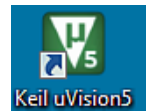
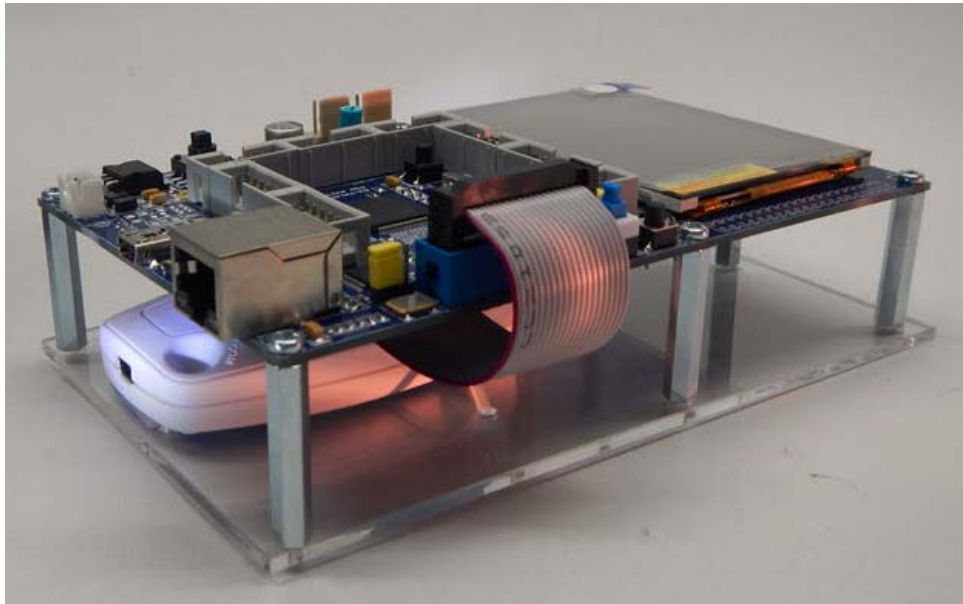


# Mikrocontroller MCB32

## Erste Schritte



# MCB32 - Embedded Programmierung Einstieg

Version: 2.00D (REVD-Grün)

Bitte beachten. Diese Unterlagen können ohne Vorankündigung jederzeit angepasst, verbessert und erweitert werden. Wir bitten Sie Wünsche und auch Fehler zu melden. ([info@mcb32.ch](mailto:info@mcb32.ch))

Version C: Print mit **transparenter** Bodenplatte. Muss mit einer speziellen LIB\_C betrieben werden.

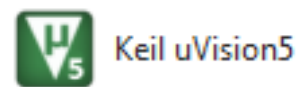
Version D: Print mit **grüner** Bodenplatte. Muss mit einer speziellen LIB\_D betrieben werden.

## Inhaltsverzeichnis

2	<b>Entwicklungsumgebung KEIL aufsetzen .....</b>	<b>3</b>
2.1	Neues Projekt einrichten mit Keil $\mu$ Vision 5 .....	3
2.2	Weitere Projekt- und Prozessor (Target)-Einstellungen .....	4
3	<b>Anhang: Umstellung von C51-Code auf ARM32-Code .....</b>	<b>8</b>
3.1	Wichtig für das Funktionieren neuer Projekte .....	8
4	<b>Anhang Touchscreen Kontrolle am <math>\mu</math>C-Board MCB32 .....</b>	<b>9</b>
5	<b>Anhang Anschlüsse am <math>\mu</math>C-Board MCB32 .....</b>	<b>10</b>
6	<b>Anhang: Referenzen .....</b>	<b>12</b>
7	<b>Anhang Literaturverzeichnis und Links .....</b>	<b>12</b>
8	<b>Anhang Wichtige Dokumente .....</b>	<b>12</b>

## 2 Entwicklungsumgebung KEIL aufsetzen

Wir arbeiten mit der IDE der Firma Keil. Damit können wir die Programme bis zu 32kB Programmcode schreiben. Das genügt für die Ausbildung.



**IDE:** Integrated Development Environment von <http://www2.keil.com/mdk5/install>

**ARM:** Advanced RISC Machines

### 2.1 Neues Projekt einrichten mit Keil $\mu$ Vision 5

#### 1. Schritt

$\mu$ Vision 5 starten:

Neues Projekt erstellen:

Start / Programme / Elektronik / KeilArm  $\mu$ Vision 5

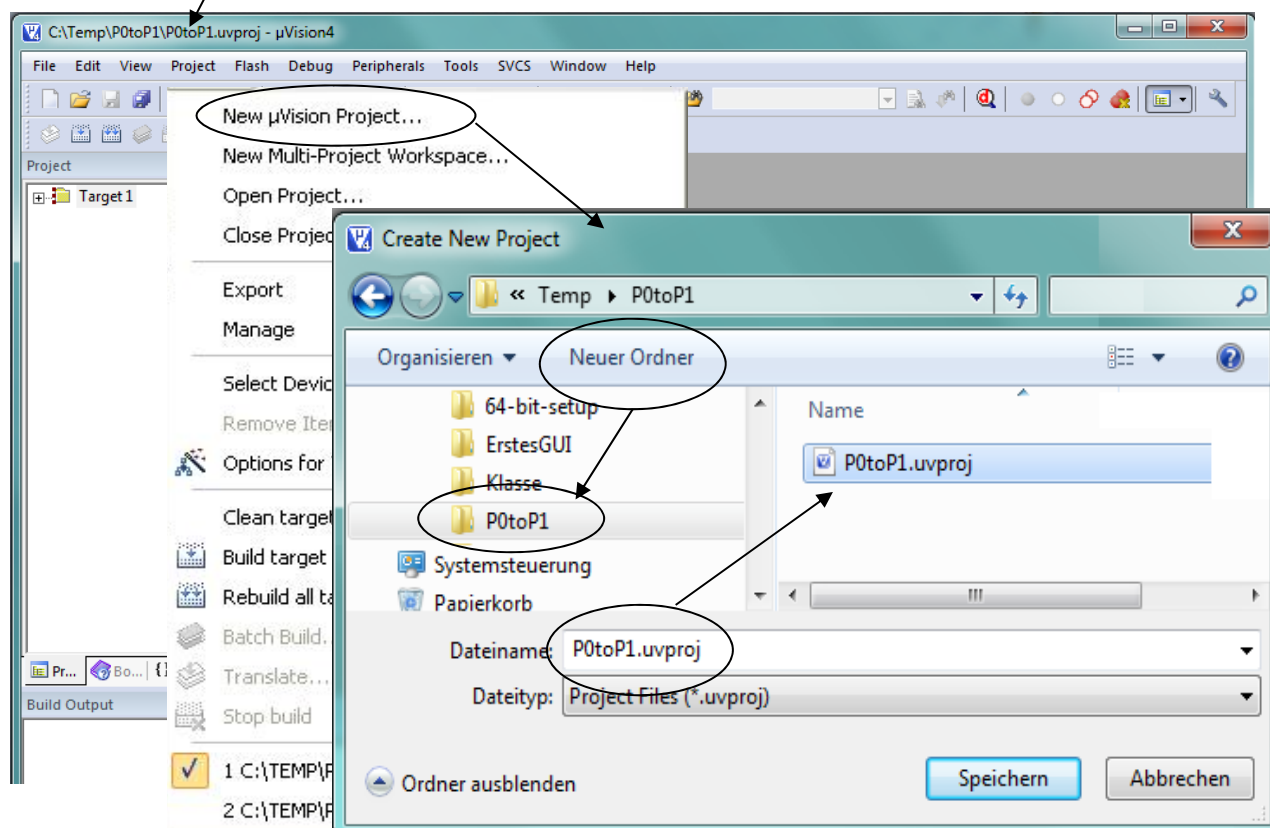
Project / New Project /

Für jedes Projekt

- einen neuen Ordner zB: P0ToP1

- dann Projektname zB: P0ToP1.uvproj

- Speichern



#### Weitere Menüpunkte der IDE welche es zu beachten gibt.

Altes Projekt öffnen: **Project** / Open Project

Projekt schliessen: **Project** / Close Project

$\mu$ Vision 4/5 verlassen: **File** / Exit

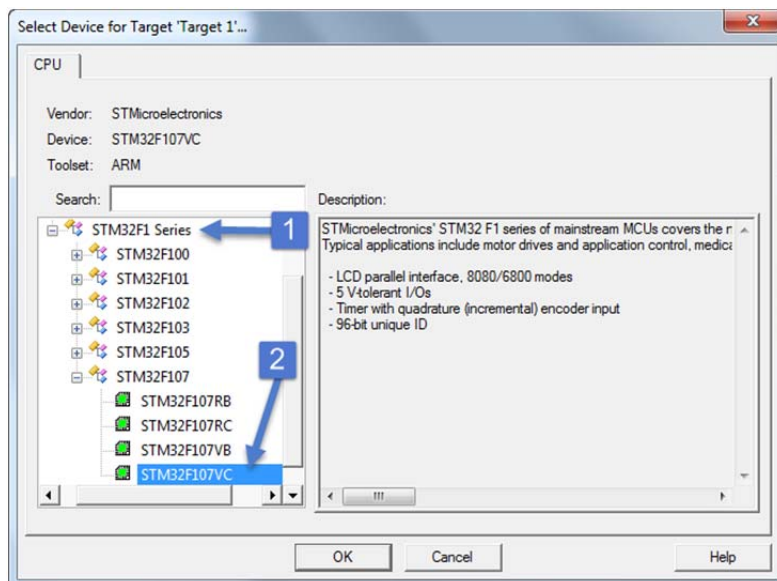
**Wichtig:** jedem Projekt müssen die beiden Files: **Touch P0P1.lib** und **TouchP0P1.h** vorhanden sein. Kopieren Sie diese Files vom Server (LIB C oder LIB D) oder von einem bekannten Ort ins Projekt.

## 2.2 Weitere Projekt- und Prozessor (Target)-Einstellungen

### 2. Schritt: Target auswählen

Nun fragt die IDE nach dem Ziel-Controller:

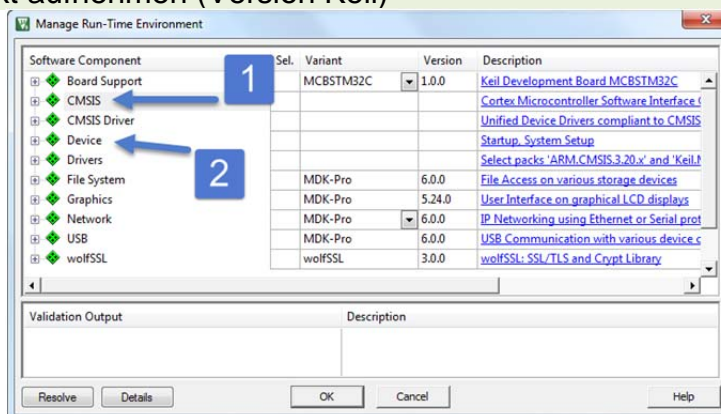
- Ziel-Controller wählen. In unserem Fall der **STM32F107VC**



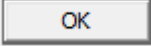
### 3. Schritt: Startup Code ins Projekt aufnehmen (Version Keil)

Nun benötigt die IDE mehr Informationen und auch „Files“ um nachher während der Kompilation alle Verknüpfungen herstellen zu können:

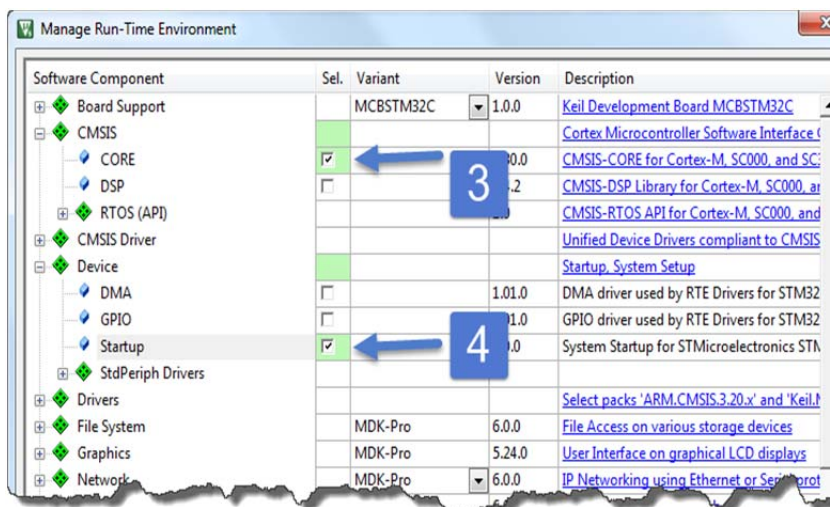
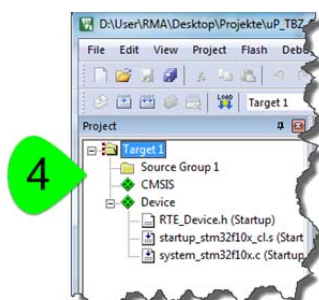
- CMSIS – Basics laden
- Device – Files laden



- CMSIS: Core auswählen
- Device – Startup auswählen

Und  klicken.

Die Projektumgebung sollte nun so aussehen:

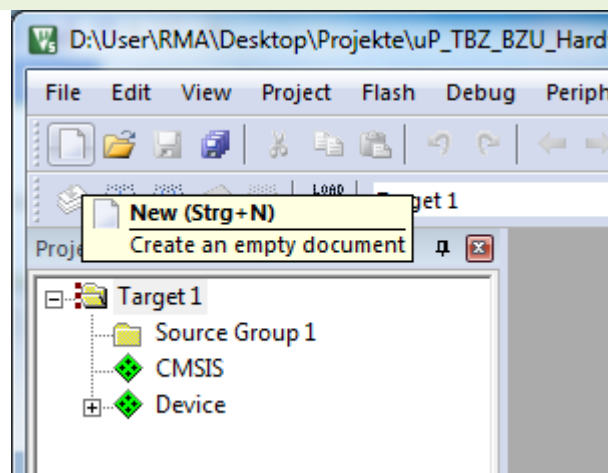


Im Prinzip ist das Projekt bezüglich der grundlegenden Einstellungen bereit. Wir müssen nun die eigentlichen Programmfiles dazufügen.

## Aufsetzen eines ersten Programmes

### 4. Schritt: Programm Files aufnehmen.

- Klicke auf NEW (oder STRG-N) und speichere das File mit dem Namen P0toP1.c ab.



### 5. Schritt: Programm schreiben.

- Schreibe nun folgenden Code.
- Speicher alles ab.



Dieses File können wir mit anderen Files nun noch dem Projekt hinzufügen.

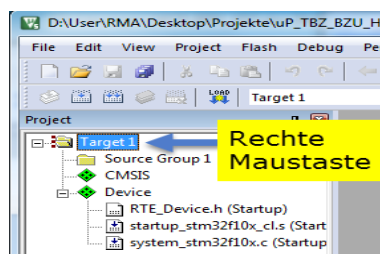
```

1 //-----
2 // MCB32 (BZU / TBZ)
3 // Autor: P0toP1.c / 2.1.14 / rw, mal / BZU, TBZ
4 // Thema: Schaltet am uC-Board MCB32 die Schalter an P0
5 // (GPIOC0..7) zu den LEDs an P1 (GPIOE8..15) durch. Mit Touchbedienung
6 //-----
7 #include <stm32f10x.h> // Mikrokontrollertyp
8 #include "TouchP0P1.h" // Library mit P0-, P1-Definition und Touch
9
10 int main(void) // Hauptprogramm
11 {
12     InitTouchP0P1("1"); // P0,P1 auf Touchscreen ON
13     while(1) // Endlosschleife
14     {
15         P1 = P0; // Portdurchschaltung
16     }
17 }

```

### 6. Schritt: Alle Programmfiles zum Projekt hinzufügen.

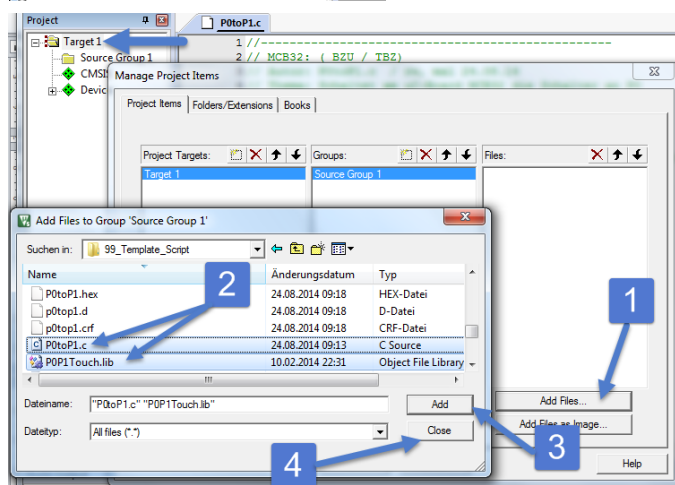
Mit der rechten Maustaste auf „Target1“ klicken.



- Nun die Schritte 1-4 abarbeiten:

Das File: P0toP1.c und die Library Touch P0P1.lib werden dem Projekt hinzugefügt.

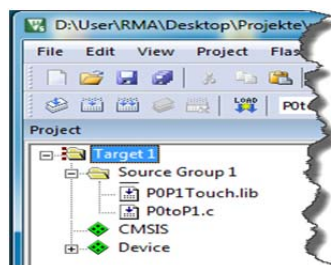
Wichtig: Die Files können mit CTRL-Click alle zusammen ausgewählt werden.





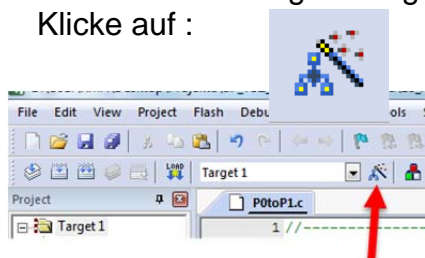
## Aufsetzen eines ersten Programmes

So sollte das Projekt nun aussehen:



### 7. Schritt: Setup vervollständigen (Debugger und JTAG)

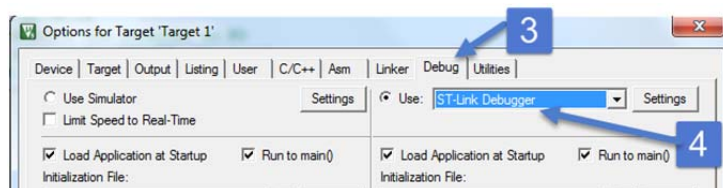
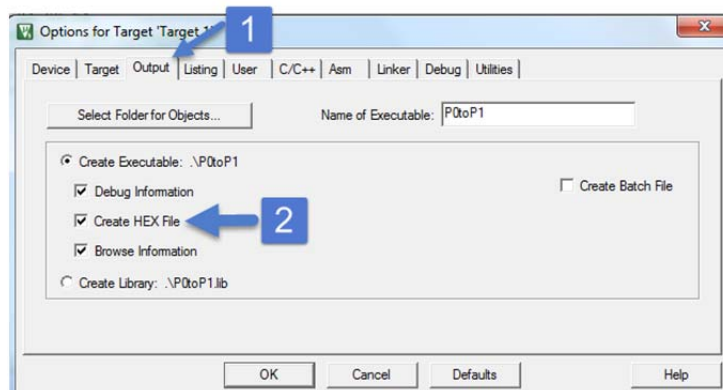
Damit das Programm nach dem Kompilieren in den ARM geladen werden kann sind weitere Einstellungen nötig. Klicke auf :



Damit werden die „Options for Target1“ ausgewählt.

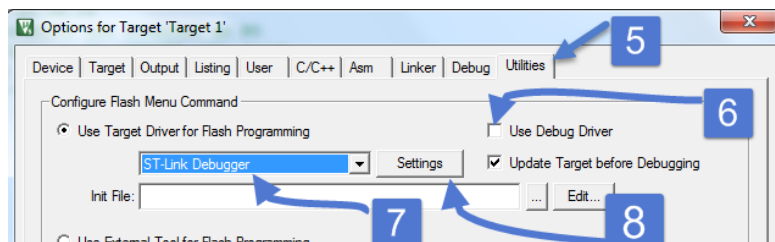
Wähle: „Debug“ und selektiere „STLink Debugger“ aus dem Menü aus.

Wähle: „Output“ und selektiere „Create HEX File“.



Wähle: „Utilities“, Klicke „Use Debug Driver“ **aus** und selektiere noch „STLink Debugger“.

Am Schluss [8] die Settings auswählen.



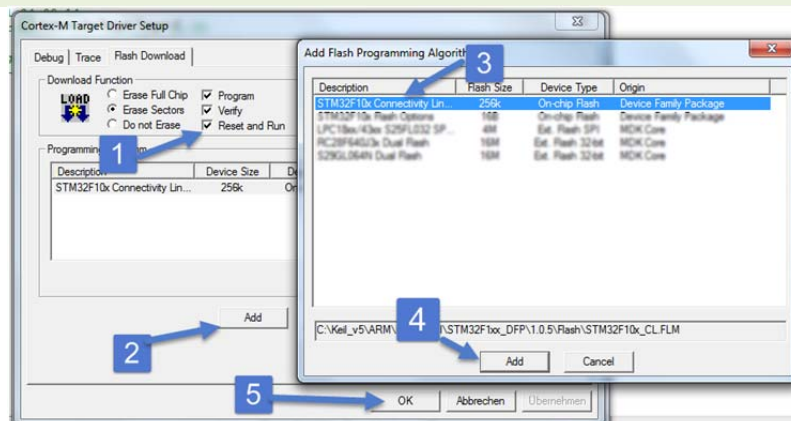
### 8. Schritt: Settings für ST-Link Download vornehmen

Bei den Settings nun „Reset and Run“ auswählen.

Mit „Add“ [2] sagen wir dem Treiber welchen Programmieralgorithmus gewählt wird. Also den für den STM32F10x [3].

Mit [4] abschliessen und mit OK [5] das Menü „Driver Setup“ schliessen.

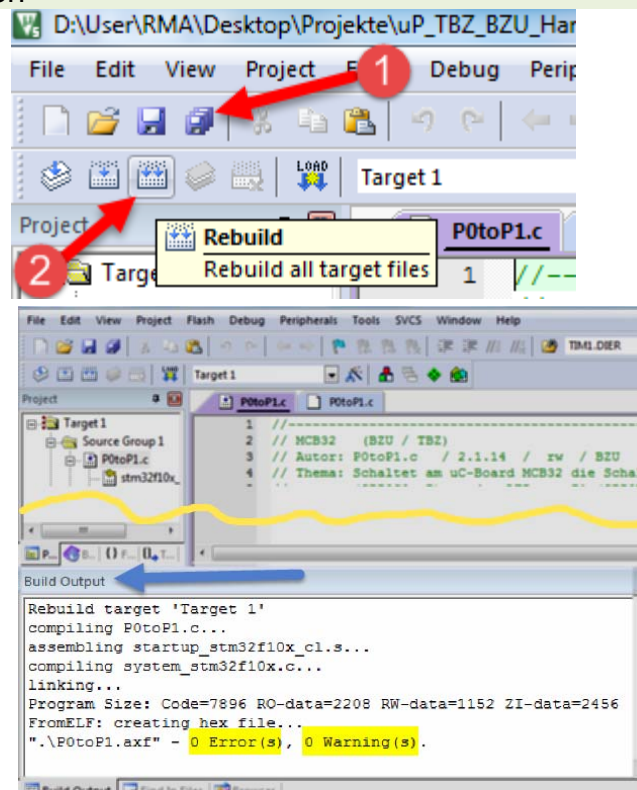
Das Fenster schliessen und fertig.



## Aufsetzen eines ersten Programmes

### 9. Schritt: Programm kompilieren

Sichere das Projekt [1] und kompiliere [2] mit dem Knopf „Rebuild all target files“



Im Fenster „Build Output“ sollte in etwa nebenstehende Meldung erscheinen.

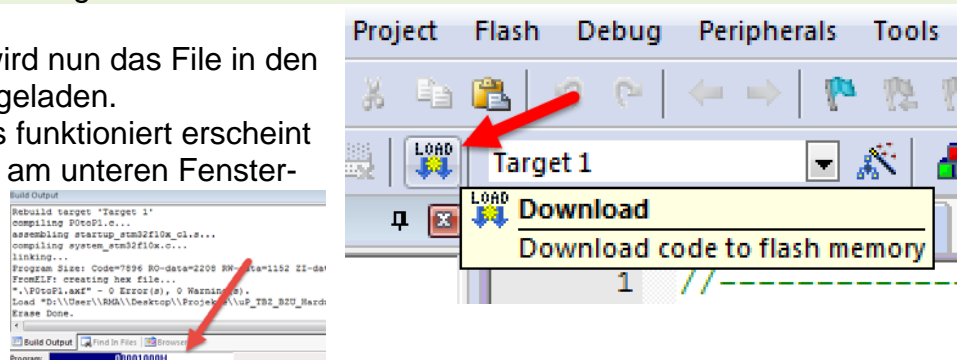
Wenn keine Fehler vorkommen kann das erzeugte HEX-File in den Prozessor geladen werden.

### 10. Schritt: Programm in den Prozessor laden.

Mit Load wird nun das File in den Kontroller geladen.

Wenn alles funktioniert erscheint ein Balken am unteren Fenster- rand.

Dieser zeigt den Download an.



### 3 Anhang: Umstellung von C51-Code auf ARM32-Code

```

/*****
* Titel:      Umstellung von C51-Code auf ARM32-Code
* Datei:      C51toARM32.c / 14.1.14 / Version 1.0
* Ersteller:   R. Weber (BSU); E. Malacarne (TBZ)
* Funktion:    Die wichtigsten Umstellungen sind in den Kommentaren dokumentiert
*****/

```

```

#include <stm32f10x.h>           // Mikrokontrollertyp
#include " TouchP0P1.h"         // P0-, P1-Definition

#define Start P0_0              // Input und Outputbits
#define Alarm P1_7              // an Ports benennen
char bTemp = 0 ;                // 'Bit'-Variablentyp char
long t;                         // Zeitvariable

```

neue #includes

Keine sfr und  
sbit mehr!

```

int main ( void )              // Hauptprog., ohne return bei Keil
{
    InitTouchP0P1 ("1");        // Touchscreen aktiv,
                                // horizontal gedreht
                                // LSB rechts

```

Main verlangt int

InitTouchP0P1 ("0"),  
wenn nur P0,P1 und  
ohne Touchscreen

```

while(1)                       // Endlos-schleife
{
    P1_0 = 0;                   // Bitverarbeitung wie bisher
    Alarm = 1;                  // Zuweisung, Invertierung,
    bTemp = ! Start;            // &, &&, |, ||, ^, !, ==, !=

    while ( P1 < 100 )           // Byteverarbeitung wie bisher
        P1 += 2;                 // Kurzformen wie bisher
    P1 = P0 & 0x0F;              // Maskierungen wie bisher

    for(t=120000; t>0; t--);     // Verzögerung 10ms
}

```

Verzögerung

vom Typ long mit  
Wert 12 / µs

#### 3.1 Wichtig für das Funktionieren neuer Projekte

**Wichtig:** Bei der Erstellung eines neuen Projektes im Schulbereich, also Vorbereitung für 8Bit-Programme „Elektroniker“ mit Port P0, P1 und Touchscreen ist folgendes zu beachten.

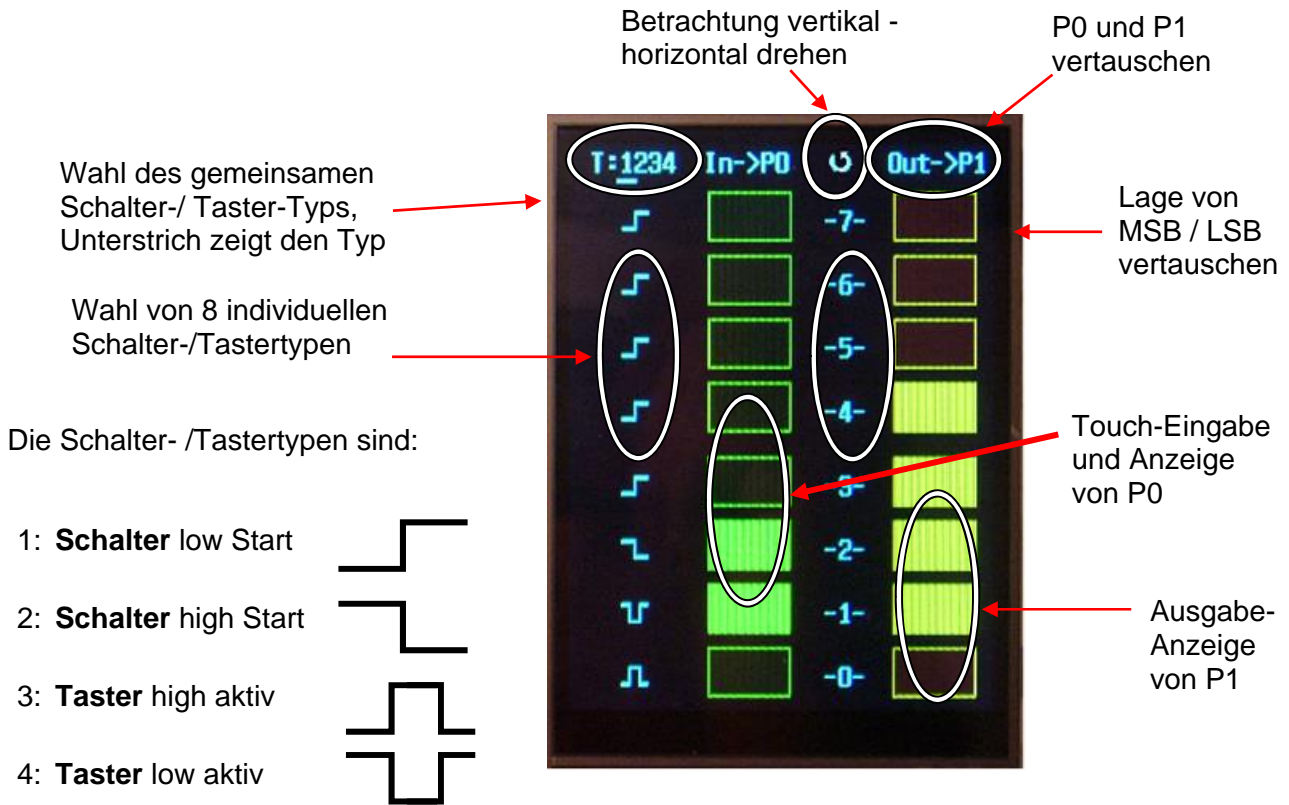
Kopieren Sie in jedes neue Projektverzeichnis diesen zwei Dateien:

- TouchP0P1.h { XE "P0P1Touch.h" } (REV C oder REV D)
- TouchP0P1.lib (REV C oder REV D) { XE "P0P1Touch.lib" }



## 4 Anhang Touchscreen Kontrolle am µC-Board MCB32

### Beschreibung der Touchscreen Oberfläche



### Touchscreen Kontrolle aus dem Quellcode

Der Projekt-Ordner muss TouchP0P1.h und TouchP0P1.lib enthalten:

Im Projekt-Manager sind zum **stm32f10x.h** die Quelldatei.c und TouchP0P1.lib aufzunehmen.

```
/* Beschreibung der 8 Bit P0- und P1-Kontrolle über den Touchscreen des MCB32
*****/
#include <stm32f10x.h> // Mikrokontrollertyp
#include " Touch P0P1.h" // P0-, P1-Definition. Angepasst für REV C oder D

void main(void) // Hauptprogramm
{
    InitTouchP0P1 ("1"); // Touchscreen aktivieren
    while(1) { } // Benutzerprogramm InitTouchP0P1("1");
}
```

1) **InitTouchP0P1 ("0");**

Der Touchscreen bleibt ausgeschaltet

P0 ist als Input, P1 als Output konfiguriert

2) **InitTouchP0P1 ("1") .. ("1 r m p");**

Der Touchscreen wird aktiviert und konfiguriert, einfachste Konfiguration mit InitTouchP0P1 ("1").

1...4: Gemeinsamer Schalter-/Tastertyp

p: P0 aussen, sonst mittig.

m: MSB oben/rechts, sonst unten/links.

r: Rotiert horizontal, sonst vertikal.

## 5 Anhang Anschlüsse am $\mu$ C-Board MCB32

### Entwicklungsanschlüsse

Fremdspeisung  
genau 5V! oder ...

USB Speisung (**roter Stecker**)

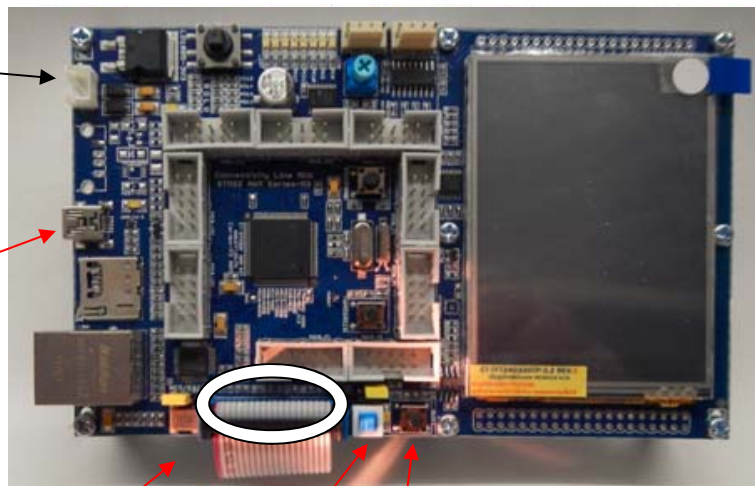
USB - ST-Link (**schwar-  
zer USB Stecker**)  
Zielsystemdebugging,  
Boardspeisung  
wie oben

Bootloadschalter:

Ungedrückt lassen **LED OFF**

**Reset**-Taster:

Programmrestart



### Digitale Ein- und Ausgaben am $\mu$ C-Board MCB32

P1: 8x onboard LEDs  
parallel zu 8x extern LEDs

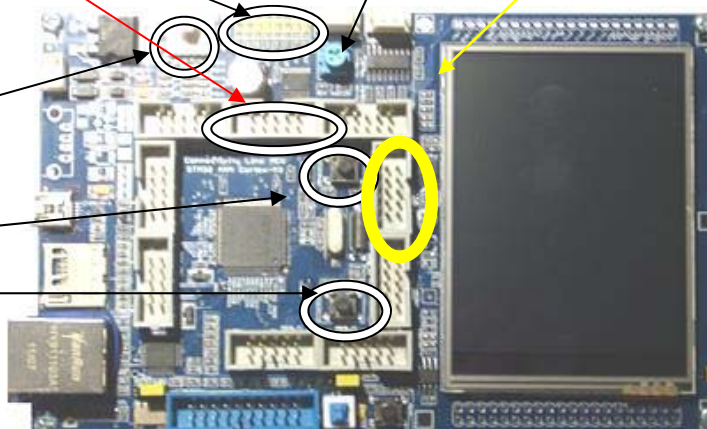
**Pot** auf Anschlag Uhrzeigersinn  
drehen, da gleichzeitig P0\_4

P0: 8 externe Schalter

ControlStick

Button 1

Button 0



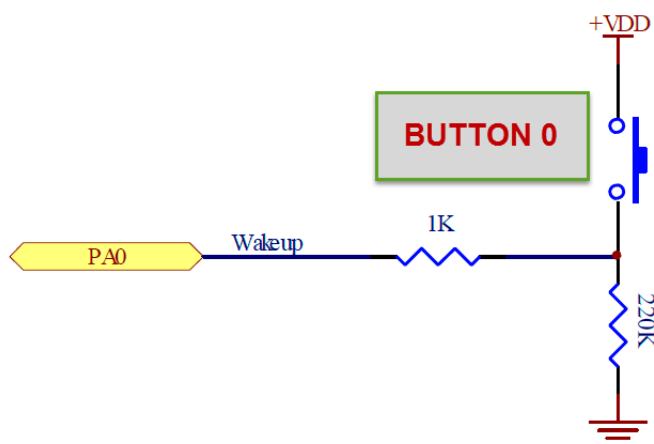
// In TouchP0P1.h definierte Pin-Bezeichnungen PA\_0 .. PD\_11, ohne Bezeichner wie Button .. !

```
char Button0      = PA_0;           // Bitwert 1/0, aktiv low, prellt wenig
char Button1      = PC_13;

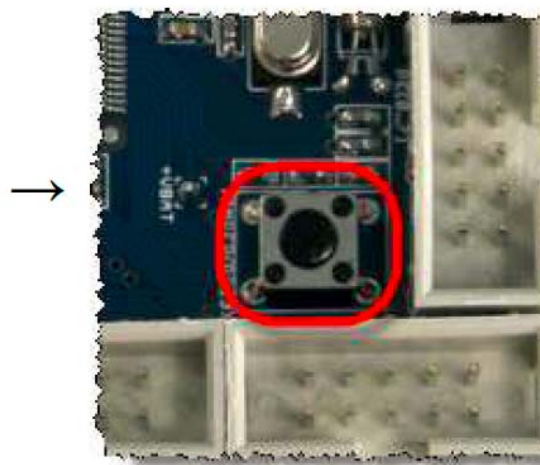
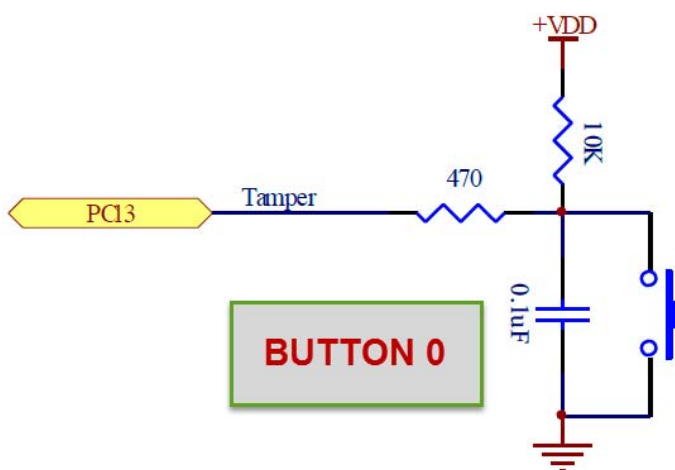
char Stick        = PD_High;        // als Byte 0xF8 open, aktiv low, alle entprellt
char StickSelect  = PD_15;          // Bitwert 1/0; Bytewert 0x80
char StickDown    = PD_14;          //          1/0;          0x40
char StickLeft    = PD_13;          //          1/0;          0x20
char StickUp      = PD_12;          //          1/0;          0x10
char StickRight   = PD_11;          //          1/0;          0x08
```

## Aufsetzen eines ersten Programmes

Button 0 (PA\_0)



Button 1 (PC\_13)



## 6 Anhang: Referenzen

- [1] Robert Weber / Berufsschulzentrum Uster / Projektvorlagen (div)  
*Danke Robi für die tollen Vorlagen.*
- [2] E. Schellenberg, E. Frei / TBZ. Programmieren im Fach HST
- [3] Unterlagen MCB32 / Cityline / E. Malacarne
- [4] Link zu Wikipedia. [http://de.wikipedia.org/wiki/ARM\\_Cortex-M3](http://de.wikipedia.org/wiki/ARM_Cortex-M3) (1.9.14)
- [5] D. Schoch ; TBZ ; « C-Grundlagen.doc 29.5.2001 » ; Neu erstellt
- [6] E. Frei ; TBZ ; Erweiterungen in [2] « C-Grundlagen.doc 29.5.2001 »
- [7] Dirk Louis ; C++ « Programmieren mit Beispielen », MT-Verlag
- [8] Link zu Wikipedia: <http://de.wikipedia.org/wiki/Einerkomplement>

## 7 Anhang Literaturverzeichnis und Links

- [1] R. Jesse, Arm Cortex M3 Mikrocontroller. Einstieg und Praxis, 1 Hrsg., www.mitp.de, Hrsg., Heidelberg: Hütigh Jehle Rehm GmbH, 2014.
- [2] J. Yiu, The definitive Guide to ARM Cortex-M3 and M4 Processors, 3 Hrsg., Bd. 1, Elsevier, Hrsg., Oxford: Elsevier, 2014.

Weblinks <http://www.mikrocontroller.net/topic/158108> // für fehlendes volatile statement

## 8 Anhang Wichtige Dokumente

Die folgende Liste zeigt auf die wichtigsten Dokumente welche im WEB zu finden sind. Beim Suchen lassen sich noch viele nützliche Links finden.

- Datenblatt{ XE "Datenblatt" } ([STM32F107VC](#)) Beschreibung des konkreten Chips für Pinbelegung etc.
- Reference Manual ([STM32F107VC](#)) (>1000Seiten in Englisch)  
Ausführliche Beschreibung der Module einer Familie. Unter Umständen sind nicht alle Module im eingesetzten Chip vorhanden – siehe Datenblatt.
- Programming Manual{ XE "Programming Manual" } ([Cortex-M3](#))  
Enthält beispielsweise Informationen zum Interrupt Controller (NVIC{ XE "NVIC" }).
- Standard Peripheral Library ([STM32F10x](#))  
Im Gegensatz zu anderen MCUs sollen die Register der STM32{ XE "STM32" } nicht direkt angesprochen werden. Dafür dienen die Funktionen der Standard Peripheral Library.  
Sie ist auf <http://www.st.com/> zusammen mit einer Dokumentation (Datei: stm32f10x\_stdperiph\_lib\_um.chm) herunterladbar.