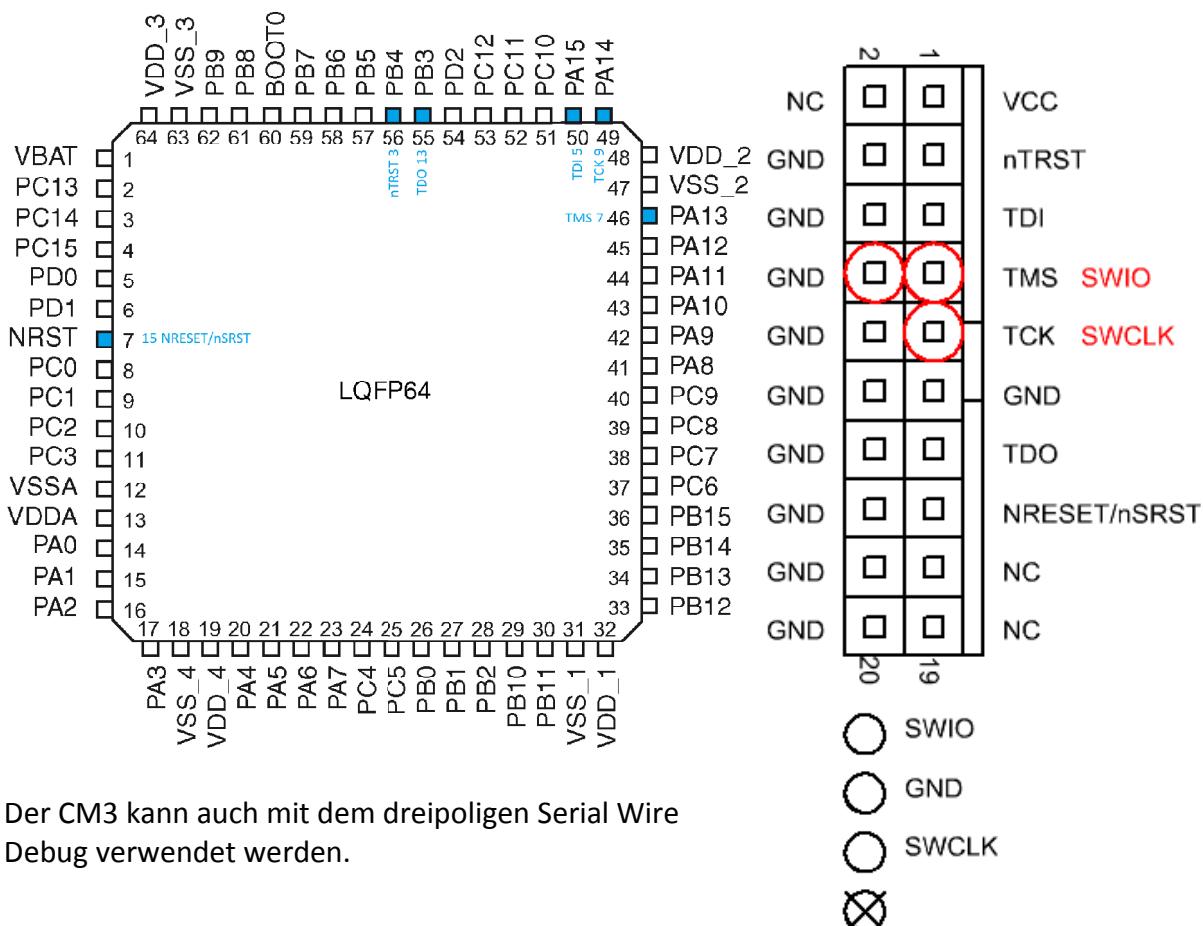
Um die diversen Komponenten zu testen ist ein LCD nötig (muss vor dem Starten des Testprogramms angesteckt werden) und man muss die RS232 Verbindung (9600 Baud) z.B. mit einem RS232-USB – Adapter mit dem Computer verbinden da das Menü des Testprogrammes über diese Schnittstelle ausgegeben wird (hierzu kann am PC das

Programm Putty verwendet werden – http://www.chip.de/downloads/PuTTY-0.60_12997392.html). Weiters muss das USB-Kabel angesteckt werden, um die 5V für das LCD zu haben.

Am Anfang des Testprogrammes (nach jedem Reset) gibt es zum Testen der hardwaremäßigen Verbindungen ein „Portincrement“. Hierbei wird 10 Sekunden lang auf allen Ports (bis auf die Ports welche für den JTAG verwendet werden) ein Rechteck ausgegeben.

Wird nach dem Portincrement kein Menü auf der RS232 ausgegeben bzw. zeigt das LCD nichts an könnten folgende Fehlerquellen überprüft werden:

-) Sind alle Pins des JTAG-Debuggers mit allen Pins der DIL-Platine verbunden?



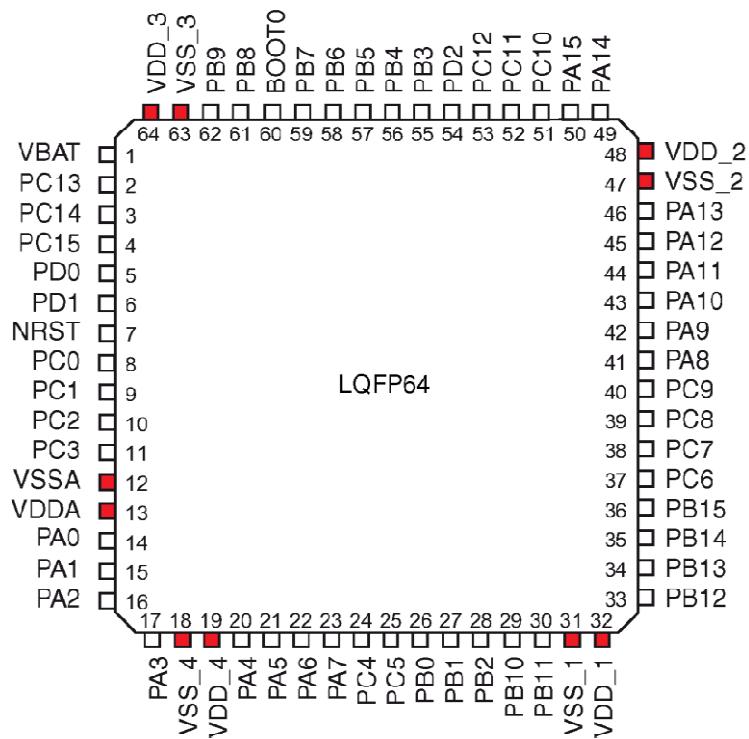
Der CM3 kann auch mit dem dreipoligen Serial Wire Debug verwendet werden.

-) Ist die Reset-Leitung durchgehend auf der DIL-Platine und am JTAG-Debugger?

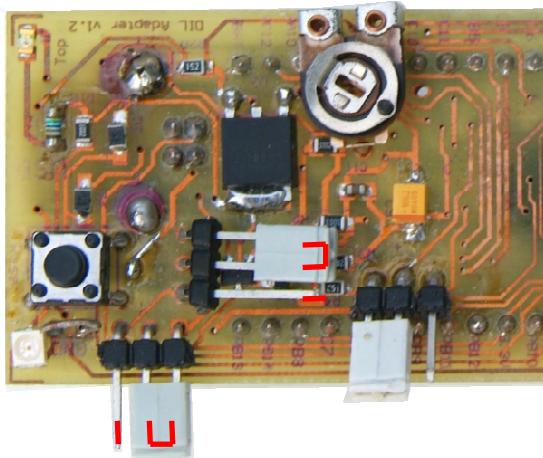
-) Haben die Kondensatoren neben dem Quarz die richtigen Werte ($C2/C5=22\text{pF}$)?



-) Sind alle Versorgungspins des STM32F103RB verbunden (3,3V/Gnd)?



-) Wurden die Jumper richtig gesetzt?



Das Menü sieht folgendermaßen aus:

```

CM3-Testprogramm v1.0
1|DIL-LEDs
2|DIL-Taster
|
3|Inkrementalgeber
4|Joystick
5|NE555/LFU
6|Iwire
7|Piezo
8|Potentiometer
9|V24-Modul
|
a|RoboLab
b|SPI
c|Matrix-Keyboard
d|I2C-UART2
|
e|USB Maus
Menuepunkt waehlen: █

```

Zu den einzelnen Tests:

Während das Menü angezeigt wird, können die LEDs und Schalter der L/S-Platine mithilfe des LCD getestet werden.

Die Schalterstellungen werden am LCD und auf den LEDs angezeigt.

1|DIL-LEDs

Die LEDs werden abwechselnd ein- und ausgeschalten und am LCD wird angezeigt, welche LEDs gerade leuchten sollten und welche nicht. Somit kann überprüft werden, ob diese richtig eingelötet sind bzw. ob die Verbindung zum jeweiligen Port unterbrechungsfrei hergestellt wurde.

Hinweis: Die rote LED (PD2) leuchtet nur, wenn der Jumper „USB/rote LED“ auf der Stellung „rote LED“ ist (siehe S.13 | *DIL-Adapter – Schaltungsbeschreibung – Jumper*).

2|DIL-Taster

Die Taster werden ebenfalls mit dem LCD getestet. Es wird am LCD angezeigt, ob ein Taster gerade gedrückt oder nicht gedrückt ist.

Hinweis: Die unteren DIL-Taster und DIL-LEDs (PA8/PC13) sind hardwaremäßig verbunden. Das bedeutet, dass beim Drücken des Tasters die benachbarte LED automatisch mit leuchtet.

3|Inkrementalgeber

Am LCD wird angezeigt, ob der Taster des Inkrementalgebers gedrückt ist oder nicht. Außerdem wird eine Zählvariable angezeigt, welche bei Drehung im Uhrzeigersinn größer und gegen den Uhrzeigersinn kleiner wird.

4|Joystick

Am LCD werden die 5 Ausgänge des Joysticks angezeigt (Links, Oben, Taster, Unten, Rechts) und jeweils ob dieser Ausgang „1“ oder „0“ ist. Es können auch 2 Richtungen gleichzeitig gedrückt werden (z.B. rechts unten).

5|NE555/LFU

Bei diesem Test ist es wichtig dass vor dem aktivieren des Tests der Jumper „Piezo“ geöffnet wurde bzw. der Jumper für „PBO“ auf „NE555“ oder „LFU“ ist.

Es wird am LCD eine Variable, welche die Zeit zwischen einer positiven und negativen Flanke misst, angezeigt. Somit muss sich diese ändern wenn der LFU zugedeckt wird bzw. wenn an den Potis oder den Jumpern des NE555 etwas verändert wird.

6|1wire

Vor dem Aktivieren dieses Tests muss der Jumper „Piezo“ geöffnet werden und der Jumper „PBO“ auf „1wire“ gesteckt werden.

Es erscheint am LCD die Umgebungstemperatur des 1wire Temperatursensors.

7|Piezo

Der Jumper „Piezo“ muss geschlossen werden.

Nun sollte man vier hintereinander folgende Töne hören.

8|Potentiometer

Bei diesem Test werden die zwei Potentiometer (eines am DIL-Adapter und eines auf der L/S-Platine) getestet. Es erscheinen am LCD 2 Bargraphen, welche sich beim Drehen am jeweiligen Potentiometer ändern müssen.

9|V-24 Modul

Hierbei wird getestet, ob das V-24 Modul funktioniert. Jedes Zeichen, das über das V24-Modul an den CM3 geschickt wird, wird um Eins inkrementiert und wieder zurückgeschickt.

a|RoboLab

Bei diesem Test können die Pins der RoboLab-Schnittstelle mithilfe der Testplatine getestet werden. Es wird auf den LEDs ein Lauflicht ausgegeben.

b|SPI

Hier werden die Pins der SPI-Schnittstelle mithilfe eines Lauflichtes auf der Testplatine getestet.

c|Matrix-Keyboard

Mit diesem Test werden die Pins der Schnittstelle für das Matrix-Keyboard mit der Testplatine getestet (durch Ausgabe eines Lauflichts).

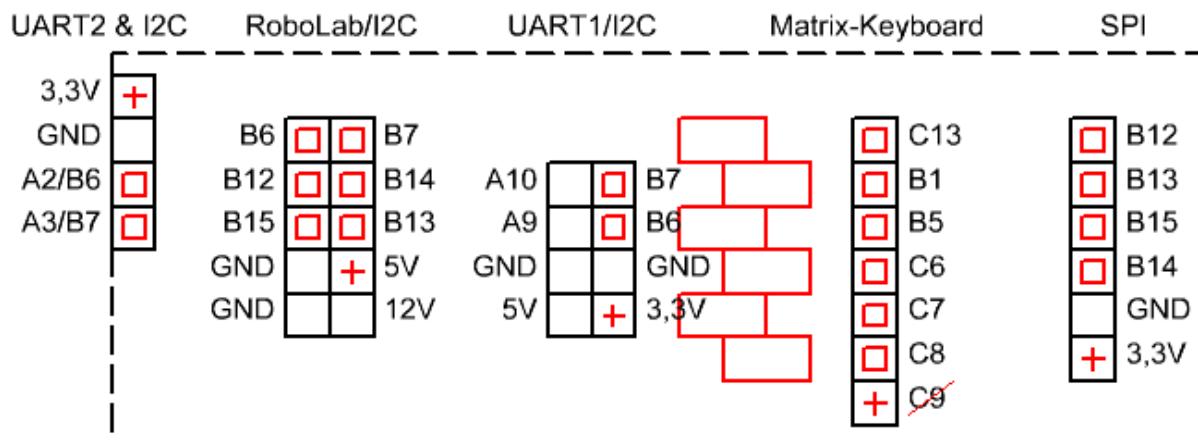
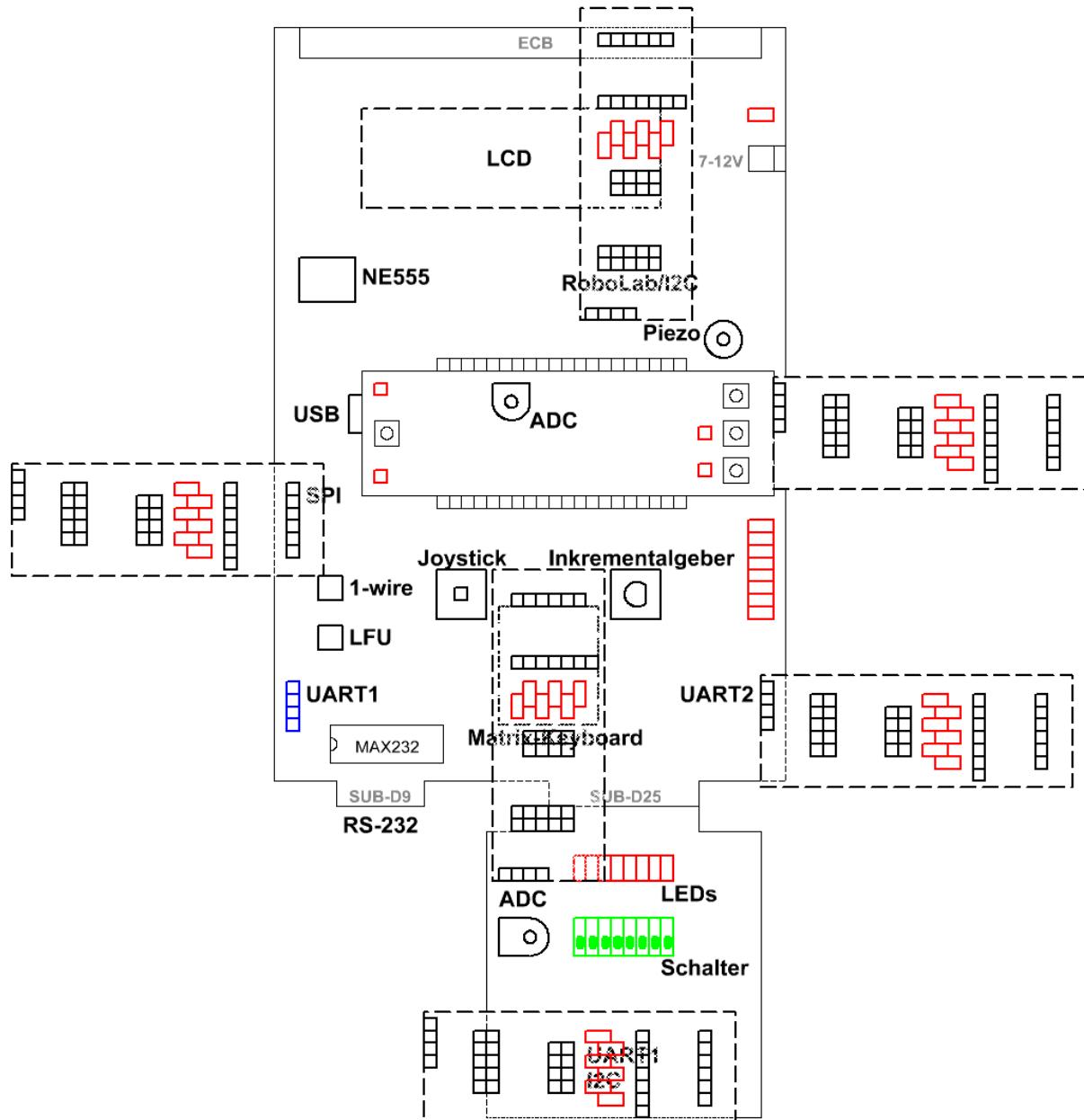
d|I2C UART

Bei diesem Test wird die I2C und UART-Schnittstelle auf der L/S-Platine mithilfe der Testplatine getestet (durch Ausgabe eines Lauflichts).

e|USB Maus

Mit diesem Test meldet sich der DIL-Adapter als Maus am Computer an, die mit dem Joystick getestet werden kann.

Wird während dem Reset der DIL-Taster-1 (PC5) gedrückt, meldet sich der DIL-Adapter als Maus am Computer an und fährt in einem Rechteck herum.

Steckmöglichkeiten der Schnittstellen-Testplatine:

Red '+' symbol indicates power supply used.

Red square indicates LED output.

3.4 Funktionen in Library

Variablennamen: siehe Blockschaltbild „BSB_v1.0.pdf“

`void dil_taster_init(void)`

initialisiert die drei Taster des DIL-Adapters

`void dil_led_init(void)`

initialisiert die drei LEDs des DIL-Adapters

`void joy_inc_init(void)`

initialisiert die Portleitungen für den Inkrementalgeber und den Joystick

`void B0_init_out(void)`

initialisiert den Port B0 als Output

`void B0_init_in(void)`

initialisiert den Port B0 als Input

`char get_switches(void)`

liefert als Returnwert die aktuelle Stellung der Schalter auf der L/S-Platine

`void set_leds (char value)`

gibt den hexadezimalen Wert „value“ auf den 8 LEDs der L/S-Platine aus

`void init_leds_switches(void)`

initialisiert die LEDs und Schalter der L/S-Platine, je zwei Taster und LEDs des DIL-Adapters und den Inkrementalgeber bzw. Joystick

`void wait_ms(int ms)`

wartet „ms“ Millisekunden

`void wait_10us(void)`

wartet 10 Mikrosekunden

`void uart_init(unsigned long baudrate)`

initialisiert den UART mit der Baudrate „baudrate“

`void uart_setpos(char x,char y)`

positioniert den Cursor am Terminal

`char uart_get_char(void)`

liefert als Returnwert ein empfangenes ASCII-Zeichen

`void uart_put_string(char *string)`

schickt den String ab dem übergebenen Pointer „*string“ ans Terminal

`void uart_clear(void)`

löscht die Zeichen am Terminal (VT100 kompatibel)

`void lcd_init (void)`

initialisiert das LCD (muss angesteckt sein!!!)

`void lcd_set_cursor (int line_nr, int column_nr)`

positioniert den Cursor des LCD in der Zeile „line_nr“ (0-3) und der Spalte

„column_nr“ (0-15)

`void lcd_clear (void)`

löscht die Zeichen auf dem LCD

`void lcd_put_string (char *string)`

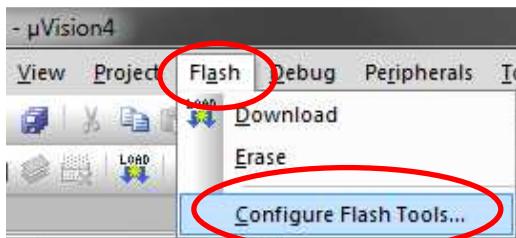
schreibt den String ab dem übergebenen Pointer „*string“ auf das LCD

`unsigned short int adc1_convert(unsigned char channel)`

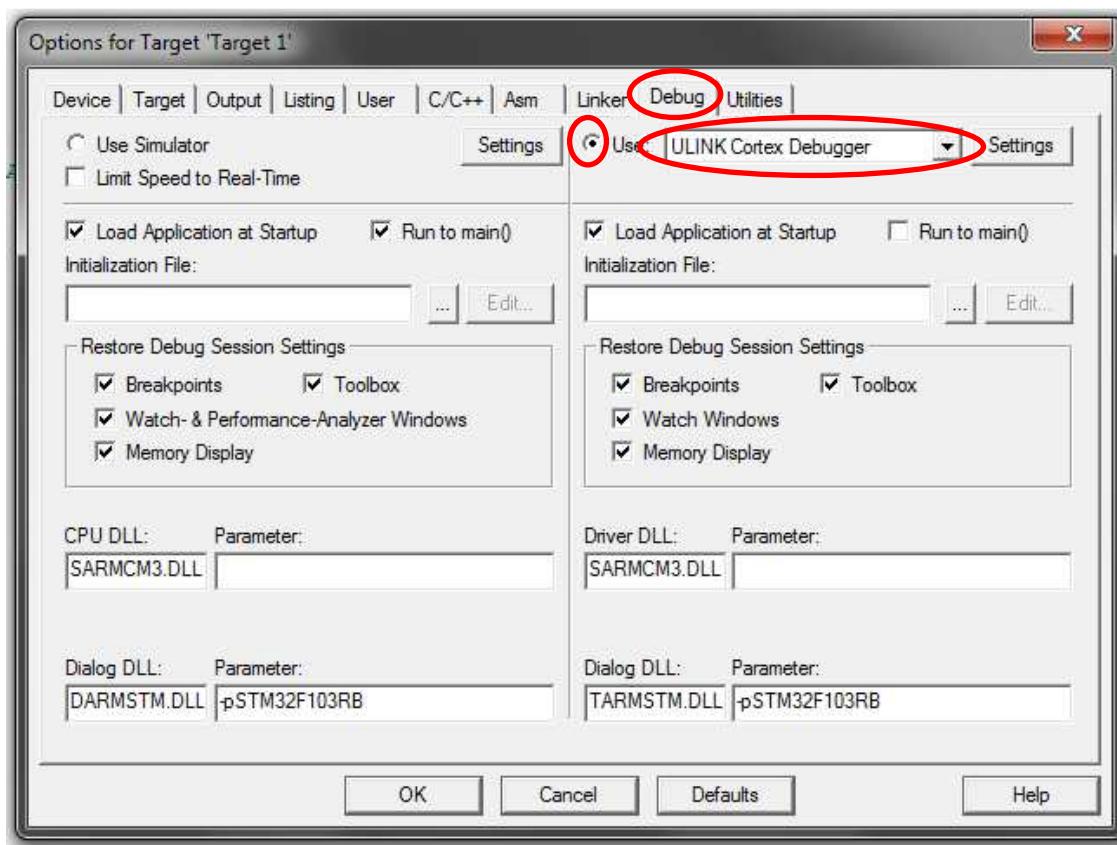
liefert als Return den aktuellen Wert von ADC1 für Kanal „channel“

3.5 Debuggen eines C-Programms

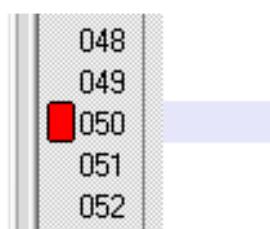
Um den Debugger zu konfigurieren ist wie folgt vorzugehen:
Im Menü „Flash“ den Punkt „Configure Flash Tools...“ auswählen.



Im sich öffnenden Fenster den Menüreiter „Debug“ auwählen und bei „Use:“ den Punkt setzen und in der dazugehörigen Auswahl „ULINK Cortex Debugger“ auswählen.



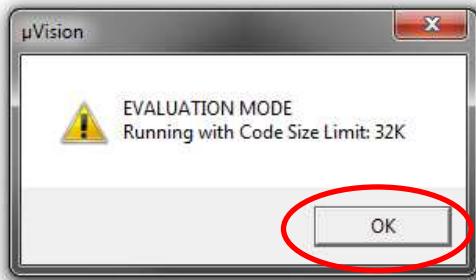
In gewünschter/gewünschten Zeilen Breakpoint setzen durch Doppelklick am Zeilenbeginn



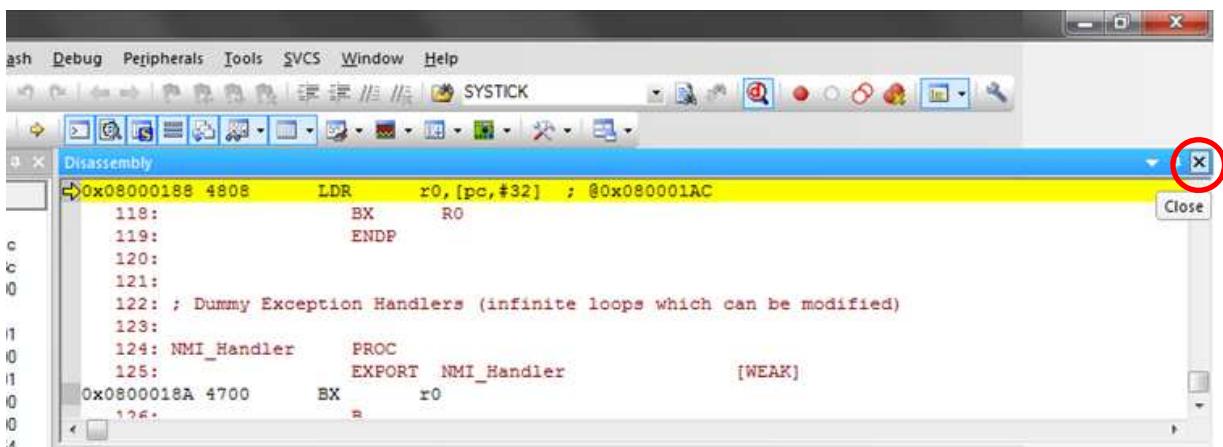
Der Debugger wird gestartet durch Drücken auf folgendes Symbol:



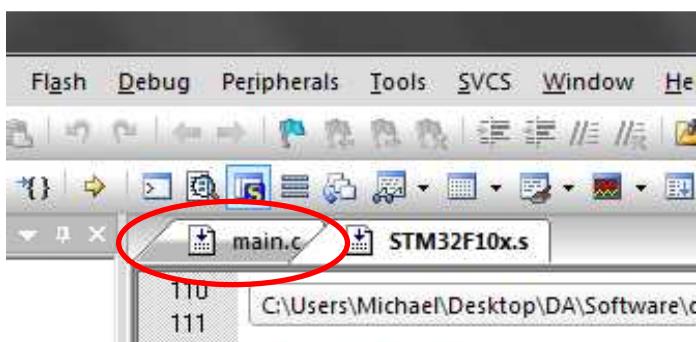
Bei einer nicht aktivierte Version von uVision4 erscheint folgende Meldung welche mit OK bestätigt werden muss.



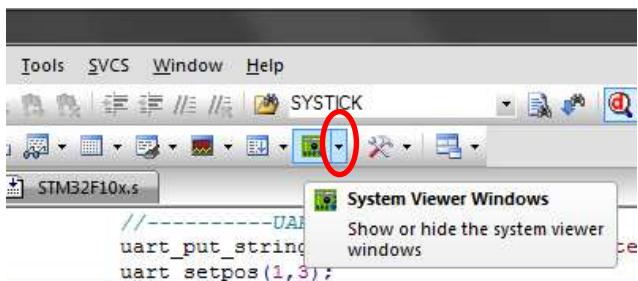
Für den klassischen Debug-Vorgang wird das Disassembly-Fenster nicht benötigt und kann mit dem X am rechten oberen Rand geschlossen werden.



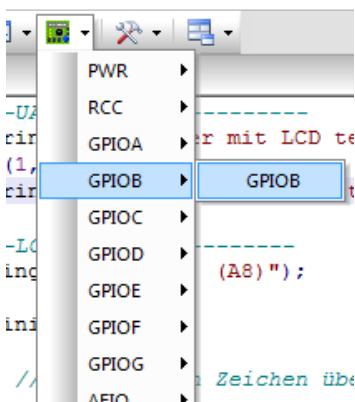
Um den Debug-Vorgang im eingenen Code verfolgen zu können muss das *.C-File im Reiter wieder in den Vordergrund geholt werden.



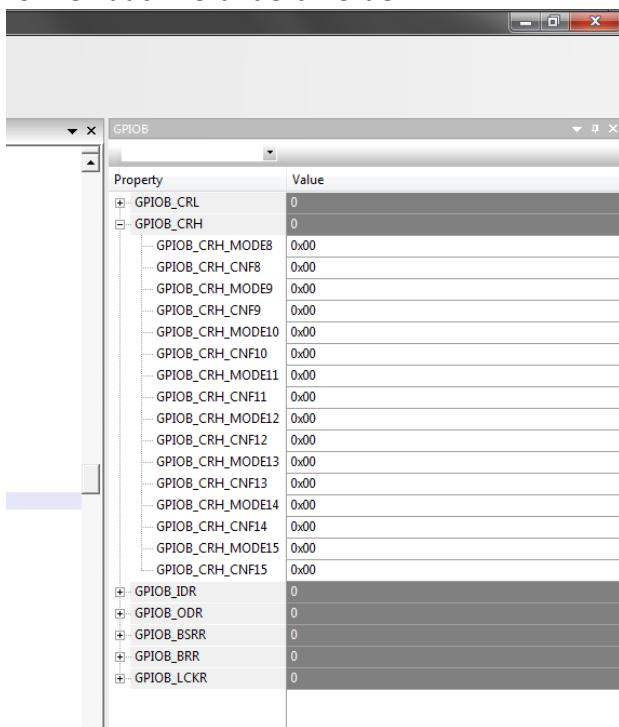
Eine Möglichkeit, sich die Register anzusehen, ist folgende:
Auf den zum Button „System Viewer Windows“ zugehörigen Pfeil klicken



Es erscheint eine Liste mit den Funktionsblöcken der Register, welche angezeigt werden können.



Nach dem Auswählen werden im rechten Bereich des Fensters die Register angezeigt und können auch verändert werden.



Eine andere Möglichkeit, die Register angezeigt zu bekommen ist mit dem Watch1-Fenster im unteren Bereich.

Um zusätzliche Register anzuzeigen muss der Reiter „Symbols“ ausgewählt werden

Im Unterpunkt „Peripheral Registers“ sind alle Register mit ihrer Adresse und dem Datentyp aufgelistet.

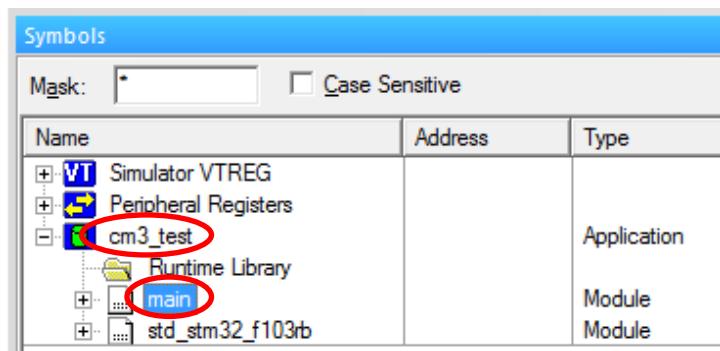
Name	Address	Type
Peripheral Registers		
ADC1_CR1	0x40012404	ulong
ADC1_CR2	0x40012408	ulong
ADC1_DR	0x4001244C	ulong
ADC1_HTR	0x40012424	ulong
ADC1_JDR1	0x4001243C	ulong
ADC1_JDR2	0x40012440	ulong
ADC1_JDR3	0x40012444	ulong
ADC1_JDR4	0x40012448	ulong
ADC1_JOFR1	0x40012414	ulong
ADC1_JOFR2	0x40012418	ulong
ADC1_JOFR3	0x4001241C	ulong
ADC1_JOFR4	0x40012420	ulong
ADC1_JSQR	0x40012438	ulong
ADC1_LTR	0x40012428	ulong
ADC1_SMPR1	0x4001240C	ulong
ADC1_SMPR2	0x40012410	ulong
ADC1_SQR1	0x4001242C	ulong
ADC1_SQR2	0x40012430	ulong
ADC1_SQR3	0x40012434	ulong

Zum Watch1-Fenster werden sie hinzugefügt durch einen Rechtsklick auf das jeweilige Register und auswählen von „Watch 1“ im Untermenü „Add ,REGISTER‘ to...“

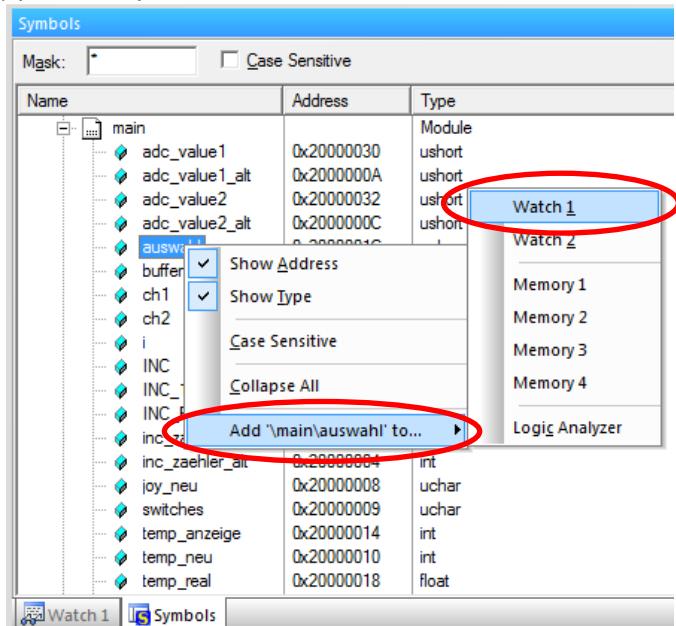
Name	Address	Type
Peripheral Registers		
ADC1_CR1	0x40012404	ulong
ADC1_CR2	0x40012408	ulong
ADC1_DR	0x4001244C	ulong
ADC1_HTR	0x40012424	ulong
ADC1_JDR1	0x4001243C	ulong
ADC1_JDR2	0x40012440	ulong
ADC1_JDR3	0x40012444	ulong
ADC1_JDR4	0x40012448	ulong
ADC1_JOFR1	0x40012414	ulong
ADC1_JOFR2	0x40012418	ulong
ADC1_JOFR3	0x4001241C	ulong
ADC1_JOFR4	0x40012420	ulong
ADC1_JSQR	0x40012438	ulong
ADC1_LTR	0x40012428	ulong
ADC1_SMPR1	0x4001240C	ulong
ADC1_SMPR2	0x40012410	ulong
ADC1_SQR1	0x4001242C	ulong
ADC1_SQR2	0x40012430	ulong
ADC1_SQR3	0x40012434	ulong

Ebenfalls können alle Variablen, die in den ins Projekt eingebunden Source-Dateien verwendet sind, ins Watch1-Fenster übernommen werden.

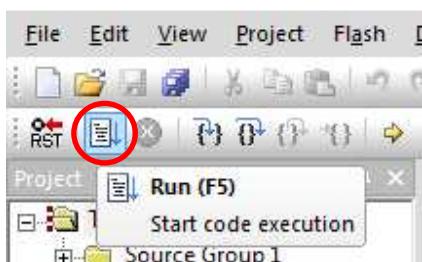
Dazu den Unterpunkt „PROJEKTNAME“ und die gewünschte Source-Datei wählen.



Jetzt kann die Variable mit Rechtsklick und Klick auf „Watch 1“ im Untermenü „Add \SOURCE\VARIABLE‘ to...“ ins Watch1-Fenster übernommen werden.



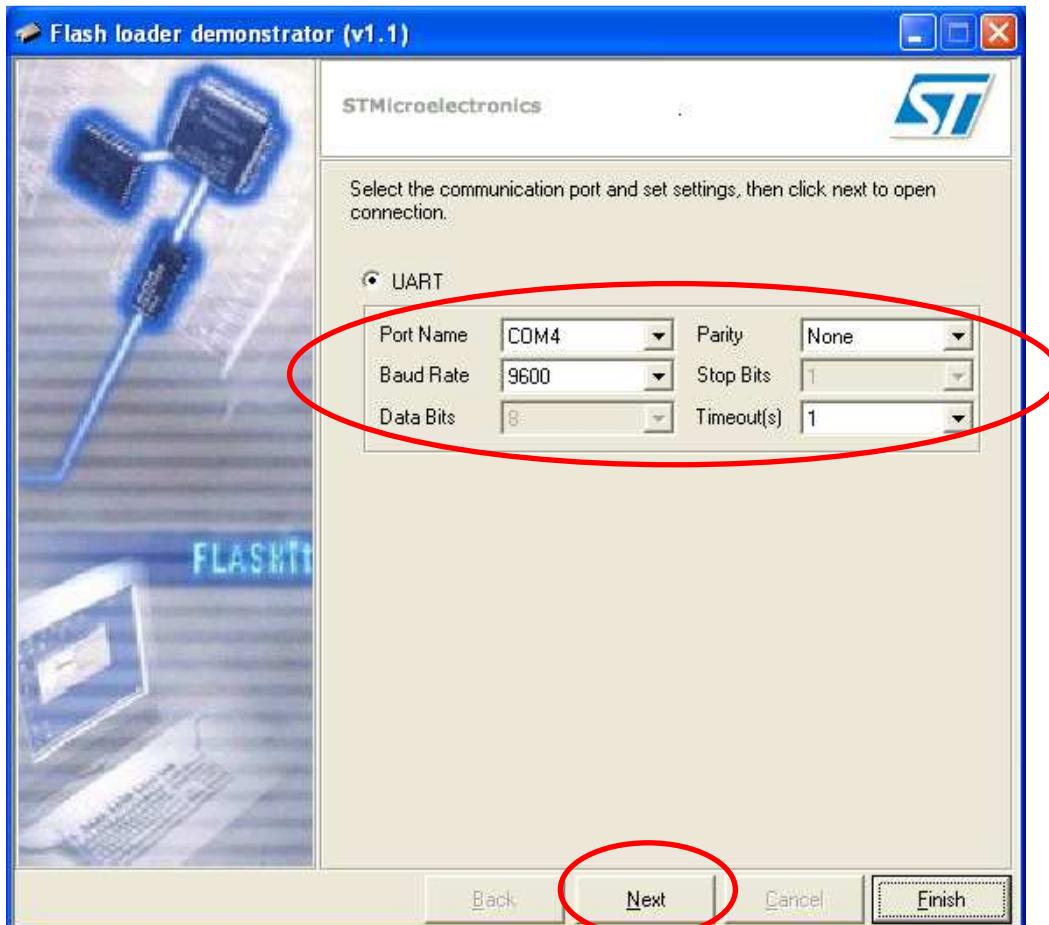
Mit Klick auf „Run“ wird der Debugger gestartet und das Programm läuft bis zum ersten gesetzten Breakpoint



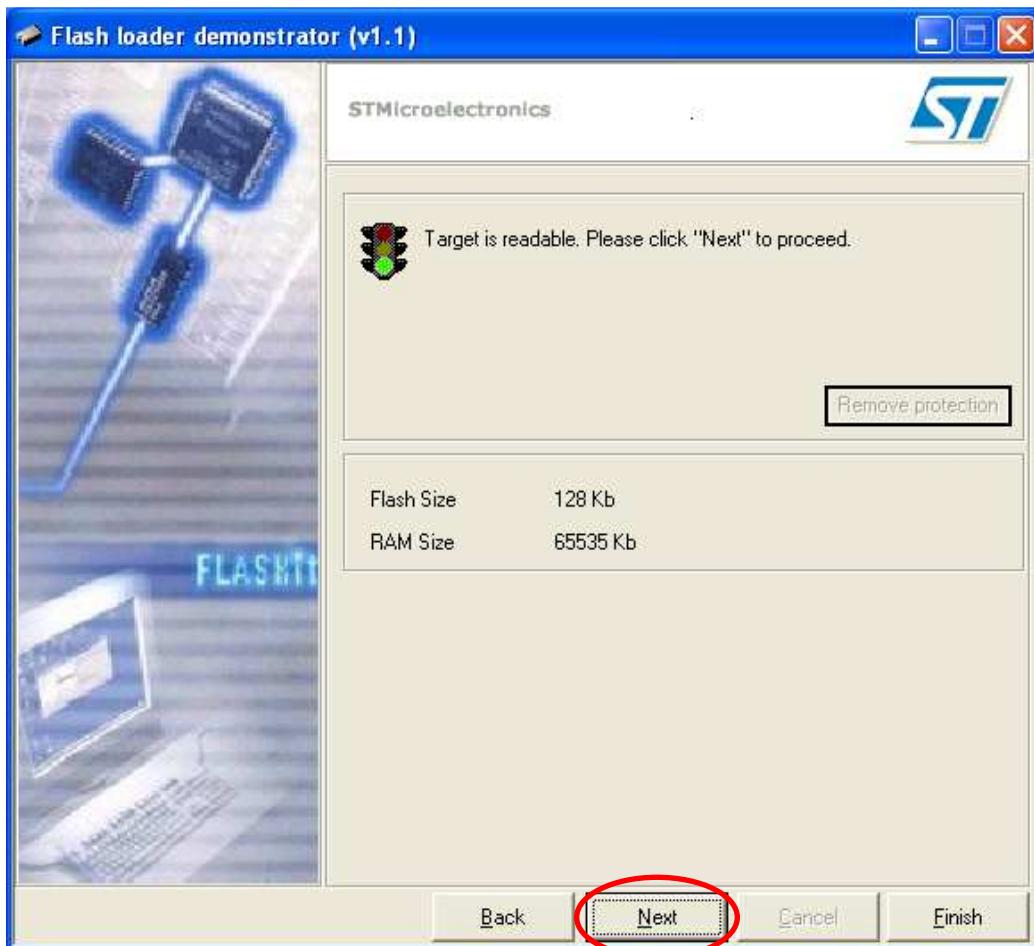
3.6 Über UART flashen

Um ein Programm über den UART zu flashen, zum Beispiel wenn kein JTAG-Debugger vorhanden ist, muss wie folgt vorgegangen werden:

Auf der DIL-Platine muss der Jumper Boot0 auf der Stellung „V24“ sein und ein Reset durchgeführt werden. „Flash loader demonstrator v1.1“ starten und Einstellungen wie in folgender Abbildung vornehmen und auf „Next“ klicken.



Wenn der Cortex bereit zum Daten empfangen ist, erscheint folgendes Fenster:



Wenn nicht können folgende Fehlermeldungen erscheinen:

„*Cannot open the COM port*“:

Am gewählten COM-Port ist nichts angeschlossen oder der gewählte COM-Port wird von einer anderen Anwendung verwendet, welche geschlossen werden muss.

„*Unrecognized device*“:

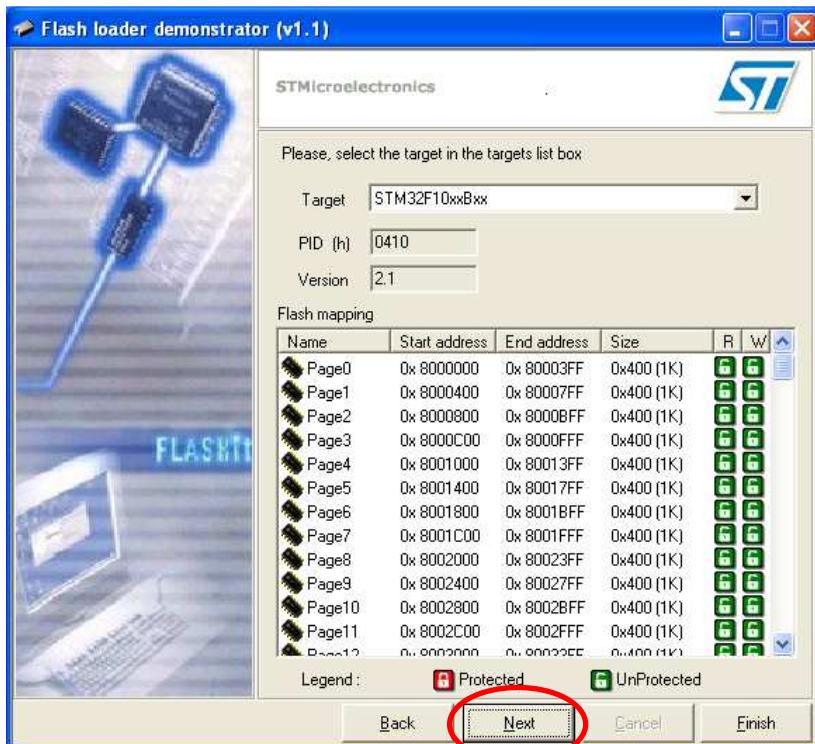
Der CM3 antwortet nicht mit den gewünschten Bits. Nach einem Reset des CM3 sollte das Problem behoben sein.

„*No response from the target*“:

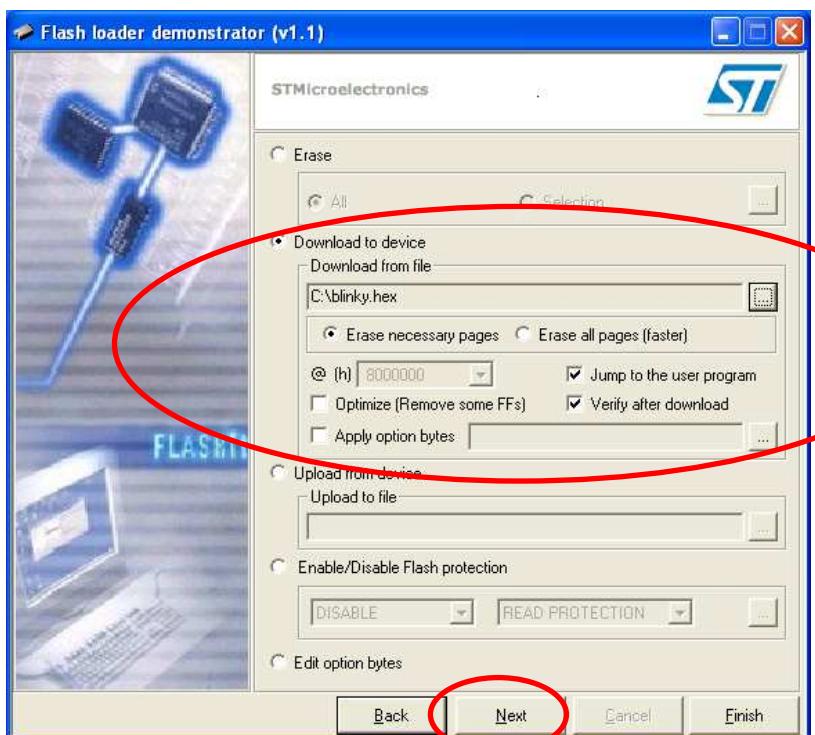
Der Jumper Boot0 ist nicht auf der Stellung „V24“.

Mit „Next“ fortfahren.

Überprüfen ob bei Target der gewünschte Microcontroller aufgelistet ist und mit „Next“ fortfahren.



Um ein Programm auf den CM3 zu spielen den Punkt bei „Download to device“ markieren, das *.hex-File auswählen und die Häkchen wie in der Abbildung gezeigt setzen.

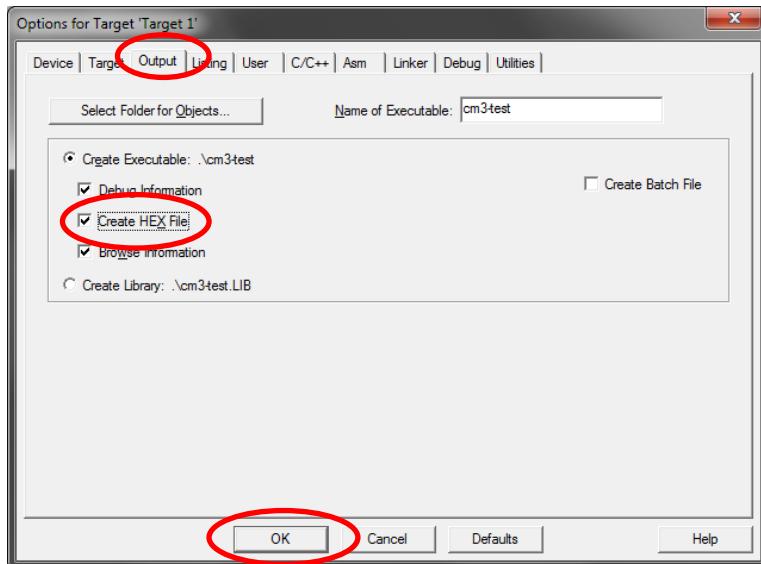


Anschließend mit „Next“ fortfahren und das Programm auf den CM3 zu spielen.

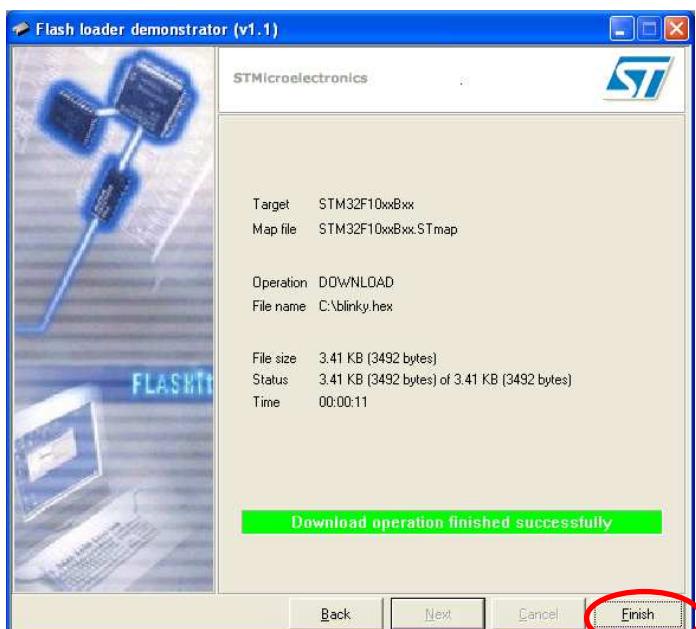
Das *.hex-File wird in uVision4 folgendermaßen generiert:



Die „Target Options...“ öffnen und im Menüreiter „Output“ bei „Create HEX File“ ein Häkchen setzen. Jetzt wird beim Komplilieren des Projektes die *.hex-Datei auch generiert und im Projektpfad abgespeichert.



Der erfolgreiche Download wird in folgendem Fenster bestätigt und das Programm kann mit „Finish“ wieder geschlossen werden.



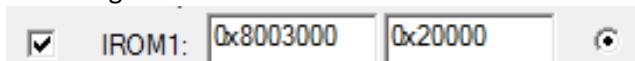
3.7 Über USB flashen

Da der STM32F103 standardmäßig nur einen UART-Bootloader hat, kann man diesen Chip eigentlich nicht über USB flashen. Da sich der STM32F103 selber flashen kann, ist es möglich ein USB Gerät zu implementieren welches ein Programm über USB einliest und dann in den Flash schreibt. Dieses Gerät nennt sich DFU (Device Firmware Update). Das Problem dabei ist, dass der Chip beim Booten nicht weiß, ob er sich beim PC als DFU anmelden soll oder ob er das geflashte Programm starten soll. Weiters darf das eigentliche Programm nicht am Anfang des Speicherbereichs stehen, da dort das DFU Programm steht. Im Folgenden wird nun erklärt wie man den DIL-Adapter über USB flashen kann.

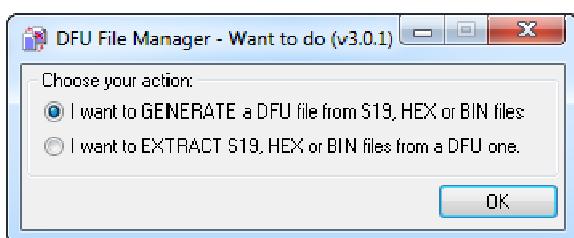
- 1.) „DFU_v1“ öffnen und mit uVision 4 kompilieren
- 2.) Programm über JTAG oder über den UART wie gewohnt flashen.
- 3.) Installation des DFU – Treibers

Im Archiv „DFU_v1“ befindet sich der Ordner DFU-Treiber.
In diesem Ordner sind dann die Treiberinstallationsdateien enthalten.
Je nach Betriebssystem nun die dpinst_xxx.exe starten.
Nun sollte der Treiber installiert sein.
- 4.) Nun ein beliebiges Projekt mit uVision öffnen, welches geflashed werden soll
- 5.) Unter dem Punkt „Projekt -> Options for Target“ die Registerkarte „Output“ auswählen und ein Häkchen bei „Create HEX File“ machen
- 6.) Nun zur Registerkarte „Target“ wechseln und diese Zeile
 

durch folgendes ersetzen:

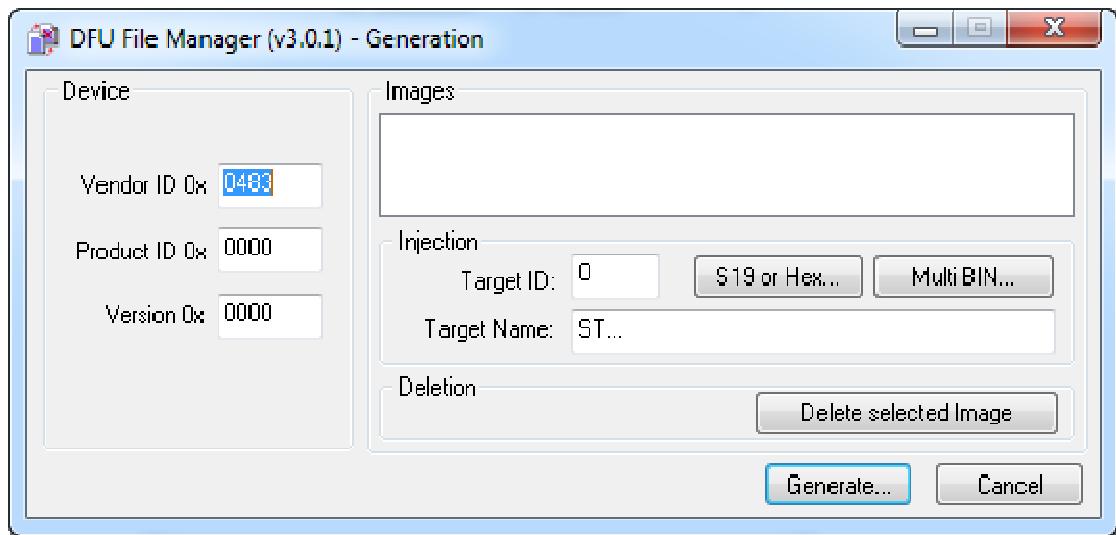


Diese Zeile setzt das eigentliche Programm hinter das DFU – Programm
- 7.) Projekt neu kompilieren
- 8.) Unter „DFU_Flashprogramme“ „DfuFileMgr.exe“ Öffnen



„I want to GENERATE a DFU file from S19,HEX or BIN files“ Auswählen und OK drücken

9.)



mit „S19 or Hex...“ das Hex-File auswählen das uVision erzeugt hat und mit „Generate...“ das .dfu – File erzeugen lassen.

10.) Nun muss das DFU – File nur noch geflashed werden

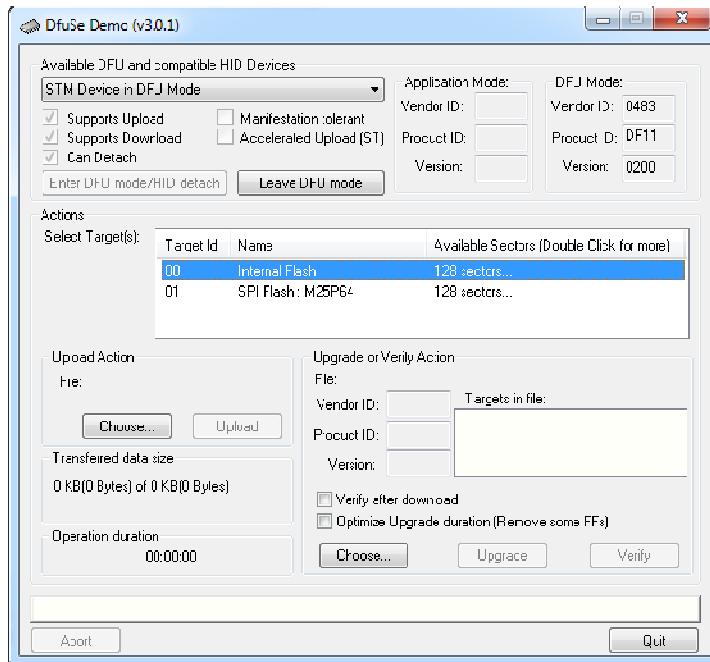
Dazu den Reset-Button am DIL-Adapter und den Schalter S1 drücken

Danach den Reset-Button auslassen und ca. 1 Sekunde später S1 loslassen.

Jetzt ist das DFU-Device gestartet.

Nun das Programm DfuSe Demo(v3.0.1) starten

Es erscheint folgende Oberfläche:



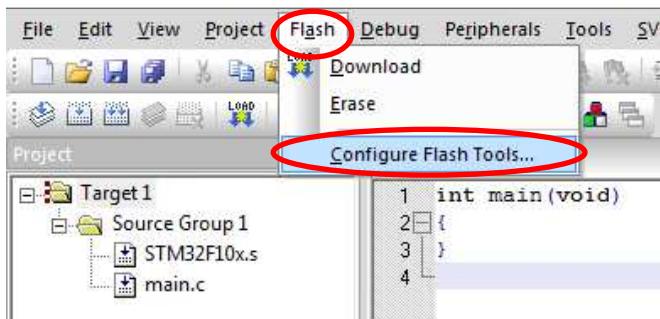
In der Mitte Unten befindet sich der Button „Choose...“

Mit einem Klick auf den Button öffnet sich folgendes Fenster:

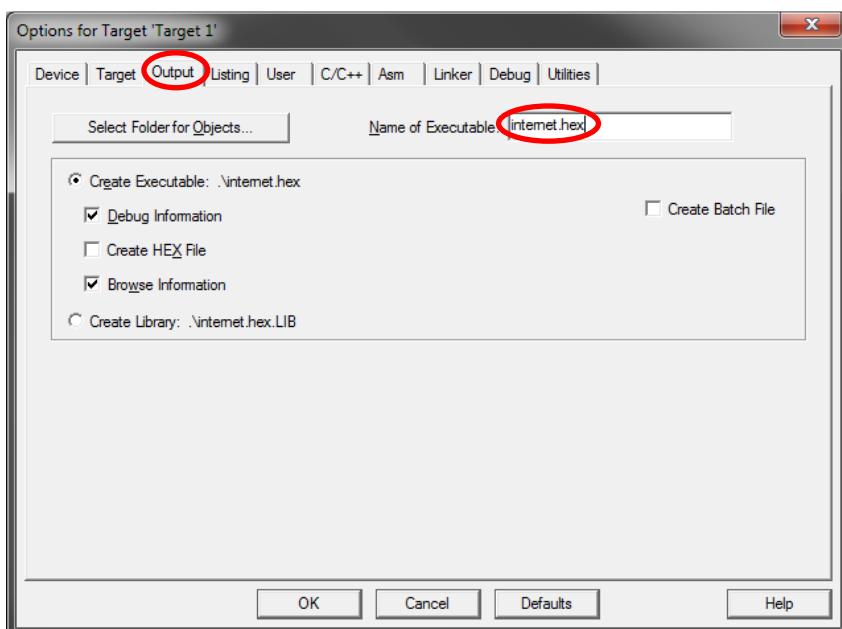
3.8 Hex-Datei flashen

Um eine *.hex-Datei auf den STM32F103RB zu flashen, ohne ein dazugehöriges Projekt zu haben (z.B. von einem anderen Compiler oder aus dem Internet) muss folgendermaßen vorgegangen werden:

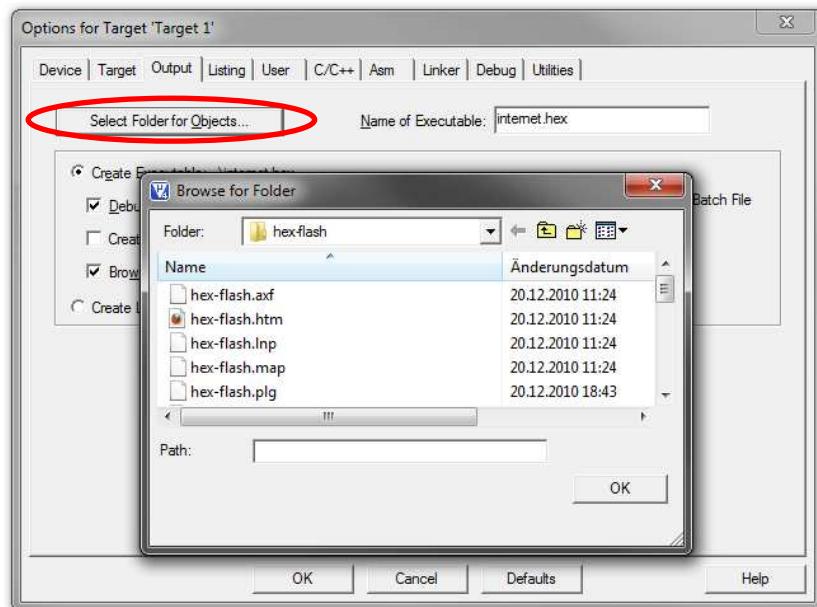
Ein neues uVision4-Projekt erstellen (siehe S.33) und im Menü „Flash“ den Punkt „Configure Flash Tools...“ auswählen.



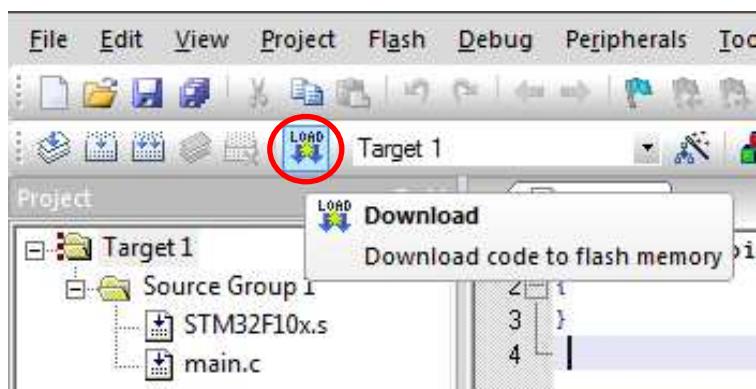
Jetzt den Menüreiter „Output“ wählen und neben „Name of Executable:“ den Namen der *.hex-Datei eintragen (inklusive Dateiendung).



Nun muss noch der Pfad der gewählten Hex-Datei durch Drücken auf „Select Folder for Objects...“ angegeben werden.



Jetzt kann die ausgewählte Hex-Datei mit Klicken auf „Download“ auf den CM3 geflasht werden.



4. FAQ

Warum startet das auf Programm erst nach dem Drücken auf die Reset-Taste?

Um das Programm automatisch nach dem Downloaden auf den CM3 zu starten, muss wie auf S.41 beschrieben vorgegangen werden.

Warum funktioniert der JTAG-Debugger nicht?

Es könnte ein falscher Debugger ausgewählt sein. Einstellungen laut S.41 auswählen.

Wie kann ich ohne der kleinen blauen U-Link Platine den Flash programmieren?

Man kann den CM3 auch über den UART (S.55) oder über USB (S.59) flashen.

Wie kann ich eine im Internet gefundene *.hex-Date für einen CM3 flashen?

Um eine *.hex-Datei ohne uVision4 Projekt auf den CM3 zu flashen muss wie auf S. 61 vorgegangen werden.

Warum zeigt mein LCD nichts an obwohl ich es programmiert habe?

Das LCD funktioniert nur wenn am DIL-Adapter die +5V-Versorgung über USB oder ein externes Netzteil angeschlossen ist.

Es könnte sein, dass eine Datenleitung vom CM3 bis zum LCD eine Unterbrechung hat. Man kann die Unterbrechungsfreiheit mithilfe des Port-Increments testen.

Warum hängt mein Programm?

Wenn das LCD im Programm verwendet wird muss es auch angesteckt sein!

Abhilfe: LCD anstecken oder auskommentieren.

Warum funktioniert die USB Verbindung nicht?

Wenn die rote LED am DIL-Adapter leuchtet wurde der Jumper rote LED/USB nicht richtig gesetzt. Siehe dazu S. 15 für genauere Informationen.

Warum flimmern einige LEDs wenn ich mit der Hand in die Nähe des LVC244 komme?

Die auf den X13 verbundenen Ports werden zur Kontrolle auf den LEDs über den LVC244 angezeigt (siehe S.22). Bei offenen Eingängen ist der Zustand nicht definiert und kann z.B. durch Näherkommen der Hand verändert werden.

Kann ich die USB Schnittstelle und den UART auch ohne Europakarte testen?

Ja! Wird am Ende des 10 Sekunden Portinkrement-Tests der DIL-Taster 1 (schräg gegenüber Reset-Taster) gedrückt, so wird ein „USB-Rechteck-Maus“-Testprogramm gestartet.
(wenn der Jumper nicht auf USB steht, leuchtet DIL_LED_1 rot)

ODER

Wird am Ende des 10 Sekunden Portinkrement-Tests der DIL-Taster 2 (mittlerer Taster) gedrückt, so wird ein (9600,N,8,1) UART Test (ch++) durchgeführt. (DIL_LED_3 leuchtet grün)

Der 1-Wire Temperaturtest funktioniert nicht! Was mache ich falsch?

Der Piezo Jumper muss offen sein und der Jumper muss auch 1-wire gestellt sein!

Warum leuchtet am DIL-Adapter immer die mittlere grüne LED?

Sie leuchtet immer, wenn der UART1 initialisiert ist und der Wakeup-Jumper gesetzt ist.