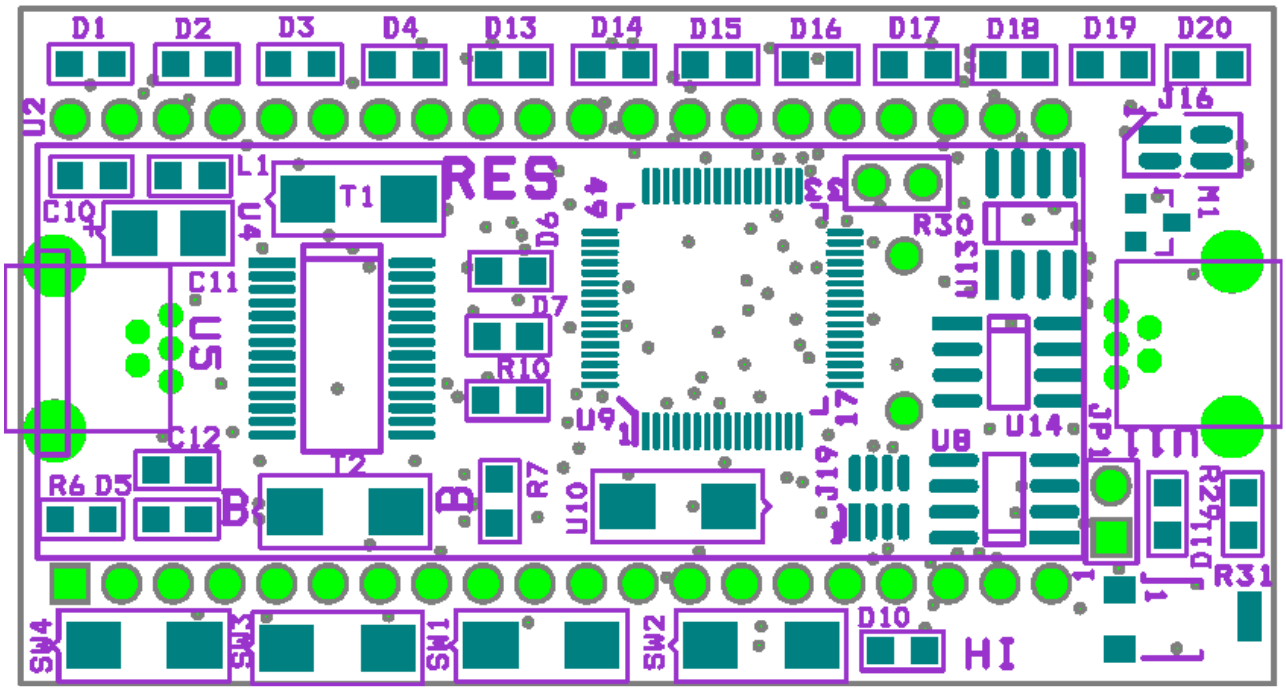
BULME GRAZ
Abteilung Elektronik / Fachtheorie & Fachpraxis
A-8051 GRAZ

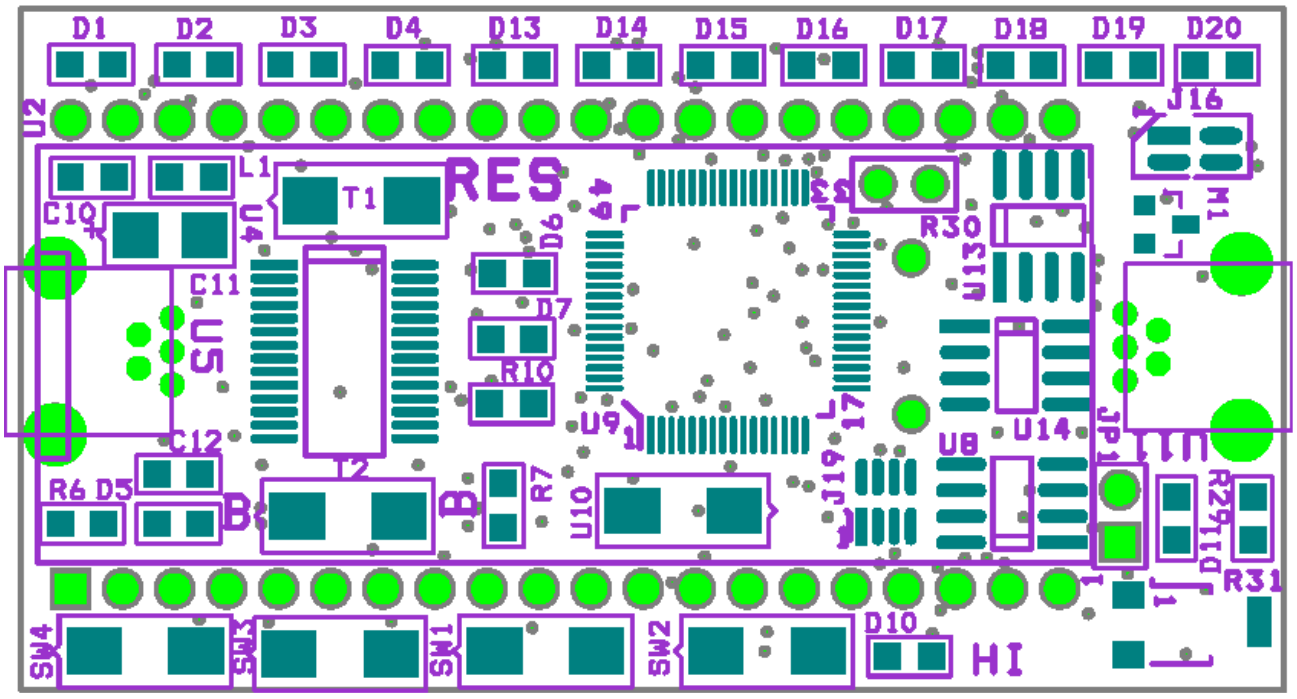
BULME CORTEX M0 BOARD 2014 „BMOB“



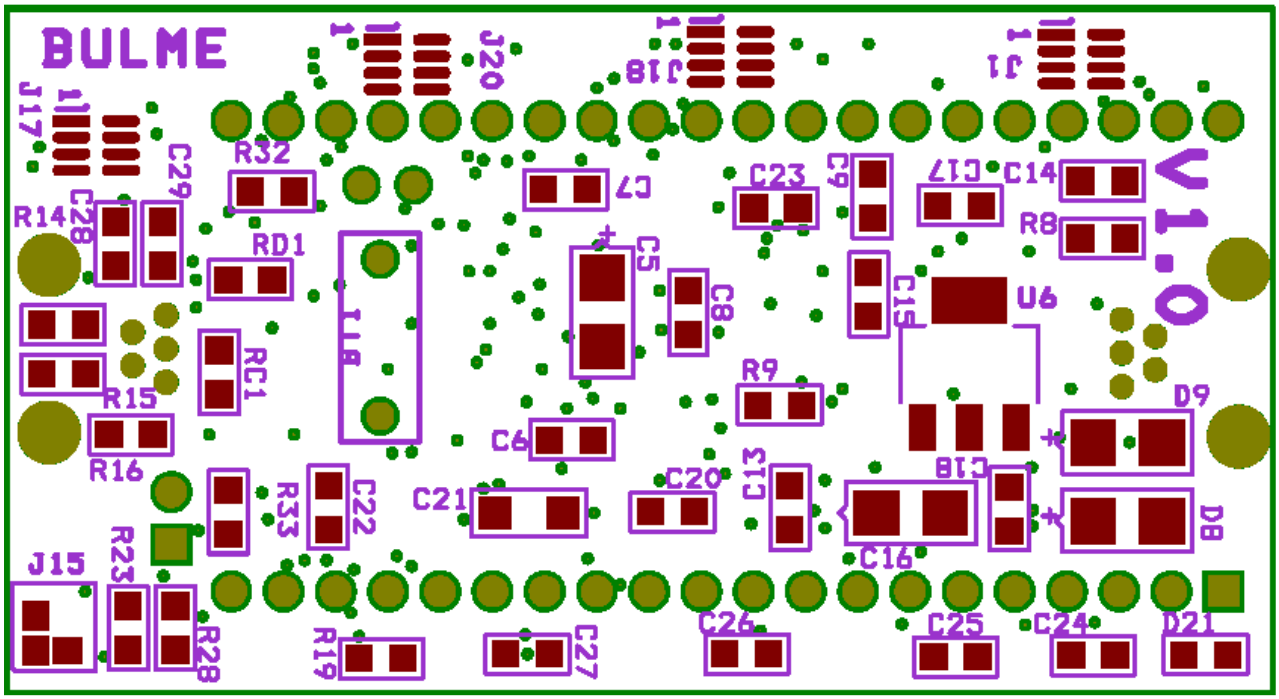
Kurzbeschreibung Version 1.0/HI,ENE Q4/13

Inhalt

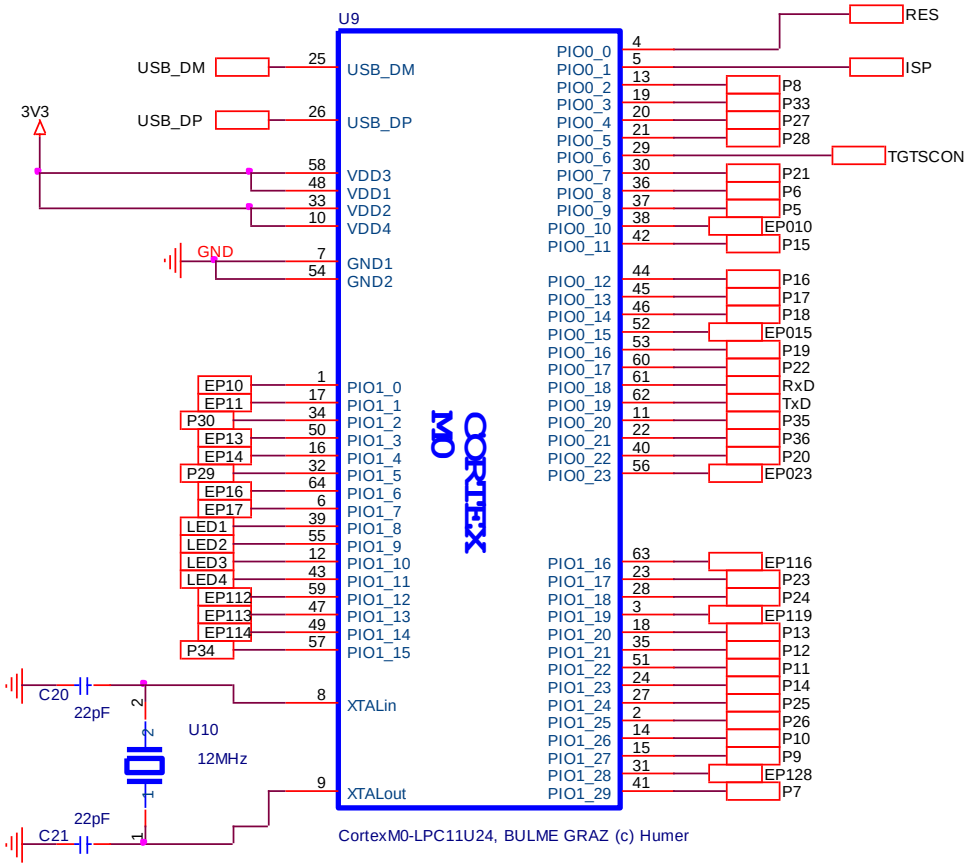
Bestückungsplan TOP



Bestückungsplan Bottom



Microcontroller CortexM0 LPC11U24



Die Anschlussbelegung des Microcontrollers ist im obigen Bild ersichtlich. Die Bezeichnung PXX ist mbed-kompatibel, die anderen Bezeichnungen sind für Erweiterungen bestimmt.

Beispiele: P25 ist kompatibel mit mbed P25

EP010 ist der Port P0.10 und wird per Software mit P0_10 angesprochen.

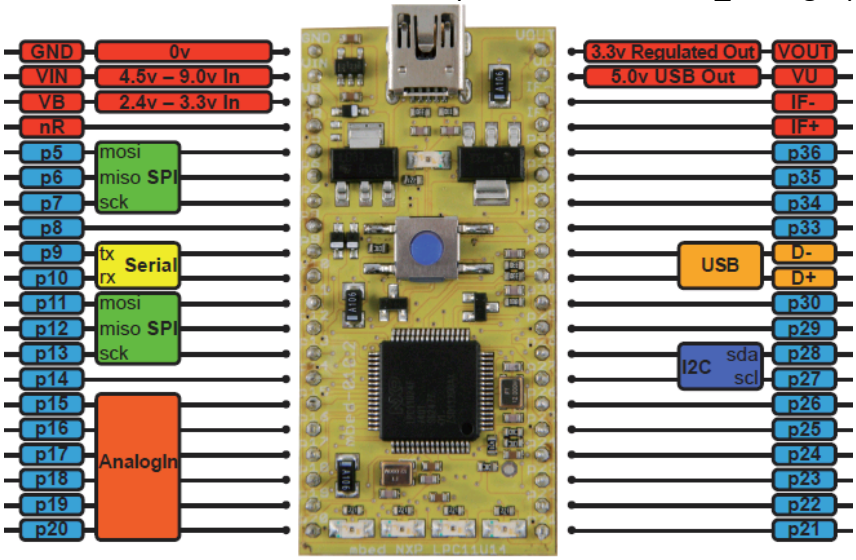


Bild: Original mbed-Pin-Belegung

Eckdaten des Microcontrollers LPC11U24

System

- ARM Cortex-M0 processor, running at frequencies of up to 50 MHz.
- ARM Cortex-M0 built-in Nested Vectored Interrupt Controller (NVIC).
- Non-Maskable Interrupt (NMI) input selectable from several input sources.
- System tick timer.

Memory

- Up to 32 kB on-chip flash program memory.
- Up to 4 kB on-chip EEPROM data memory; byte erasable and byte programmable.
- Up to 10 kB SRAM data memory.
- 16 kB boot ROM (USB or Serial).
- In-System Programming (ISP) and In-Application Programming (IAP) for flash and EEPROM via on-chip bootloader software.
- ROM-based USB drivers. Flash updates via USB supported.
- ROM-based 32-bit integer division routines.

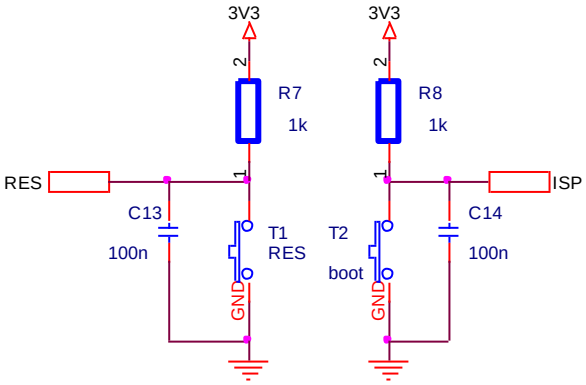
Digital peripherals

- Up to 54 General-Purpose I/O (GPIO) pins with configurable pull-up/pull-down resistors, repeater mode, and open-drain mode.
- Up to 8 GPIO pins can be selected as edge and level sensitive interrupt sources.
- Two GPIO grouped interrupt modules enable an interrupt based on a programmable pattern of input states of a group of GPIO pins.
- High-current source output driver (20 mA) on one pin.
- High-current sink driver (20 mA) on true open-drain pins.
- Four general-purpose counter/timers with a total of up to 5 capture inputs and 13 match outputs.
- Programmable Windowed WatchDog Timer (WWDT) with a dedicated, internal low-power WatchDog Oscillator (WDO).
- Analog peripherals:
- 10-bit ADC with input multiplexing among eight pins.
- Serial interfaces:
- USB 2.0 full-speed device controller.
- USART (Universal Synchronous Asynchronous Receiver/Transmitter) with fractional baud rate generation, internal FIFO, a full modem control handshake interface, and support for RS-485/9-bit mode and synchronous mode. USART supports an asynchronous smart card interface (ISO 7816-3).
- Two SSP (Synchronous Serial Port) controllers with FIFO and multi-protocol capabilities.
-2C-bus interface supporting the full I2C-bus specification and Fast-mode Plus with a data rate of up to 1 Mbit/s with multiple address recognition and monitor mode.

Clock generation

- Crystal Oscillator with an operating range of 1 MHz to 25 MHz (system oscillator).
- 12 MHz high-frequency Internal RC oscillator (IRC) that can optionally be used as a system clock.
- Internal low-power, low-frequency WatchDog Oscillator (WDO) with programmable frequency output.
- PLL allows CPU operation up to the maximum CPU rate with the system oscillator or the IRC as clock sources.
- A second, dedicated PLL is provided for USB.
- Clock output function with divider that can reflect the crystal oscillator, the main clock, the IRC, or the watchdog oscillator.
- Power control:
- Integrated PMU (Power Management Unit) to minimize power consumption during Sleep, Deep-sleep, Power-down, and Deep power-down modes.
- Power profiles residing in boot ROM provide optimized performance and minimized power consumption for any given application through one simple function call.
- Four reduced power modes: Sleep, Deep-sleep, Power-down, and Deep power-down.
- Processor wake-up from Deep-sleep and Power-down modes via reset, selectable GPIO pins, watchdog interrupt, or USB port activity.
- Processor wake-up from Deep power-down mode using one special function pin.
- Power-On Reset (POR).
- Brownout detect with four separate thresholds for interrupt and forced reset.LPC11U2X All information provided in this document is subject to legal disclaimers.
  © NXP B.V. 2013. All rights reserved.
- Product data sheet Rev. 2.1 — 17 September 2013 3 of 74 NXP Semiconductors LPC11U2x 32-bit ARM Cortex-M0 microcontroller
- Unique device serial number for identification.
- Single 3.3 V power supply (1.8 V to 3.6 V).
- Temperature range -40 °C to +85 °C.
- Available as LQFP64, LQFP48, TFBGA48, and HVQFN33 packages.

Reset und Boot

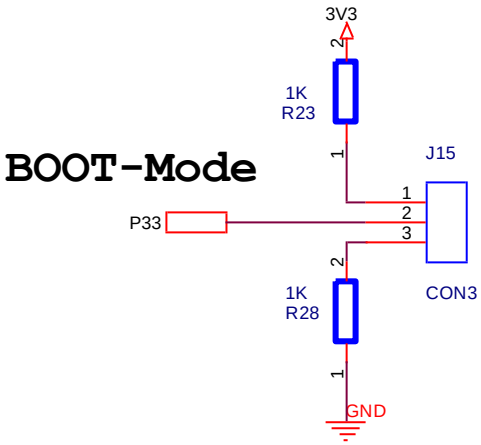


Mit der Taste T1 (RES) wir das Programm neu gestartet. Für den Boot-Modus (Programmlademodus) muss die Taste T2 (boot) gedrückt und gehalten werden, zusätzlich wird die RESET-Taste kurz gedrückt. Der Boot-Modus ist auch erkennbar durch das schwache Leuchten der LEDs („glimmen“).

Boot-Mode

Der Microcontroller LPC11U24 hat prinzipiell 2 Boot-Modi:

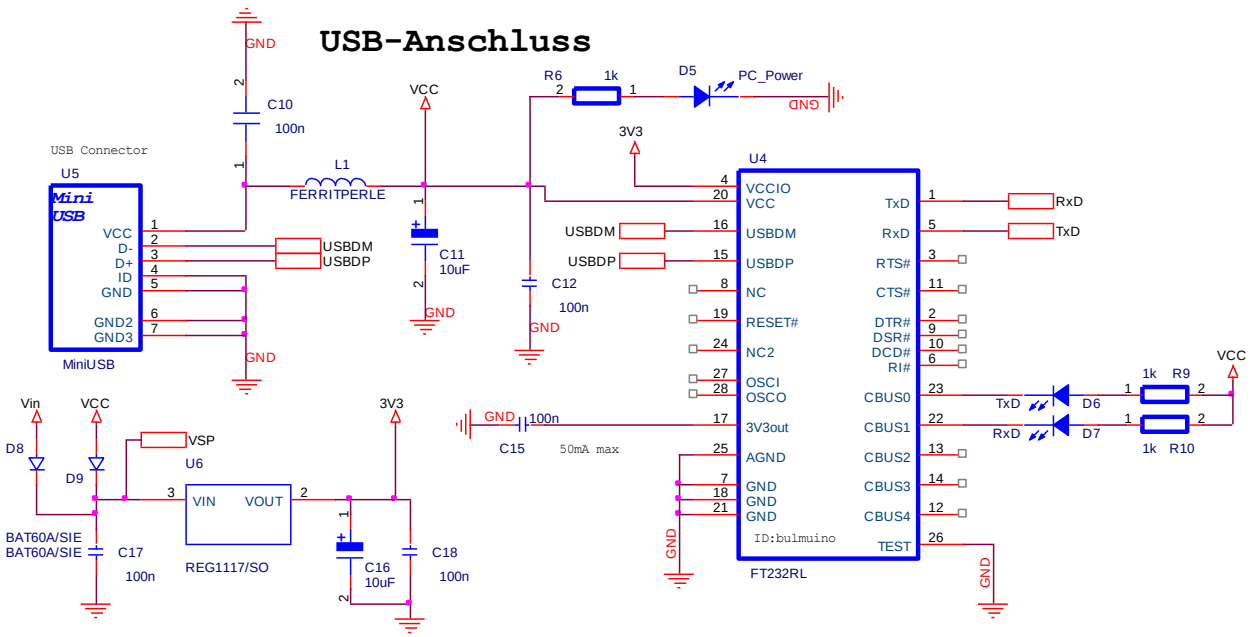
a) Booten über die USB-Schnittstelle U11 (Funktioniert wie ein USB-Stick)
b) Booten über die RS232-Schnittstelle (über den FTDI-Baustein auf die 2te USB gelegt) Hier muss für die Programmierung jedoch die Software „Flash-Magic“ von NXP verwendet werden. Diese Software ist gratis im Internet verfügbar.



Der jeweilige Boot-Modus ist über den Lötjumper J15 einstellbar.

Jumperstellung J15 1-2: Boot-Modus USB Schnittstelle (U11)
Jumperstellung J15 2-3: Boot-Modus Serielle Schnittstelle (U5)

Die serielle Schnittstelle



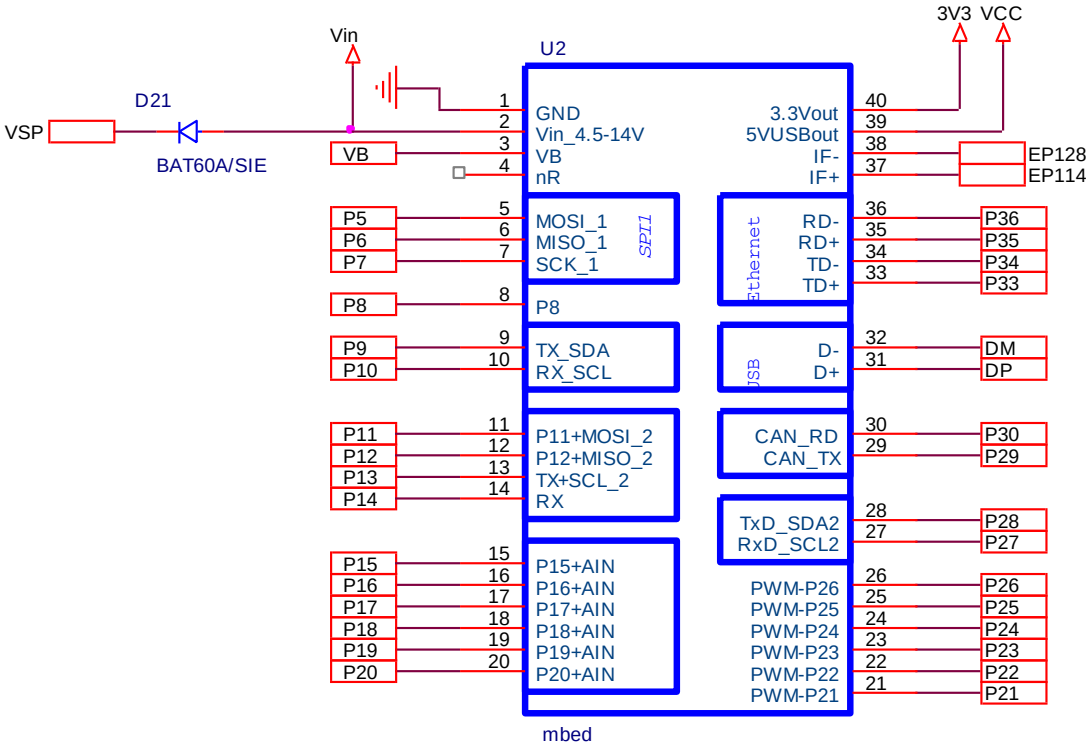
Die serielle Schnittstelle ist über den Treiberbaustein FT232RL realisiert. Als Visualisierung dienen 2 LEDs, RxD (Receive Data) und TxD (Transmitt Data). Auf dem Bild ist auch noch die Spannungsversorgung über den Längsregler REG1117-3.3V ersichtlich.

Softwareimplementierung:

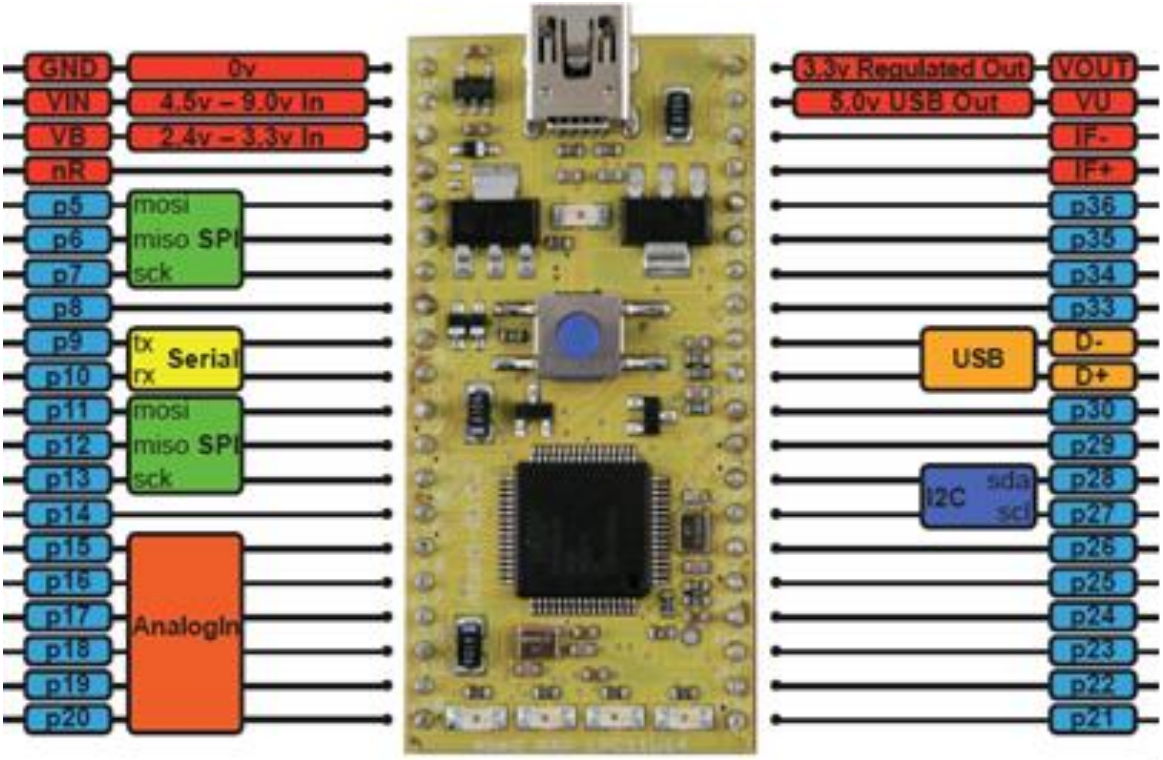
```
#include "mbed.h"

Serial pc(USBTX, USBRX);            // tx, rx, 9600baud

int main()
{
      while(1)
      {
            pc.printf("Hello World!\n");
            wait(0.1);                    // Warte 100ms
      }
}
```

Für die Visualisierung der seriellen Schnittstelle am PC ist noch eine Software am PC notwendig. Z.B. TeraTerm oder RealTerm oder Ähnliche sind kostenlos im Internet erhältlich.

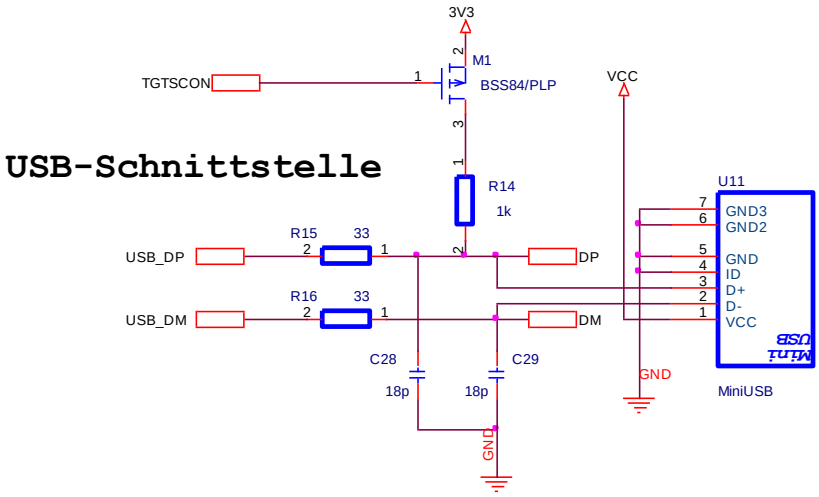
Erweiterung



Im Vergleich zum originalem mbed-Board

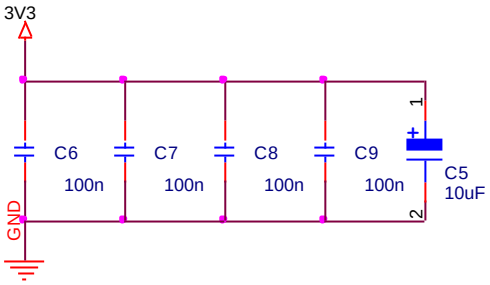


USB-Schnittstelle



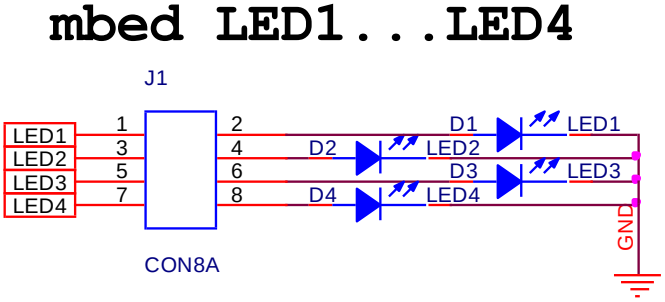
Stützkondensatoren

Für die verwendeten ICs werden folgende Kondensatoren für die Spannungsstabilisierung verwendet:



LED1..LED4 (mbed)

Für die mbed-Kompatibilität wurde für die LEDs1..4 folgende Zuteilung getroffen.



LED-Balkenanzeige

Zusätzlich zu den 4LEDs aus mbed-Welt sind noch weiter 8 LEDs implementiert.

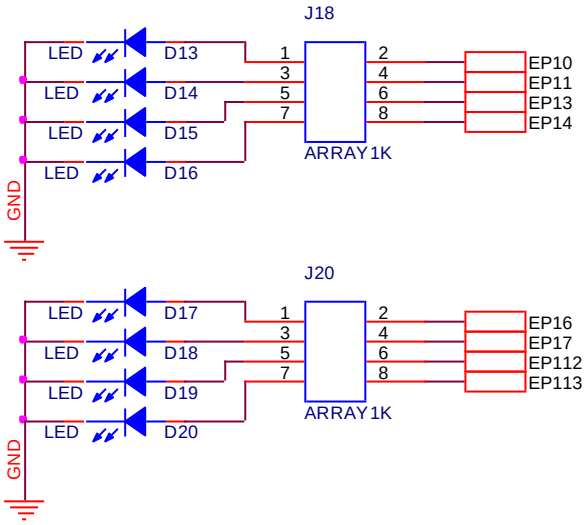


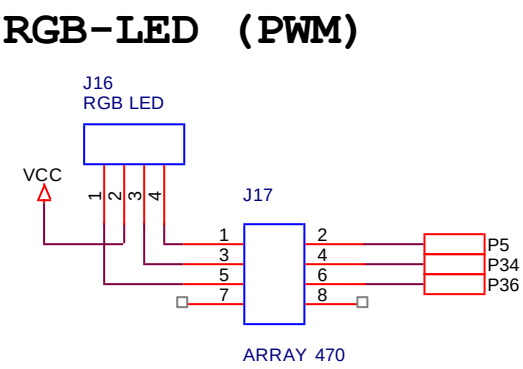
Bild: 8-fach LED

Die Leuchtstärke der LEDs wird über die verwendeten Widerstandsarrays bestimmt.

Programmbeispiel für einen 12 bit – Zähler:

```
#include "mbed.h"

// Beispiel 12 bit Zähler mit Bus-Konfiguration
// BULME Graz, Oktober 2013, Humer

BusOut myled(P1_13,P1_12,P1_7,P1_6,P1_4,P1_3,P1_1,P1_0,LED4,LED3,LED2,LED1);
//          LSB                                                          MSB

/* ************** Variablendefinition *************************** */
int i;

/* *****************   Hauptprogramm      ********************** */
int main()
{
    while(1)                                // Endlosschleife
        {
            i++;                            // Zähler
            if(i==4096) i=0;                // Rücksetzen nach 12bit
            myled=i;                        // Zählerstand Zuweisung
            wait(0.5);                      // Warte 0.5 Sekunden
        }
}
```

RGB-LED (Puls-Weitenmodulation PWM)



Die RGB-LED (rot-grün-blau) LED ist auf PWM-fähige Ports angeschlossen, dabei wurde folgende Zuordnung getroffen:

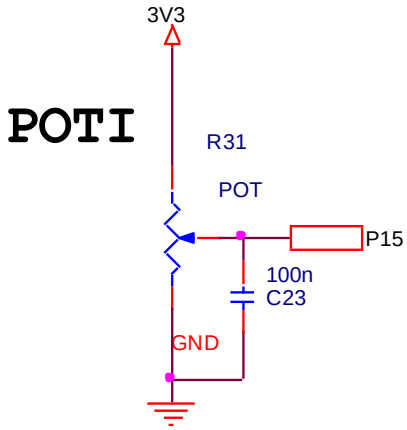
LED-rot	Anschluss	P36
LED-grün	Anschluss	P5
LED-blau	Anschluss	P34

Softwarebeispiel:

```
#include "mbed.h"

PwmOut g(p5);                       // Definition LED grün, PWM
PwmOut b(p34);                      // Definition LED blau, PWM
PwmOut r(p36);                      // Definition LED rot, PWM
AnalogIn poti(p15);                 // Definition Analogeingang, POTI

int main()
{
    r.period(0.001);          // PWM-Periode = 1ms
    while(1)
        {
// ******************** Einlesen der Spg. Am Potentiometer
// ************ Wertebereich von r: 0 <= r <= 1, float
            r=poti.read();
            g=1;                    // LED off
            b=1;                    // LED off
            wait (0.01);            // ***** Warte 10ms
        }
}
```

In diesem Programmbeispiel wird die Helligkeit der roten LED über die Einstellung am Potentiometer gesteuert.
Eingelesen wird ein Float-Wert (Wertebereich 0..1).
Die Übergabe an die PWM-Einheit erfolgt ebenfalls über eine float-Variable (0..1), wobei die Zuordnung über 1=LED off und 0 =LED 100% hell definiert ist.

Potentiometer



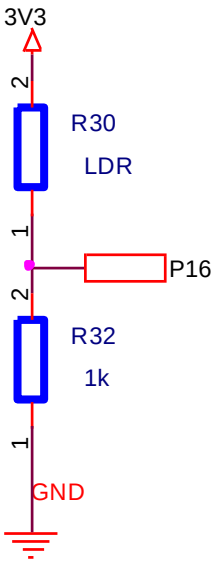
Das Potentiometer ist am Analog-Eingang P15 angeschlossen. Die Initialisierung erfolgt über:

```
AnalogIn  <variable>(p15);
```

Das Ergebnis kann als float-Wert oder als 16bit Integer-Wert erfolgen:

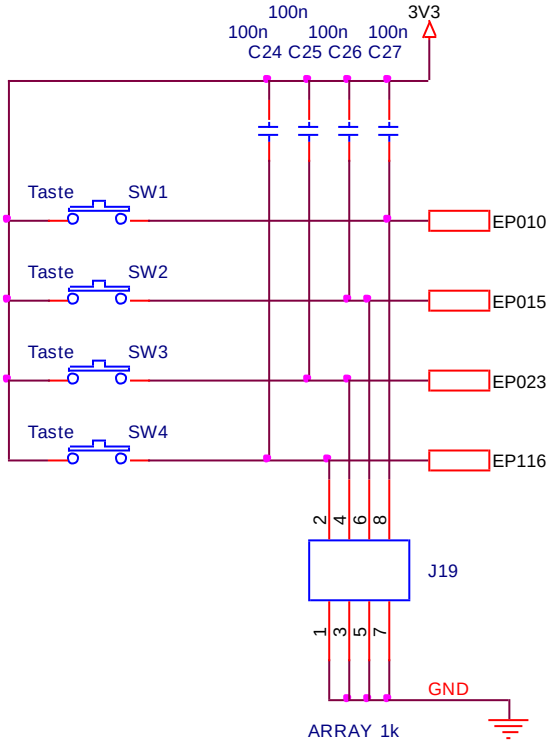
```
<varibable>.read();       // Einlesen als Float-Wert
<variable>.read_u16();  // Einlesen als 16bit Wert  0x0000 bis 0xFFFF
```

LDR



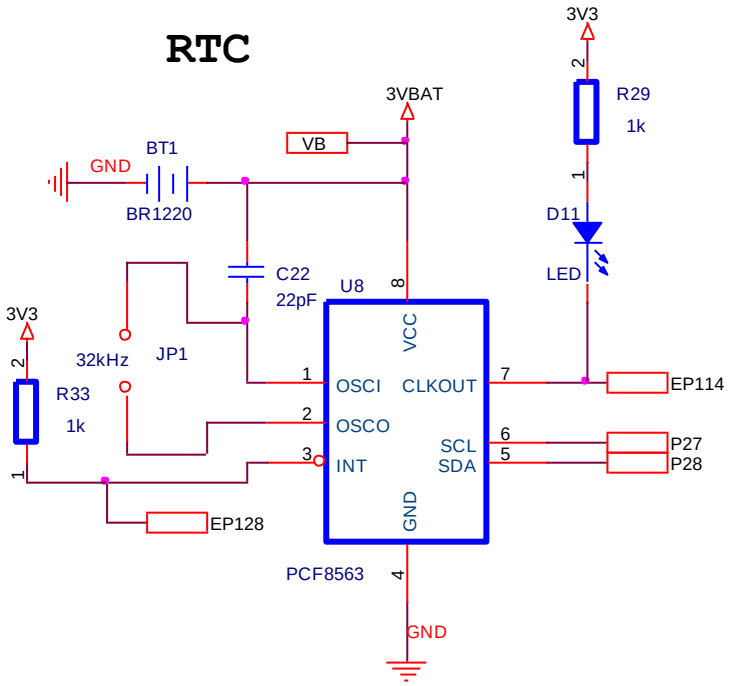
Der Helligkeitssensor (LDR) ist am Port P16 (AnalogIn) angeschlossen.

Tasten (4-fach)



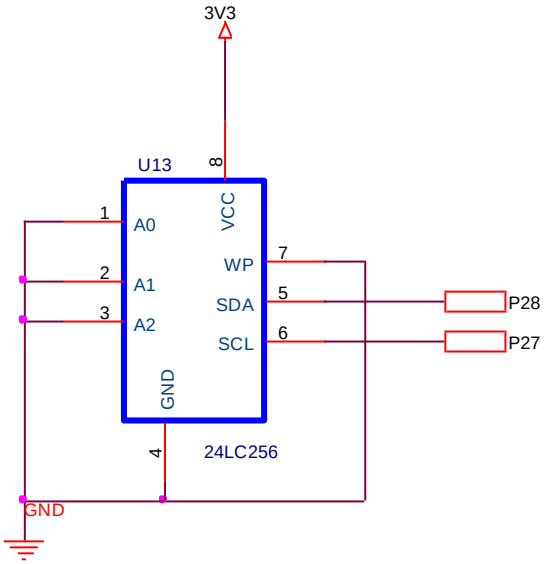
Für einfache Programmbeispiele sind auf dem Board 4 Tasten implementiert. Diese Tasten sind entweder im Polling oder über Interrupt programmierbar.
Die Bezeichnung EP010 bedeutet Port P0.10 und wird per Software über P0_10 angesprochen.

I2C RTC



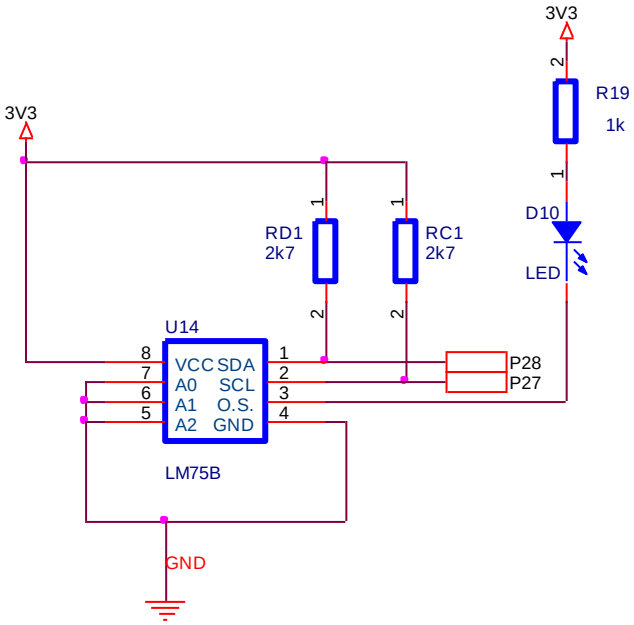
Für Programmierübungen mit der I2C-Schnittstelle ist eine Echtzeituhr (RTC) implementiert. Das Datenblatt für den Baustein PCF8563 ist im Netz kostenlos verfügbar. Es wurden alle Funktionen des Bausteins eingebunden. Die Interrupt-Funktion (z.B. Alarm) ist über den Port P1.28 und das programmierbare CLKOUT-Signal ist über Port P1.14 angeschlossen.
Eine zusätzliche Lithiumbatterie ermöglicht eine Versorgung der Echtzeituhr auch ohne Spannungsversorgung.

I2C-EEPROM



Für Programmbeispiele mit Permanentspeicher 32kB steht dieser Baustein zur Verfügung.

Digitaler Temperatursensor LM75B



Stellvertretend für viele digitale Sensoren ist auf dem Board ein Temperatursensor LM75B integriert. Die digitale Schnittstelle ist über I2C realisiert. Die Bibliothek ist in der mbed-Umgebung zu finden, eine entsprechende Implementierung einfach. Der Temperaturbereich beträgt dabei -25°C bis +100°C bei einer Genauigkeit von +/- 2°C . Die Auflösung ist bei diesem Baustein 9bit, das entspricht einer Temperaturauflösung von 0,5°C.
Um die Genauigkeit des Sensors zu erhöhen besteht die Möglichkeit einer Kalibrierkurve, die dann im I2C EEDATA – Baustein permanent gespeichert sein kann.

Stückliste

				Graz, am 20.10.13	
	Abteilung Elektronik CortexM0 Board				
Stück	**St/Pkg**	**Bezeichnung**	**Bestellnr.**	**EPreis**	**Gesamt**
3	10	Tantalkondensator 22uF/10V ESR 0.9Ohm	406-9918	0,2	€ 0,60
2	5	Schottky-Diode BAT60	738-4778P	0,078	€ 0,16
2	5	USB Mini Buchse PCB	515-2011	1,27	€ 2,54
1	5	Spannungsregler reg1117-5V 0,8A	661-6193	1,9	€ 1,90
1	1	FT232RL	406-580P	3,82	€ 3,82
1		Ferritperle	724-1475	0,068	€ 0,07
6		Tasten	756-1678P	0,47	€ 2,82
1		BSS84	436-8091	0,08	€ 0,08
5		Widerstandsarray 1k Bauform 1206	435-1807	0,02	€ 0,10
1		RGB LED, China, ebay etc.		0,5	€ 0,50
1		LPC11U24/64 PIN	755-9931P	1,91	€ 1,91
1		Quarz 12MHz SMD	753-7045P	0,56	€ 0,56
5	Rolle	Kondensatoren 18pF Bauform 805	766-5809	0,01	€ 0,05
1		PCF8563 SOIC8	436-8300P	0,76	€ 0,76
1		24LC256 SOIC8	454-331P	0,59	€ 0,59
1		LM75B Temperatursensor SOIC8	504-5632P	1,09	€ 1,09
2	Rolle	Widerstände 2K7 / 805 Rollenware	732-6242	0,01	€ 0,02
10	Rolle	Widerstände 1K0 / 805 Rollenware	698-0075	0,01	€ 0,10
14	Rolle	Kondnesatoren 100nF / 805	766-6177	0,01	€ 0,14
1		SMD Potentiometer	486-7526P	0,74	€ 0,74
1		LDR Widerstand, Neuhold, ebay etc.		0,2	€ 0,20
2	Rolle	Widerstand 33Ohm 0805	697-9952	0,01	€ 0,02
1		Platine, England, China		1	€ 1,00
1		32kHz Uhrenquarz, Neuhold		0,5	€ 0,50
					€ 20,26
				20%	€ 4,05
			GESAMT		€ 24,32
		Lieferantennummern RS-Components			

Software – Beispiele

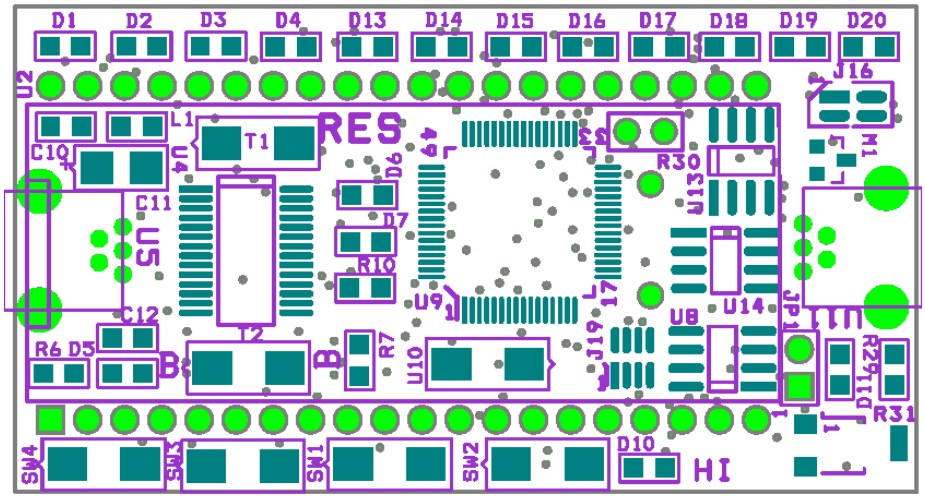
Digital-Out Bitmuster

Die vier Ausgänge (LED1…LED4) werden hier zeitgesteuert mit verschiedenen Bitmustern belegt. Die Funktion wait() enthält einen Zahlenwert, der für die Sekunden steht.

```
#include "mbed.h"

DigitalOut myled1(LED1);
DigitalOut myled2(LED2);
DigitalOut myled3(LED3);
DigitalOut myled4(LED4);

int main() {
    while(1) {
        myled1=1;
        wait(1);
        myled1=0;
        myled2=1;
        wait(1);
        myled2=0;
        myled3=1;
        wait(1);
        myled3=0;
        myled4=1;

        wait(1);

        myled4=0;
        myled3=1;
        wait(1);
        myled3=0;
        myled2=1;
        wait(1);
        myled2=0;
        myled1=1;
        wait(1);
    }
}
```

DigitalOut 12bit Zähler (BusOut)

Zur Visualisierung des 12 bit Zählers werden die 12 LEDs am M0 Board genutzt. Zu diesem Zweck werden die LEDs als Bus zusammengefasst.



Die Diode LED1 (MSB) wird über die Diode D1, und das LSB wird über die Diode D20 (=Port P1.13) abgebildet.

```
#include "mbed.h"

// Beispiel 12 bit Zähler mit Bus-Konfiguration
// BULME Graz, Oktober 2013, Humer

BusOut myled(P1_13,P1_12,P1_7,P1_6,P1_4,P1_3,P1_1,P1_0,LED4,LED3,LED2,LED1);
//    LSB (D20) .....                                          MSB (D1)

/* ***************** Variablendefinition ************************** */
int i;

/* ***************    Hauptprogramm      *************************** */

int main()
{
    while(1)                                    // Endlosschleife
    {
       i++;                                     // Zähler
       if(i==4096) i=0;                         // Rücksetzen nach 12bit
       myled=i;                                 // Zählerstand Zuweisung
       wait(0.5);                               // Warte 0.5 Sekunden
    }
}
```

Lauflicht „KIT“

In diesem Beispiel ist ein Lauflicht „hin und her“ (wie in der Fernsehserie „KIT“) realisiert. Beginnend mit der Leuchtdiode D20 werden die einzelnen LEDs nach jeweils 100ms einzeln eingeschaltet, bis die LED1 (Leuchtdiode D1) erreicht ist, dann beginnt das Spiel von der linken Seite.

```
#include "mbed.h"

// Beispiel 12 bit Lauflicht "KIT"
// BULME Graz, Oktober 2013, Humer

BusOut myled(P1_13,P1_12,P1_7,P1_6,P1_4,P1_3,P1_1,P1_0,LED4,LED3,LED2,LED1);
//    LSB (D20) ..........................................................     MSB (D1)

int i;

int main()
{
    while(1)
        {
            myled=0x0001;                 // = 0000 0000 0000 0001
            wait(0.1);
            for (i=1;i<12;i++)            // Schleife i=1 bis i=11
            {
                myled=myled<<1; // bitweises Verschieben um 1 n. links
                wait(0.1);
            }
            myled=0x0800;                 // = 0000 1000 0000 0000
            for (i=1;i<12;i++)            // Schleife i=1 bis i=11

            {
                myled=myled>>1;    // bitweises Verschieben nach rechts
                wait(0.1);
            }


        }
}
```

DigitalIn

```
#include "mbed.h"

DigitalIn enable(P0_10);            // Taste sw2
DigitalOut led(LED1);

int main()
{
    while(1)
    {
        if(enable)
        {
            led = !led;             // Signal an der LED invertieren
        }
        wait(0.25);                 // warte 0,25 Sekunden
    }
}
```

DigitalIn – Interrupt

```
#include "mbed.h"
            /* ************** Definitionen ****************** */
InterruptIn sw2(P0_10);
InterruptIn sw1(P0_15);
InterruptIn sw3(P0_23);
InterruptIn sw4(P1_16);
DigitalOut l1(LED1);
DigitalOut l2(LED2);
DigitalOut l3(LED3);
DigitalOut l4(LED4);

DigitalOut secled(P1_12);                    // LED für Sekundentag

void push1()                    // Interruptfunktion push1
{
    l4 = !l4;
}
void push2()                    // Interruptfunktion push2
{
    l3 = !l3;
}
void push3()                    // Interruptfunktion push3
{
    l2 = !l2;
}
void push4()                    // Interruptfunktion push4
{
    l1 = !l1;
}

int main()
{
    sw2.rise(&push1);   // attach the address of the push function to the rising edge
    sw1.rise(&push2);
    sw3.rise(&push3);
    sw4.rise(&push4);

    while(1)                                // Endlosschleife
   {
        secled = !secled;                   // Sekundenanzeige
        wait(0.5);
   }
}
```

Serielle Schnittstelle (USART)

```
#include "mbed.h"

Serial pc(USBTX, USBRX);                    // tx, rx, 9600 baud


int main()
{
    while(1)
    {
        pc.printf("Hello World!\n");
         wait(1);
    }
}
```

Für die Visualisierung am PC muss das Kabel auf die USB-UART Schnittstelle umgesteckt werden. Weiters muss der Treiber für den FT232RL Baustein installiert werden.
Für die Visualisierung selbst ist dann noch ein Terminalprogramm am PC wichtig. Dieses ist im Internet kostenlos downloadbar.

AnalogIn Potentiometer

Das Potentiometer ist am Port p15 angeschlossen. Die abgegriffene Spannung hat einen Wertebereich von 0 bis 3.3 Volt. Der AnalogDigitalUmsetzer hat eine Auflösung von 10bit, genau diese werden zur Visualisierung an die LED Bank geschickt.
Weiters wir das Messergebnis alle 100ms über die serielle Schnittstelle (USB-UART) an den PC weitergeleitet.

```
#include "mbed.h"

AnalogIn poti(p15);                 // Definition POTI-Eingang

                                    // Busdefinition
BusOut myled(P1_13,P1_12,P1_7,P1_6,P1_4,P1_3,P1_1,P1_0,LED4,LED3,LED2,LED1);

Serial pc(USBTX, USBRX);            // tx, rx, 9600 baud


                                    // ***** Hauptprogramm
int main()
{
    while (1)
    {
       myled=poti.read_u16()/64;
                                // Ausgabe des Analogwertes auf 10bit
                                // Ausgabe an die serielle Schnittstelle
       pc.printf("\r%u %f\n",(poti.read_u16()/64), poti.read());
       wait(0.1);               // Warte 100ms
    }
}
```

AnalogIn LDR

```
#include "mbed.h"


AnalogIn LDR(p16);                          // Definition LDR-Eingang
                                            // Busdefinition
BusOut myled(P1_13,P1_12,P1_7,P1_6,P1_4,P1_3,P1_1,P1_0,LED4,LED3,LED2,LED1);

Serial pc(USBTX, USBRX);                    // tx, rx, 9600 baud
                                            // ***** Variablendefinition

                                            // ***** Hauptprogramm
int main()
{
    while (1)
    {
       myled=LDR.read_u16()/64;
                                // Ausgabe des Analogwertes auf 10bit
                                // Ausgabe an die serielle Schnittstelle
       pc.printf("\r%u %f\n",(LDR.read_u16()/64), LDR.read());

       wait(0.1);               // Warte 100ms
    }
}
```

Zeitticker

```
#include "mbed.h"


Ticker flipper;
DigitalOut led1(LED1);
DigitalOut led2(LED2);



void flip()                             // Funktion flip() für den Ticker
{
    led2 = !led2;
}

int main()
{
    led2 = 1;
// the address of the function to be attached (flip) and the interval (2 seconds)
    flipper.attach(&flip, 2.0);

            // spin in a main loop. flipper will interrupt it to call flip
    while(1)
    {
        led1 = !led1;
        wait(0.2);
    }
}
```

Messung einer Zeitdauer

In diesem Programmbeispiel wird die Zeitdauer der Ausgaberoutine printf() gestoppt und dann als String an den PC weitergegeben.

```
#include "mbed.h"

Serial pc(USBTX, USBRX);                    // tx, rx, 9600 baud
Timer t;

int main()
{
    t.start();
    pc.printf("Hello World!\n");
    t.stop();
    printf("Zeitverbrauch %f Sekunden\n", t.read());
}
```

I2C Temperatursensor

Es muss beachtet werden die Library „LM75B“ auch wirklich in das Projekt einzubinden.

```
#include "mbed.h"
#include "LM75B.h"

//Create an LM75B object at the default address (ADDRESS_0)
LM75B sensor(p28, p27);

int main()
{
                                                    //Try to open the LM75B
    if (sensor.open()) {
        printf("Device detected!\n\r");

        while (1) {
                                                //Print the current temperature
            printf("Temp = %.3f\n\r", (float)sensor);

                                                    //Sleep for 0.5 seconds
            wait(0.5);
        }
    } else {
        error("Device not detected!\n");
    }
}
```

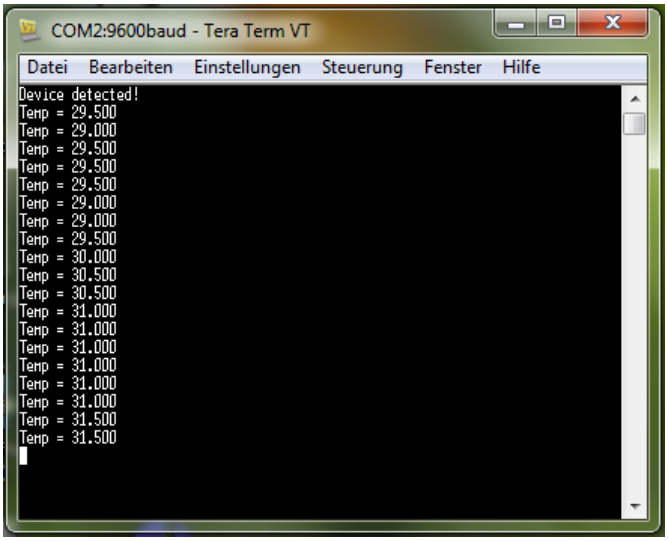


Bild: Ausgabe an einem Terminalprogramm, hier TeraTerm

Alternativ ohne Library für LM75B

```
#include "mbed.h"

                                        // Read temperature from LM75BD

I2C i2c(p28, p27);

const int addr = 0x90;

int main()
{
    char cmd[2];              // Definition Array mit 2 char Variablen
    while (1)                 // Endlosschleife
        {
        cmd[0] = 0x01;
        cmd[1] = 0x00;
        i2c.write(addr, cmd, 2);              //Schreibe am I2C Bus

        wait(0.5);

        cmd[0] = 0x00;
        i2c.write(addr, cmd, 1);              //Schreibe am I2C Bus
        i2c.read(addr, cmd, 2);               //Lese am I2C Bus

                                        // Temperaturwert errechnen
       float tmp = (float((cmd[0]<<8)|cmd[1]) / 256.0);
                                        // Ausgabe an PC
        printf("Temp = %.2f\n\r", tmp);
        }
}
```