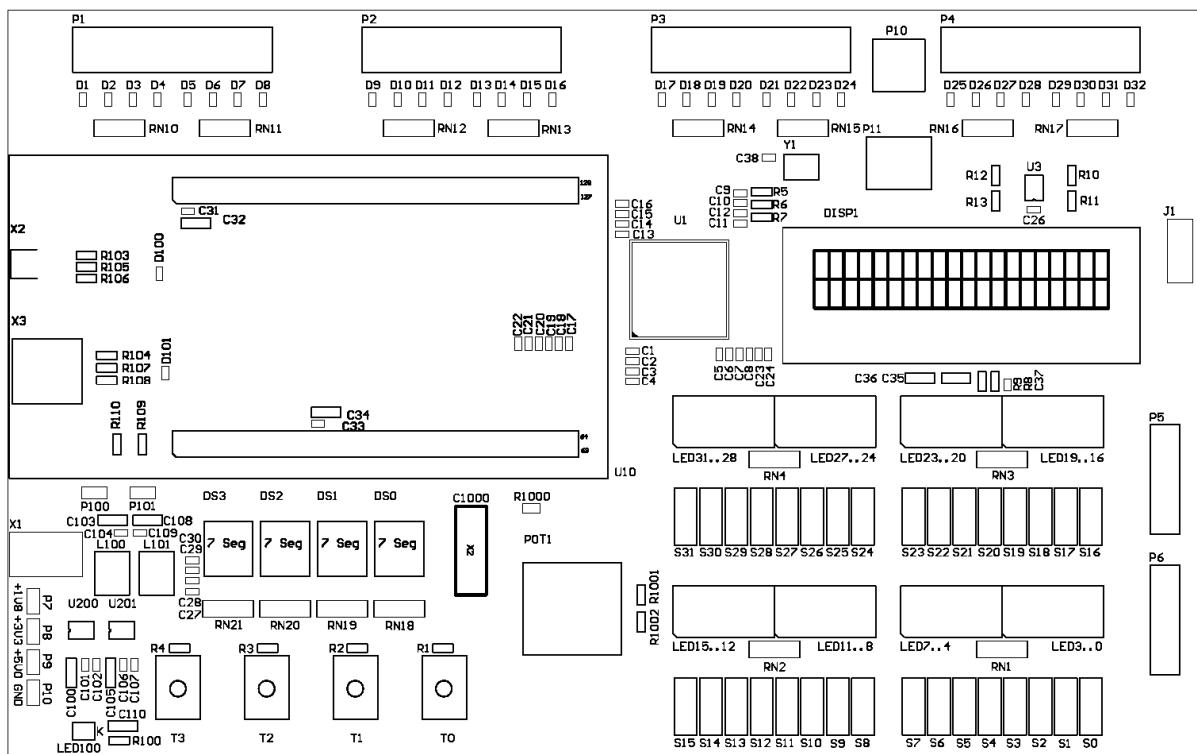


Version 2.1



Inhalt

1	Übersicht CT Board	4
1.1	Mikrocontroller Board	4
1.2	Programmieren des Mikrocontroller Boards	4
2	Keil µVision	5
2.1	Installation	5
2.2	ST-Link Programmer	5
3	Coding Style	6
3.1	Installation	6
3.2	Benutzung	6
4	Erstellen eines C Projekts	7
4.1	Projektverzeichnis vorbereiten	7
4.2	Neues Projekt erstellen	8
4.3	Debugger einrichten	11
4.4	Linker einrichten	13
4.5	Optionale Einstellungen	14
4.6	Beispielprojekt	15
5	Erstellen eines Assembler Projekts	17
5.1	Unterschiede zu C Projekt	17
5.2	Erweitertes Assembler Projekt	18
5.3	Minimales Assembler Projekt	19
6	Beispiel Projekte	20
7	Kompilieren und debugging in Keil µVision	21
7.1	Kompilieren und linken	21
7.2	Download auf Zielsystem mit Remote Debugger	21
7.3	Ausführen des Programms in Einzelschritten	22
7.4	Breakpoints	22

8	CT Board Mode Switch	23
9	Eingabemöglichkeiten	24
9.1	DIP Switches.....	25
9.2	Buttons.....	26
9.3	HEX Switch	27
9.4	Potentiometer.....	28
10	Ausgabemöglichkeiten	29
10.1	LED Balken	30
10.2	7-Segmentanzeigen.....	31
10.3	LCD Character Display	34
11	General Purpose Ein- / Ausgabe.....	41
11.1	GPIO über CPLD	42
11.2	GPIO über Mikrocontroller	44
11.3	Externes Speicherbus Interface.....	48
11.4	Externes Speicherbus Interface (zur Anzeige am Oszilloskop)	49

1 Übersicht CT Board

Das CT Board besteht im Wesentlichen aus zwei Komponenten:

- Mikrocontroller Board STM32F429 Discovery von ST Microelectronics
- I/O Erweiterung via CPLD

1.1 Mikrocontroller Board

Das Discovery Board enthält den Mikrocontroller. Es handelt sich dabei um einen STM32F429ZI von ST Microelectronics. Der Mikrocontroller enthält einen ARM Cortex-M4F Kern. Die Handbücher zum Discovery Board und dem Mikrocontroller sind unten verlinkt.

Discovery Board User Manual: http://www.st.com/.../user_manual/DM00093903.pdf

STM32F4xx Reference Manual: http://www.st.com/.../reference_manual/DM00031020.pdf

STM32F429 Datenblatt: <http://www.st.com/.../datasheet/DM00071990.pdf>

Diese Dokumente sind in der Entwicklungsumgebung (Siehe Kapitel 2) über das Book Tab aufrufbar.

1.2 Programmieren des Mikrocontroller Boards

Eine detaillierte Beschreibung dieses Vorgangs finden Sie im Discovery Board User Manual. Die Programmierschnittstelle (Mini USB) befindet sich auf der linken Seite des Discovery Boards (Siehe Abbildung 1).

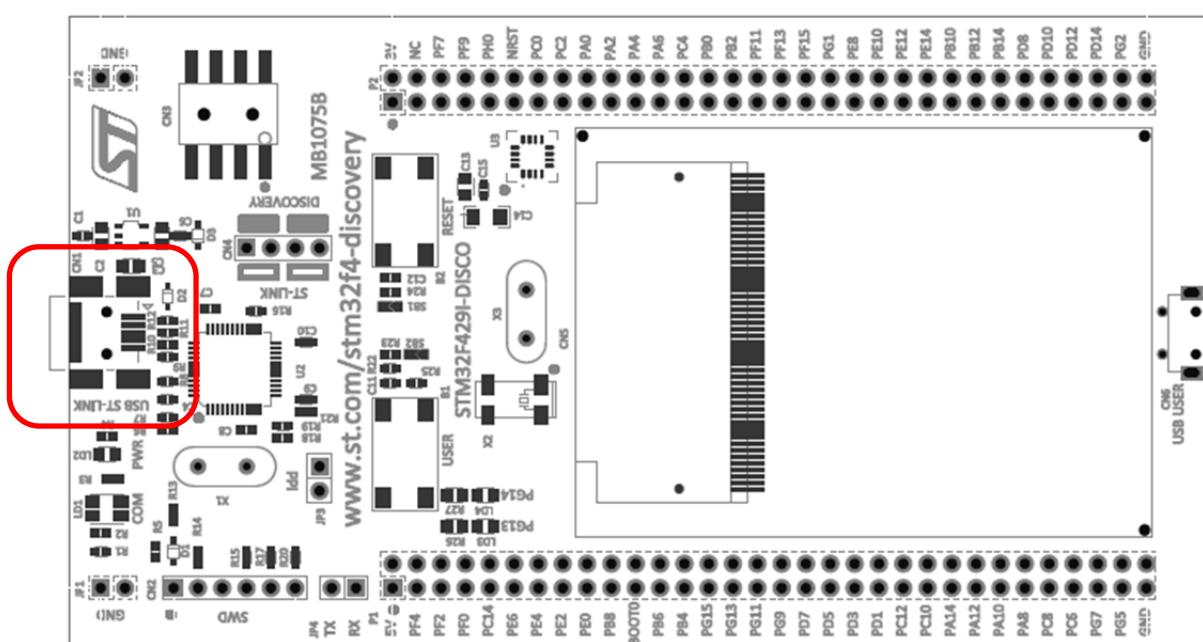


Abbildung 1: Discovery Board

2 Keil µVision

Für das Erstellen von Software Projekten für das CT Board wird Keil µVision 5 verwendet. Die Light Version von µVision kann gratis heruntergeladen werden, ist aber beschränkt auf 32KB (Grösse des Binaries).

Download µVision: <http://www.keil.com/arm/mdk.asp>

2.1 Installation

Bei der Installation bzw. beim erstmaligen starten von µVision wird man aufgefordert, ein Software Pack zu installieren. Für die Arbeit mit dem CT Board stellt das InES ein eigenes Pack bereit, welches separat installiert werden muss.

Da auf dem CT Board ein STM32F429ZI von ST Microelectronics verbaut ist, kann optional das STM32F4xx_DFP installiert werden.

Das InES µVision Pack kann über OLAT heruntergeladen werden. Installieren lässt sich das Pack mit einem Doppelklick auf die heruntergeladene .pack Datei.

2.2 ST-Link Programmer

Um den integrierten ST-Link nutzen zu können, muss das CT Board mit einem USB-Kabel mit dem PC verbunden werden. Beim ersten Verbinden wird Windows nach den entsprechenden Treibern fragen. Mit der Installation von Keil µVision 5 werden die benötigten Treiber in folgendes Verzeichnis kopiert.

ST-Link Treiber: C:\Keil_v5\ARM\STLink

3 Coding Style

Um den InES Coding Style umzusetzen wird Uncrustify (<http://uncrustify.sourceforge.net>), ein Source Code Beautifier, eingesetzt. Verwenden Sie die Version die auf OLAT (.zip) bereitgestellt wird. Diese Version ist bereits entsprechend den InES Vorgaben eingestellt.

3.1 Installation

Die ZIP Datei direkt nach C:\ entpacken. Danach kann in Keil µVision über den Menupunkt „Tools → Customize Tools Menu...“ der Beautifier eingerichtet werden (Siehe Abbildung 2).

- Command: C:\uncrustify-0.61.3\do_command.cmd
- Initial Folder: .\app
- Arguments: .*.c .*.h

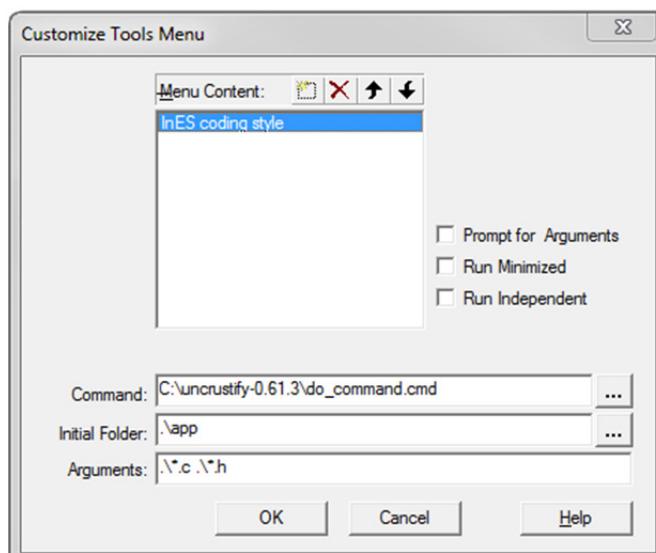


Abbildung 2: Uncrustify Einstellungen

3.2 Benutzung

Wenn das Tools Menu gemäss Kapitel 3.1 eingerichtet wurde kann der Beautifier über den Menupunkt „Tools → InES coding style“ gestartet werden.

4 Erstellen eines C Projekts

Dieses Kapitel beschreibt die Einrichtung eines Projekts in Keil µVision 5. Es wird dabei ein C Projekt erstellt. Möchte man stattdessen ein Assembler Projekt erstellen, sind die in Kapitel 5 beschriebenen Hinweise zu beachten.

4.1 Projektverzeichnis vorbereiten

Erstellen Sie ein neues Verzeichnis und benennen Sie es entsprechend dem Projekt. In diesem Verzeichnis legen Sie folgende zwei neuen Verzeichnisse an:

- app Hier kommt Ihr Quellcode rein, den Sie schreiben.
- build In diesem Verzeichnis speichert der Compiler die Artefakte, die während dem Erstellen des Executables anfallen (siehe Kapitel 4.5).

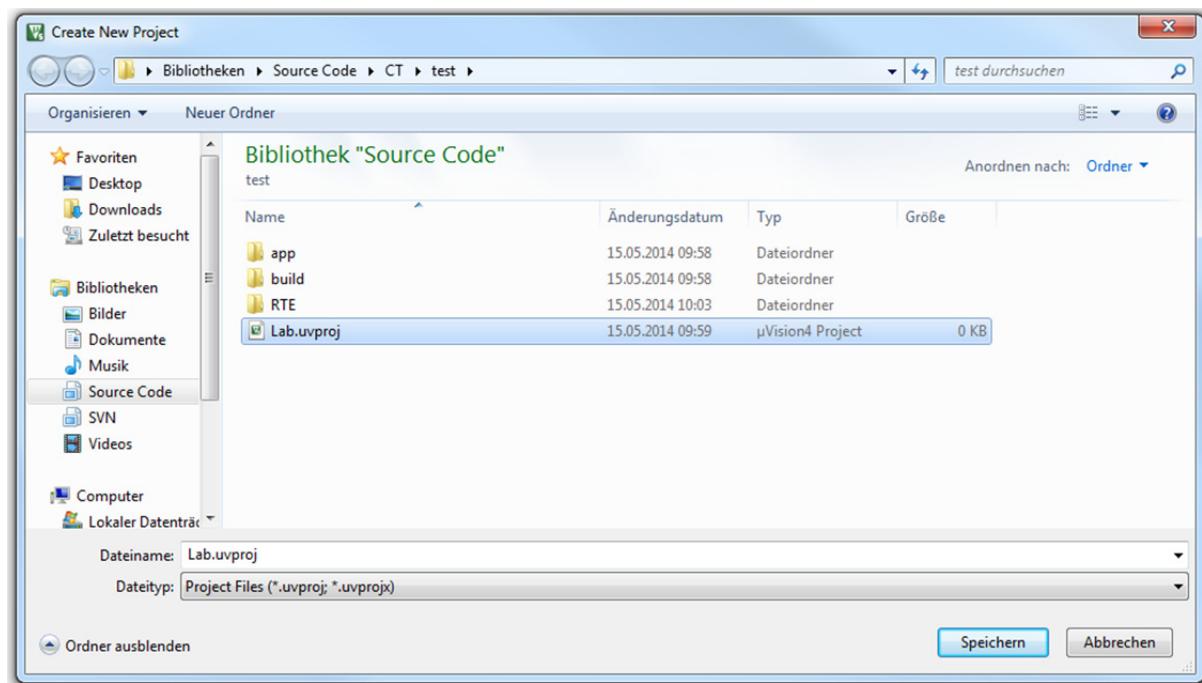


Abbildung 3: Neues Projekt erstellen

4.2 Neues Projekt erstellen

Keil µVision 5 starten und über den Menupunkt „Project → new µVision Project“ ein neues Projekt erstellen. Im anschliessenden Dialog kann nun das gewünschte Device ausgewählt werden.

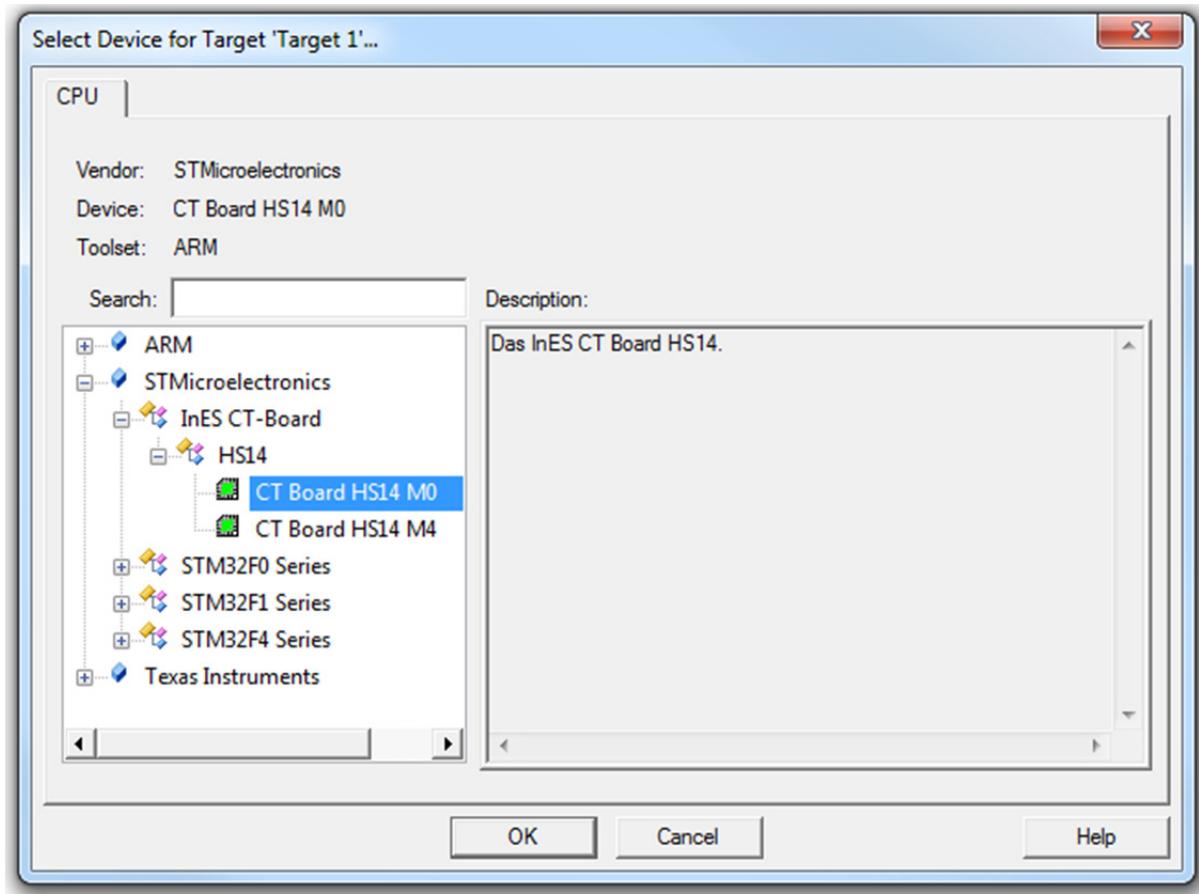


Abbildung 4: Device auswählen

Verfügbare Devices:

- CT Board HS14 M0 Mit diesem Device wird nur das Cortex-M0 Instruction Set verwendet.
- CT Board HS14 M4 Mit diesem Device hat man Zugriff auf das volle Cortex-M4 Instruction Set.

Falls das InES µVision Pack installiert wurde, können die Komponenten „Startup“ und „Utilities“ ausgewählt werden. Diese finden sich unter dem Menupunkt „Project → Manage → Run-Time Environment“ innerhalb der Kategorie „Device“.

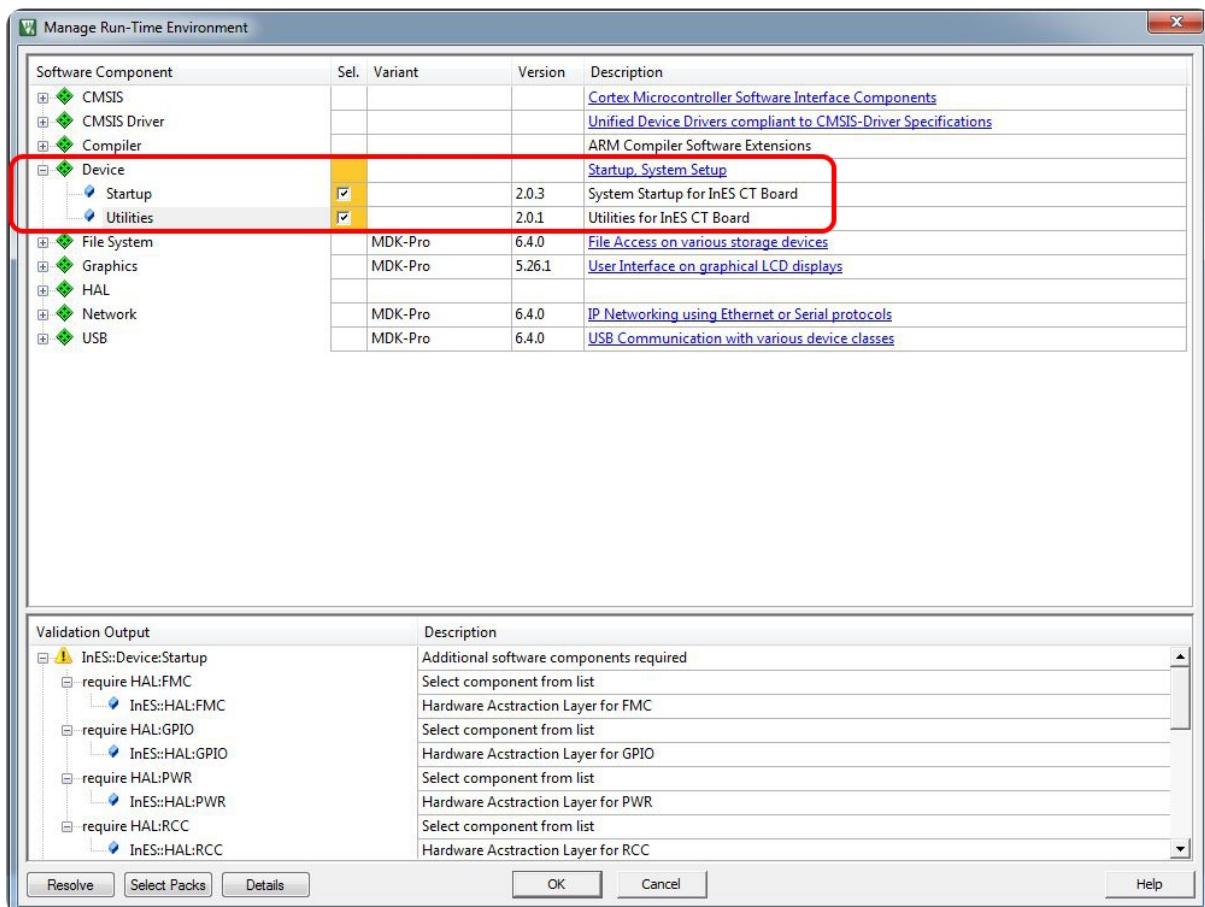


Abbildung 5: Run-Time Komponente hinzufügen

Relevante Komponenten:

- Startup Diese Komponente muss ausgewählt werden, da sie alles enthält, um den Mikrocontroller zu initialisieren und den FMC (Flexible Memory Controller) für den Zugriff auf das CT Board einrichtet.
- Utilities Diese Komponente enthält die Funktionen, um lesend bzw. schreibend auf das Memory zuzugreifen.

```
#include <utils_ctboard.h>
```

Da die ausgewählten Komponenten auf die verschiedenen HAL zugreifen, müssen diese ebenfalls selektiert und somit in das Projekt einbezogen werden.

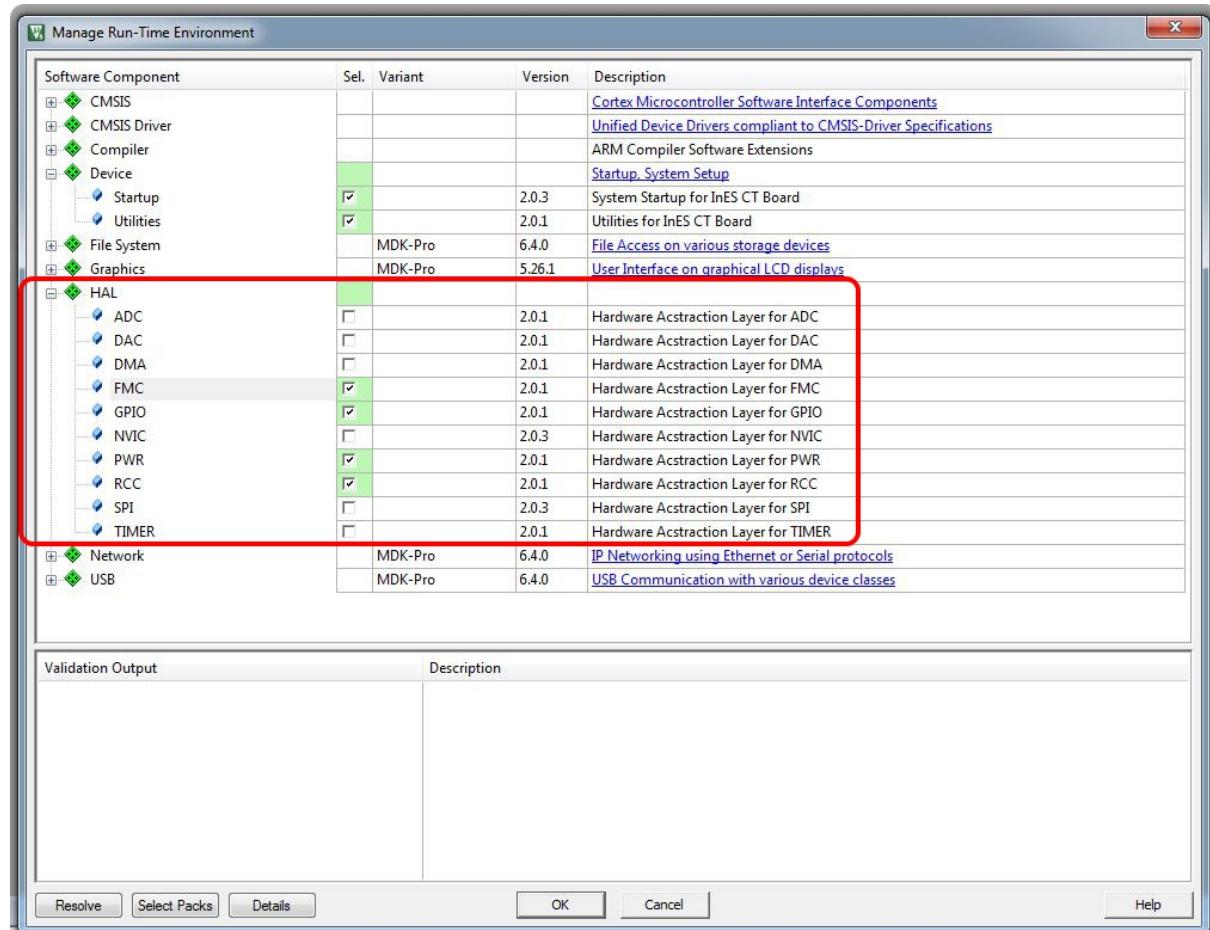


Abbildung 6: Run-Time Komponente HAL hinzufügen

Unter der Kategorie HAL muss minimal FMC, GPIO, PWR und RCC ausgewählt werden. Im Textfeld Description sollte nun keine Meldung mehr erscheinen. Zudem sollten nun alle Auswahlen grün hinterlegt sein.

4.3 Debugger einrichten

Standardmässig wird als Programmer/Debugger der ULink2 von Keil eingerichtet. Wir verwenden jedoch den, ins CT Board integrierten, ST-Link Debugger.

Hier den Menupunkt „Project → Options for Target ,Target 1“ auswählen:

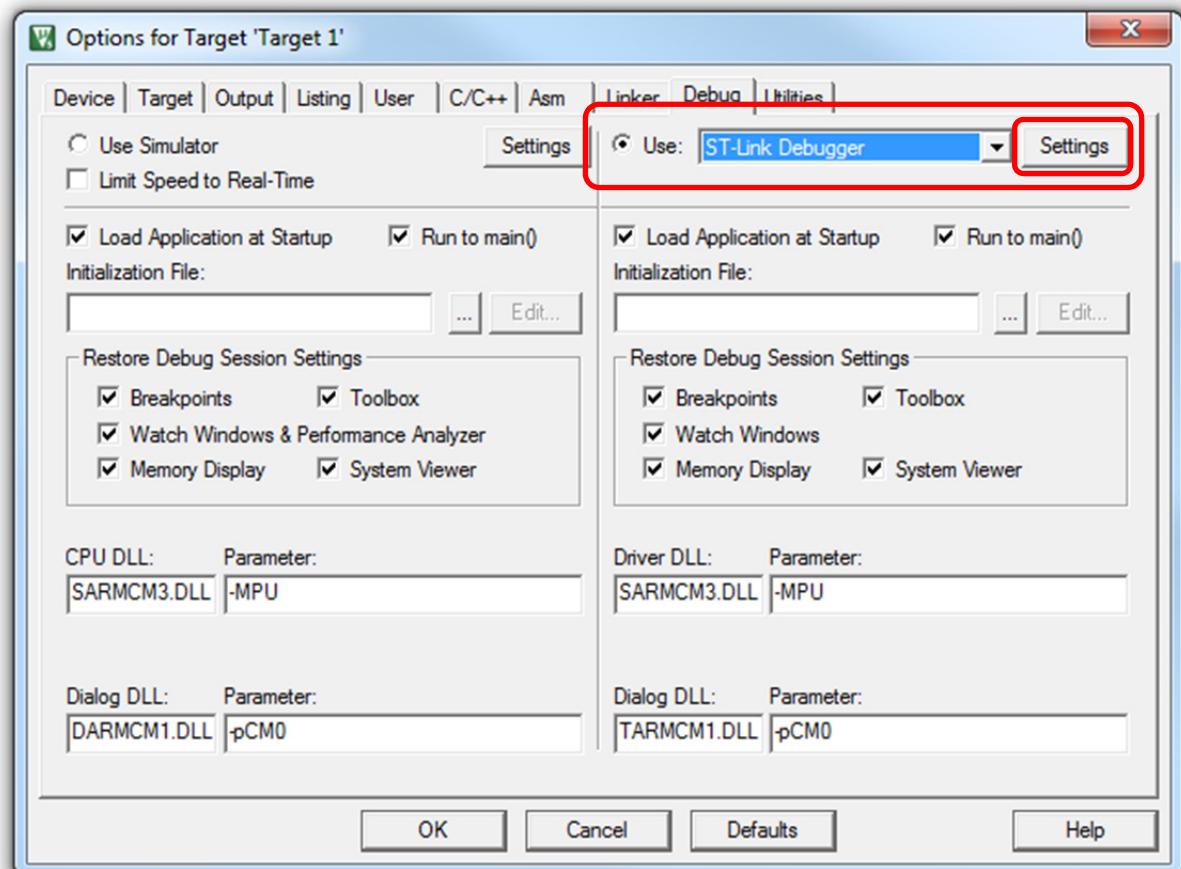


Abbildung 7: Debug Einstellungen

Als nächstes müssen die Debugger Settings angepasst werden (Button „Settings“). Um diese Einstellungen vorzunehmen muss das CT Board an den Computer angeschlossen und eingeschaltet sein.

Einerseits muss der Debug Port von „JTAG“ auf „SW“ umgestellt werden. Andererseits muss ein geeigneter Flash-Algorithmus ausgewählt werden. Der Flash Speicher des STM32F4xx ist 2MB gross. Entsprechend wird der „STM32F4xx 2MB Flash“ Algorithmus ausgewählt.

Hinweis: Es kann sein, dass bei vorgegebenen Projekten, vom SVN oder OLAT, die Debug Einstellungen nicht korrekt sind. Bei Problemen mit dem Debuggen oder Downloaden von Programmen kurz die Einstellungen prüfen.

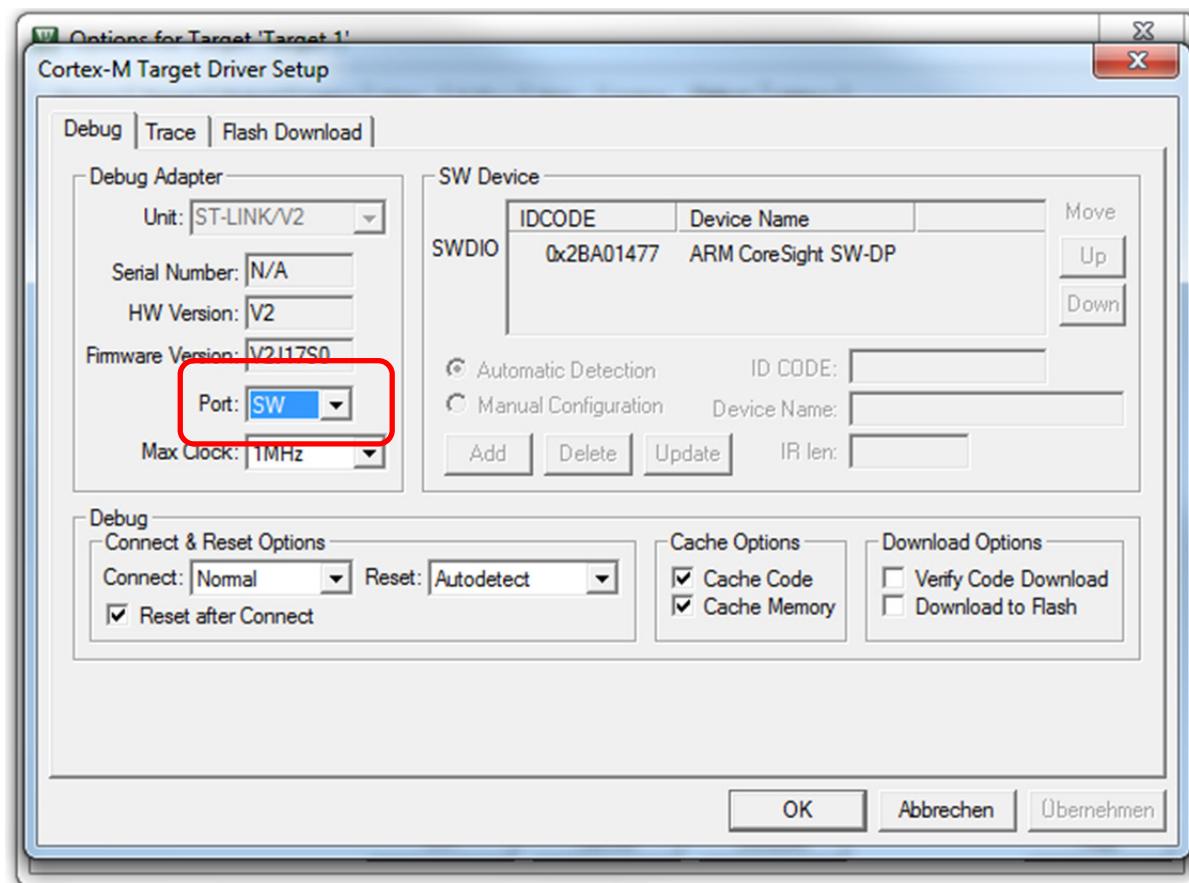


Abbildung 8: Erweiterte Debug Einstellungen (1)

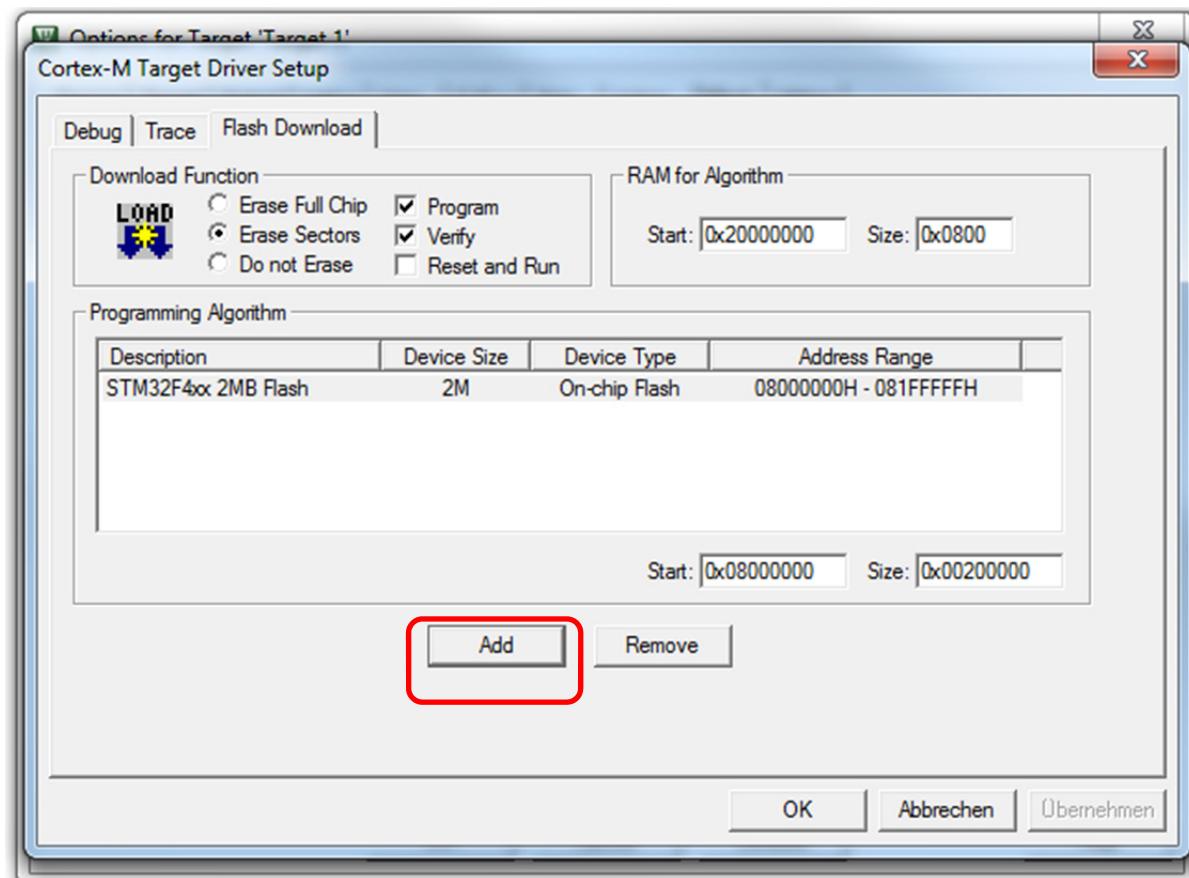


Abbildung 9: Erweiterte Debug Einstellungen (2)

4.4 Linker einrichten

Beim linken der einzelnen Objektdateien wird folgende Warnung ausgegeben:

```
warning: L6314W: No section matches pattern *(InRoot$$Sections).
```

Um diese eine Warnung auszublenden kann der Parameter `--diag_suppress 6314` dem Linker mitgegeben werden (siehe Abbildung 10).

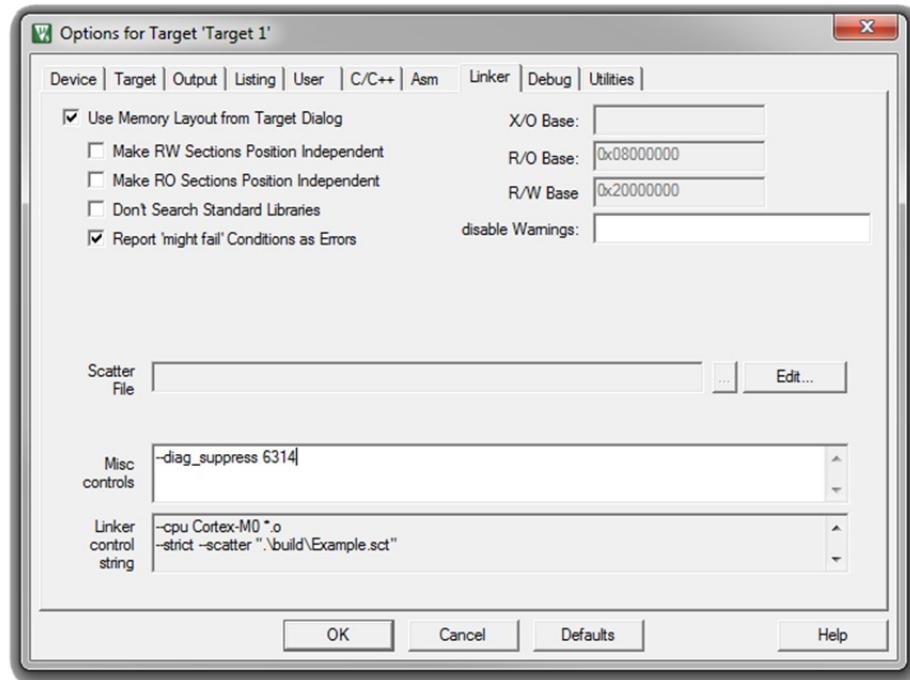


Abbildung 10: Linker Optionen

4.5 Optionale Einstellungen

Dieser Schritt ist optional, hilft aber bei der Organisation des Projektordners. Mittels „Select Folder for Objects...“ bzw. „Select Folder for Listings...“ lässt sich der Ordner festlegen, in welchem die Buildartefakte gespeichert werden (Siehe 4.1, *build* Ordner).

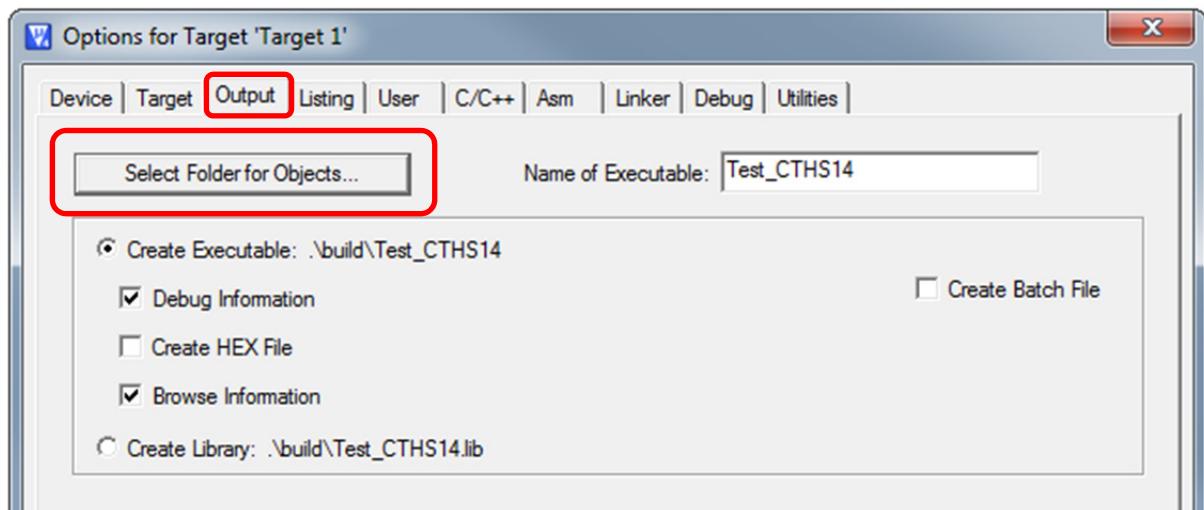


Abbildung 11: Output Einstellungen

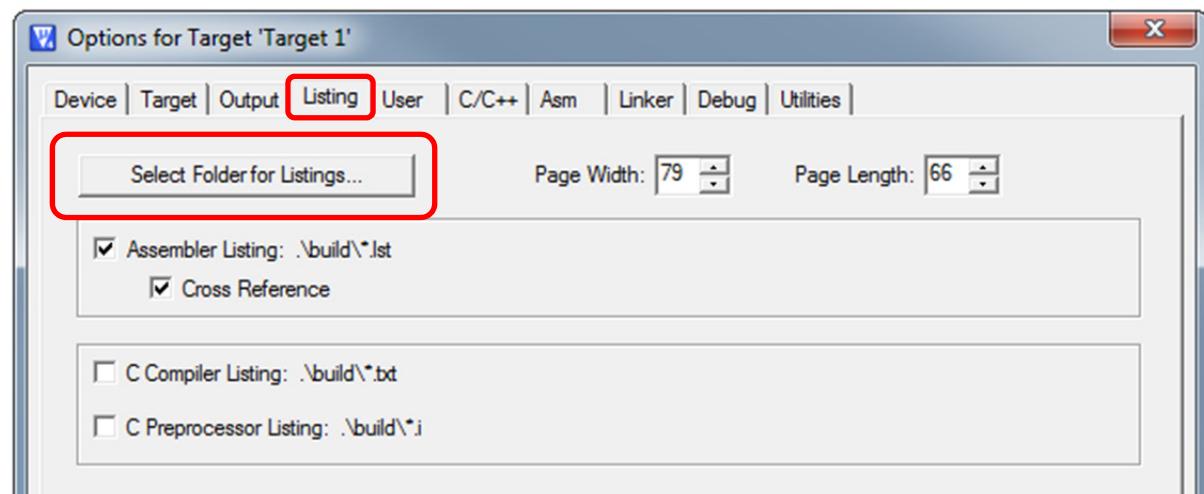


Abbildung 12: Listing Einstellungen

4.6 Beispielprojekt

Der unten abgebildete Code stammt aus dem Projekt „*Example_C*“. Das Projekt besteht aus einem einzigen Modul.

Beispielcode Modul main.c:

```
#include <utils_ctboard.h>

#define ADR_LED          0x60000100
#define MASK_LED_UPPER    0xfffff0000
#define MASK_LED_LOWER    0x0000ffff
#define PAUSE              0xfffffff

void wait(uint32_t value)
{
    for(; value > 0; value--) { }

}

int main(void)
{
    while(1) {
        write_word(ADR_LED, MASK_LED_UPPER);
        wait(PAUSE);
        write_word(ADR_LED, MASK_LED_LOWER);
        wait(PAUSE);
    }
}
```

Um eine neue Datei zu erzeugen und dem Projekt hinzuzufügen klicken Sie im Projektfenster mit der rechten Maustaste auf den Ordner „Source Group 1“ und anschliessend auf „Add New Item to Group“.

Um eine bereits existierende Datei einzubinden wählen Sie „Add Existing Files to Group“.

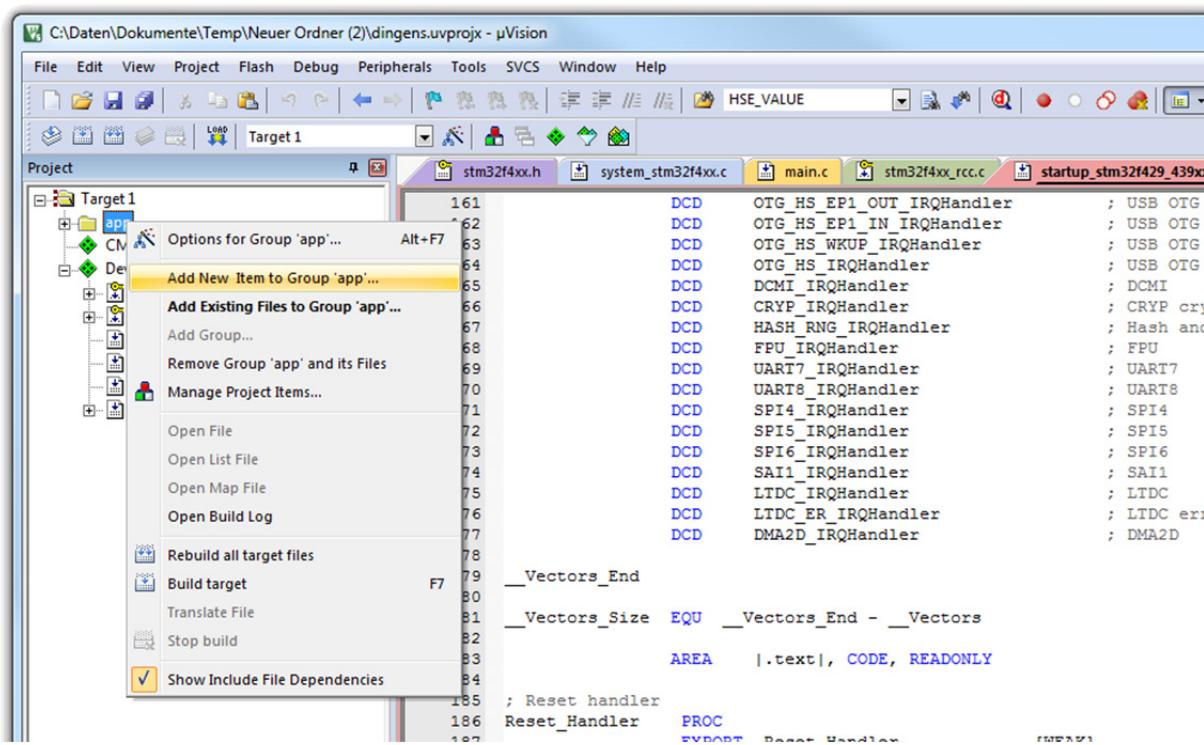


Abbildung 13: Datei hinzufügen (1)

Wählen Sie den entsprechenden Typ. Geben Sie einen Namen ein und kontrollieren Sie den Speicherort (\app).

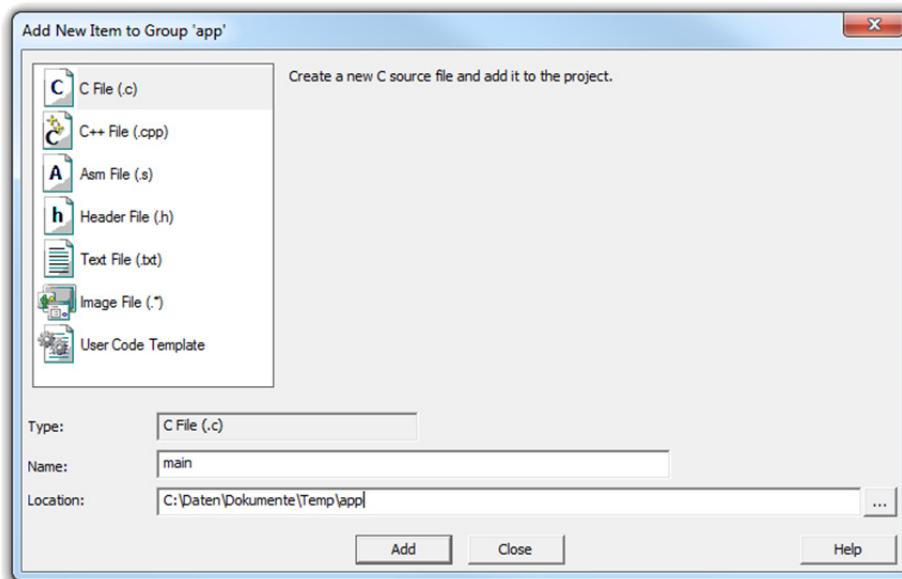


Abbildung 14: Datei hinzufügen (2)

5 Erstellen eines Assembler Projekts

In diesem Kapitel wird das Erstellen eines Assembler Projekts in µVision 5 beschrieben. Die meisten Schritte sind identisch zu denen in Kapitel 4, es werden daher nur die entsprechenden Unterschiede erklärt.

5.1 Unterschiede zu C Projekt

Der einzige Unterschied besteht darin, dass das Modul „Utilities for InES CT Board“ nicht verwendet werden kann (Kapitel 4.2 bzw. Abbildung 15).

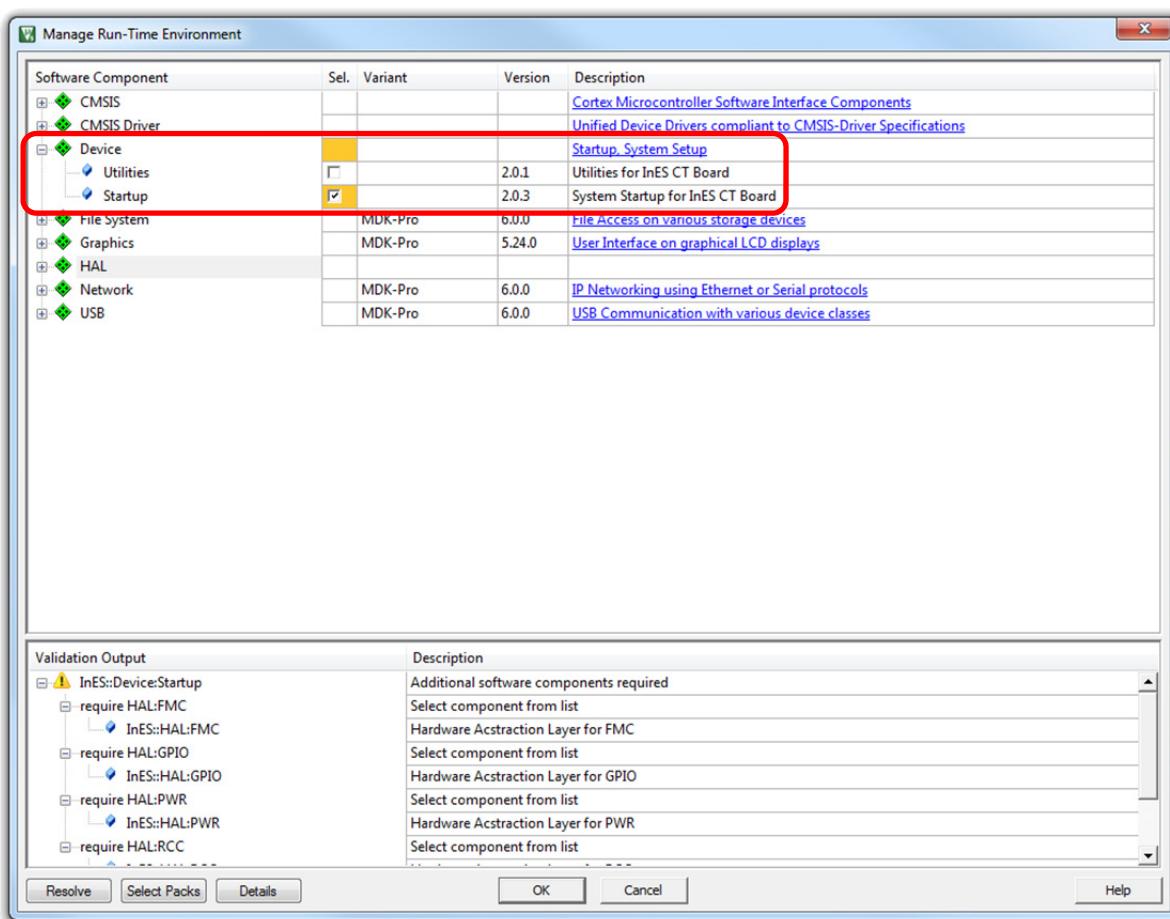


Abbildung 15: Run-Time Komponente hinzufügen (ASM)

5.2 Erweitertes Assembler Projekt

Ein erweitertes Assembler Projekt meint ein Assembler Projekt, welches den Mikrocontroller in derselben Weise aufsetzt wie das C Projekt (Interrupt Vektor Tabelle, Memory Controller). Für den eigenen Assembler Code muss man eine Procedure „main“ erstellen (Siehe Codebeispiel weiter unten), welche aus dem Startup Code aufgerufen wird.

Beispielcode mit „main“ Procedure, welche aus Startup aufgerufen wird:

```
AREA myCode, CODE, READONLY

THUMB

ADR_LED      EQU      0x60000100
ADR_DIPSW    EQU      0x60000200

main          PROC
              EXPORT main

loop          LDR      R0, =ADR_LED
              LDR      R1, =ADR_DIPSW
              LDR      R2, [R1, #0]
              STR      R2, [R0, #0]
              B       loop

ENDP

ALIGN
END
```

Hinweis: Falls Sie Code aus einem PDF Dokument herauskopieren, kann es sein, dass Whitespace characters ([Wiki-Link](#)) mitkopiert werden, die den Assembler durcheinander bringen.

5.3 Minimales Assembler Projekt

Für ein minimales Assembler Projekt (ohne aufsetzen des Mikrocontrollers) wird einfach kein Startup Code ausgewählt (Siehe Kapitel 4.2). Dies bedeutet, dass der Memory Controller nicht aufgesetzt ist und somit kein Zugriff auf die Peripherie des CT Boards gemacht werden kann.

Beispielcode für minimales Assemblerprojekt:

```
AREA RESET, DATA, READONLY
EXPORT __Vectors

__Vectors
    DCD 0x2002ffff          ; top of stack
    DCD Reset_Handler        ; reset vector

AREA myCode, CODE, READONLY
THUMB

CONST_VALUE_X EQU 12

ENTRY

Reset_Handler

; -- Your code belongs here -----

    LDR R0, =0
    LDR R1, =CONST_VALUE_X
loop      ADD R0, R1
        B loop

; -- Your code ends here -----

END
```

6 Beispiel Projekte

Anstelle des Erstellens eines neuen Projekts, kann auch über das PACK auch ein Beispielprojekt geladen werden. In diesem sind alle Projekt- und Debugg- Einstellungen schon vorgenommen und der Code kann direkt auf das CT-Board geladen werden.

Dazu unter Menupunkt „Project“ auf „Manage“ → „Pack Installer...“ öffnen.

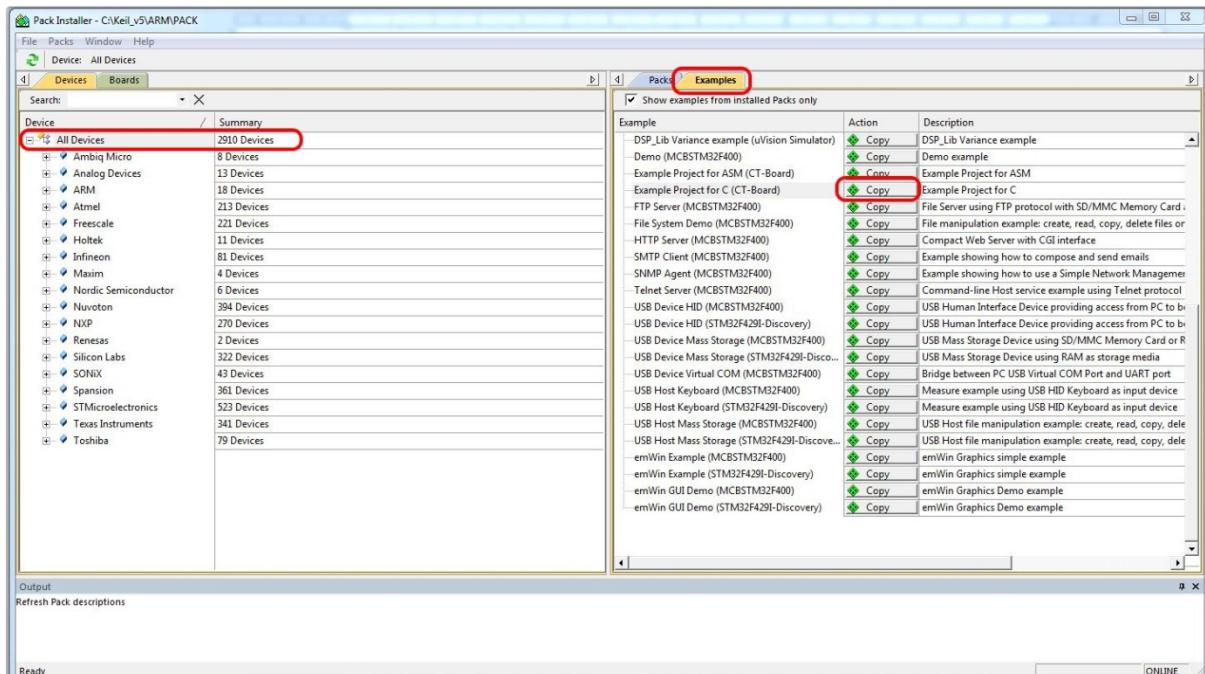


Abbildung 16: Beispiel Projekt importieren

Unter Registrierkarte „Devices“ „All Devices“ auswählen. Öffnen sie nun die Registrierkarte „Examples“, dort sollten nun die Einträge „Example Project for ASM (CT-Board)“ und „Example Project for C (CT-Board)“ vorhanden sein.

- *Example Project for ASM (CT-Board)* Minimales Assembler Projekt
- *Example Project for C (CT-Board)* Minimales C Projekt

Wählen sie nun abhängig davon, ob in C oder Assembler programmiert werden soll, dass richtige Beispiel-Projekt aus und klicken sie auf “Copy”.

Nun muss noch der Speicherort für das Projekt festgelegt werden. Durch bestätigen mit „OK“ öffnet sich das gewählte Beispiel-Projekt. Es kann kompiliert und geladen werden.

Hinweis: Die Darstellung in Abbildung 16 kann je nach verwendeter Version von µVision leicht abweichen.

7 Kompilieren und debugging in Keil µVision

In den vorherigen Kapiteln wurde gezeigt, wie ein C- bzw. Assemblerprojekt erstellt wird. In diesem Kapitel wird beschrieben wie man ein Projekt kompiliert, linkt und danach debugged.

Die unten beschriebenen Techniken können am Beispiel „*Example_C*“ (Siehe Kapitel 4.6) ausprobiert werden.

7.1 Kompilieren und linken

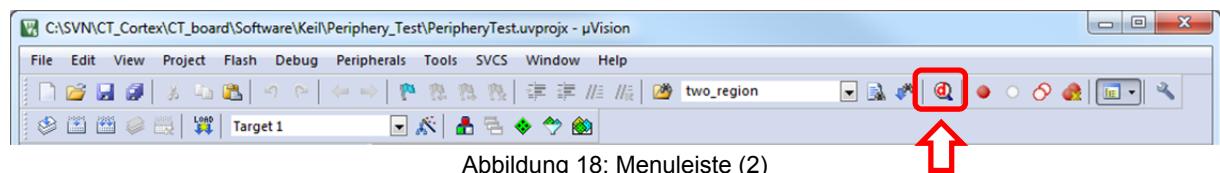
Das Kompilieren des Projektes kann über die Buttons *Build* oder *Rebuild* gestartet werden. Der Unterschied besteht darin, dass bei *Rebuild* alle Files erneut kompiliert werden, während bei *Build* nur die veränderten Files kompiliert werden. Das Ergebnis wird Ihnen im Build Output Fenster von µVision 5 angezeigt.



7.2 Download auf Zielsystem mit Remote Debugger

Schalten Sie das Zielsystem ein. Kontrollieren Sie, dass die USB-Verbindung auf der linken Seite des Zielsystems mit dem PC verbunden ist.

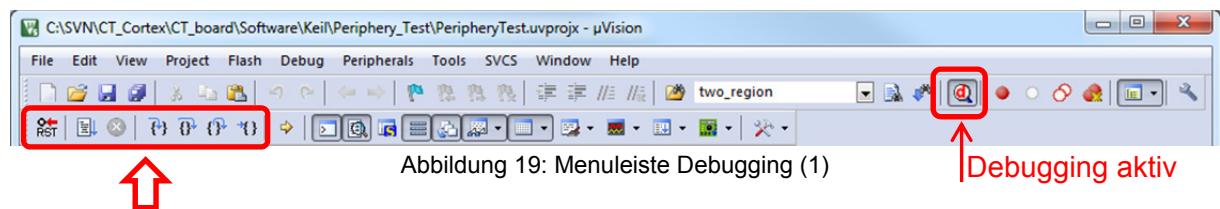
Starten Sie den Remote Debugger. Das Programm wird nun über die USB Schnittstelle in den Flashspeicher des Zielsystems geladen und vor der ersten Instruktion im Code Bereich angehalten.



7.3 Ausführen des Programms in Einzelschritten

Mit folgenden Aktionen können Sie Ihr Programm Schritt für Schritt ausführen:

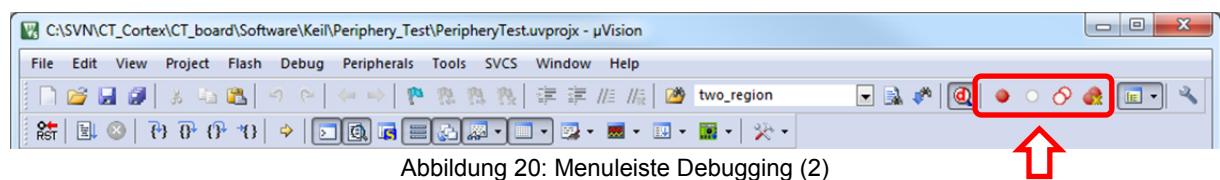
- Mit dem Button „**Step**“ (**F11**) wird jeweils eine Zeile ausgeführt.
- Mit „**Step Over**“ (**F10**) wird jeweils eine Zeile ausgeführt. Dabei wird jedoch nicht in aufgerufene Funktionen gestept.
- Durch den Befehl „**Step Out**“ (**Ctrl + F11**) kann man einfach aus einer Subroutine zur nächsten Zeile nach dem Aufruf der Subroutine springen.
- Mittels „**Run to Cursor Line**“ (**Ctrl + F10**) wird aller Code bis zum Cursor ausgeführt.
- Mit dem Button „**Run**“ (**F5**) wird aller Code bis zum nächsten Breakpoint ausgeführt.
- Mit „**Reset**“ wird die CPU zurückgesetzt.



7.4 Breakpoints

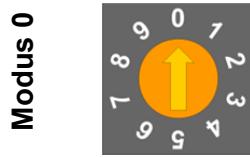
Das Setzen von Breakpoints funktioniert mit folgenden Aktionen:

- Mittels „**Insert/Remove Breakpoint**“ (**F9**) wird an der Stelle des Cursors ein Breakpoint eingefügt oder entfernt
- Durch den Befehl „**Enable/Disable Breakpoint**“ (**Ctrl + F9**) wird der selektierte Breakpoint aktiviert/deaktiviert.
- Mit dem Button „**Disable all Breakpoints**“ werden alle Breakpoints deaktiviert.
- Mit „**Kill all Breakpoints**“ (**Ctrl + Shift + F9**) werden alle Breakpoints entfernt.



8 CT Board Mode Switch

Das CT Board lässt sich in verschiedenen Modi betreiben. Der gewünschte Modus kann über den Drehschalter P10 ausgewählt werden.



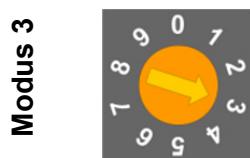
Modus 0
Das CT Board ist vom Microcontrollerboard (STM32F4 Discovery) getrennt. So kann z.B. das LCD (Touch-) Display genutzt werden, dessen Anschlüsse sonst vom Memory Controller verwendet werden.



Modus 1
Die Ports P1 bis P4 sind als GPIO konfiguriert (Siehe Kapitel 11.1).



Modus 2
Die Ports P1 bis P4 sind mit dem Memory Controller verbunden (Siehe Kapitel 11.3).



Modus 3
Die Ports P1 bis P4 sind ebenfalls mit dem Memory Controller verbunden. Die Spezielle Belegung der Ports macht es möglich direkt die Logik Sonden des Oszilloskops anzuschliessen (Siehe Kapitel 11.4).

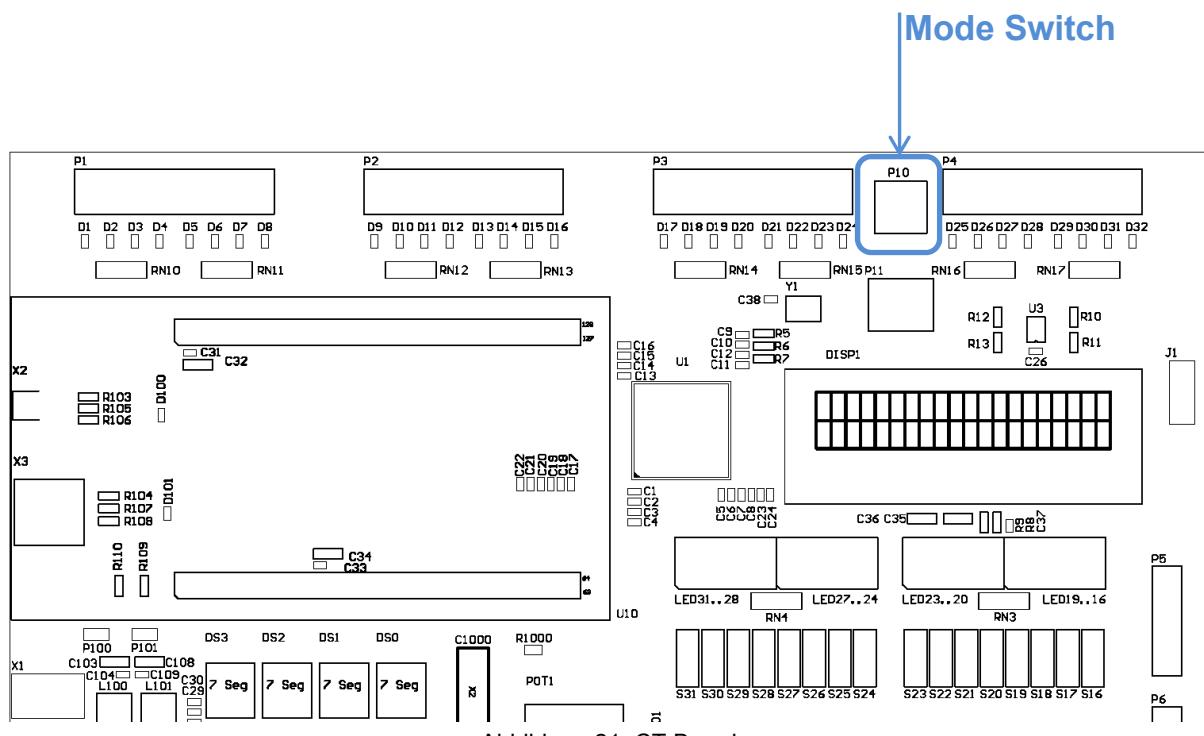
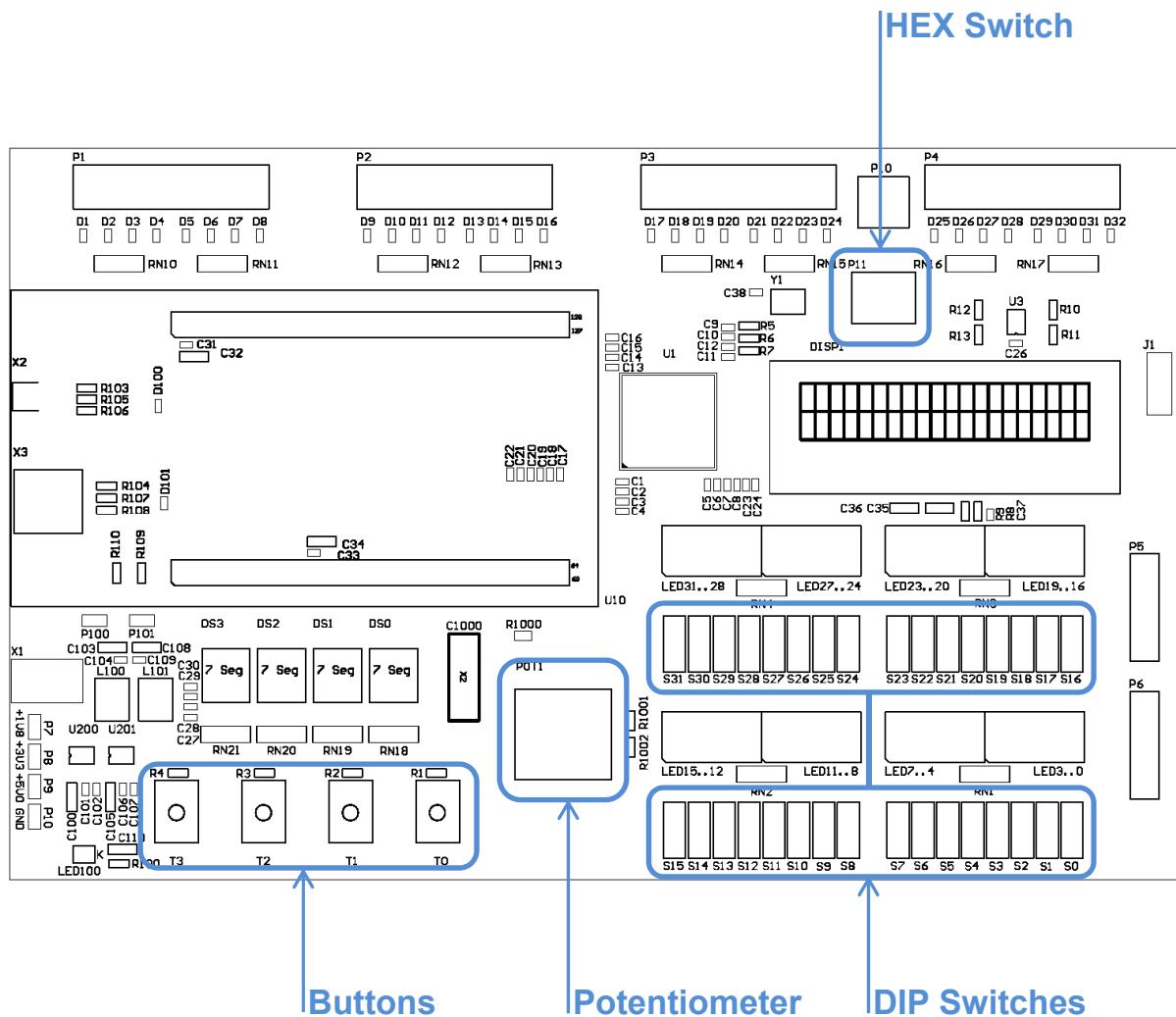


Abbildung 21: CT Board

9 Eingabemöglichkeiten

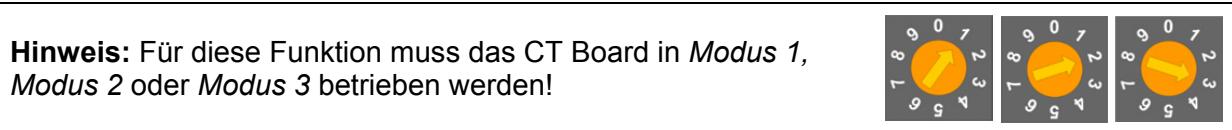
Für die Eingabe von Daten stellt das CT Board folgende Möglichkeiten bereit:

- 4x Taster / Buttons
- 1x Potentiometer 0..3,3V
- 1x HEX Switch 0..F
- 32x DIP Switch



9.1 DIP Switches

Die 32 DIP Switches geben in der Stellung „high“ eine logische „1“ aus (Siehe Blockschaltbild unten).



Basisadresse: **0x6000'0000**
Zugriffsart: **read**

Offset	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x200	S7	S6	S5	S4	S3	S2	S1	S0
0x201	S15	S14	S13	S12	S11	S10	S9	S8
0x202	S23	S22	S21	S20	S19	S18	S17	S16
0x203	S31	S30	S29	S28	S27	S26	S25	S24

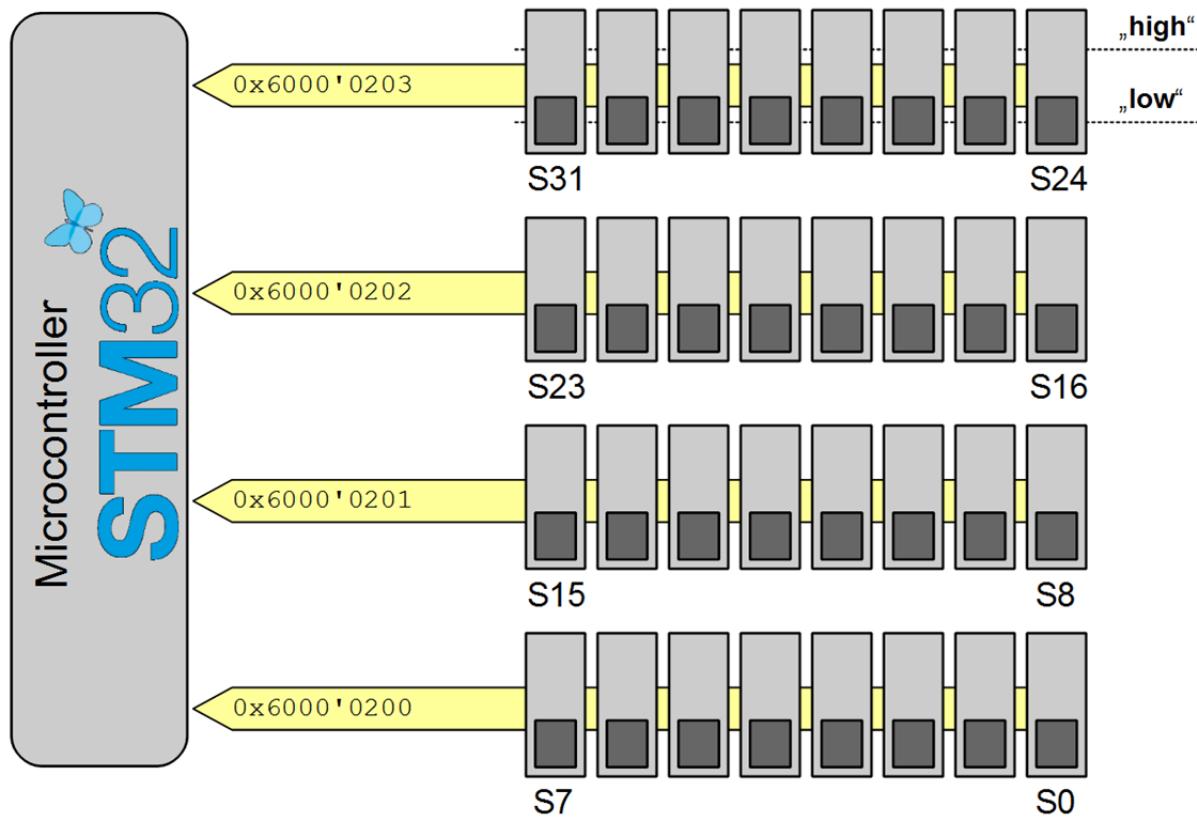


Abbildung 22: DIP Switches

9.2 Buttons

Die Taster T0 bis T3 sind „active-high“ und entprellt. D.h. bei gedrückter Taste wird eine logische „1“ ausgegeben.

Hinweis: Für diese Funktion muss das CT Board in *Modus 1*, *Modus 2* oder *Modus 3* betrieben werden!



Hinweis: Die Taster sind ebenfalls an Port C des Mikrocontrollers angeschlossen. Die Taster können dadurch auch Interrupts auslösen.

Basisadresse: **0x6000'0000**

Zugriffsart: **read**

Offset	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x210	Reserved				T3	T2	T1	T0

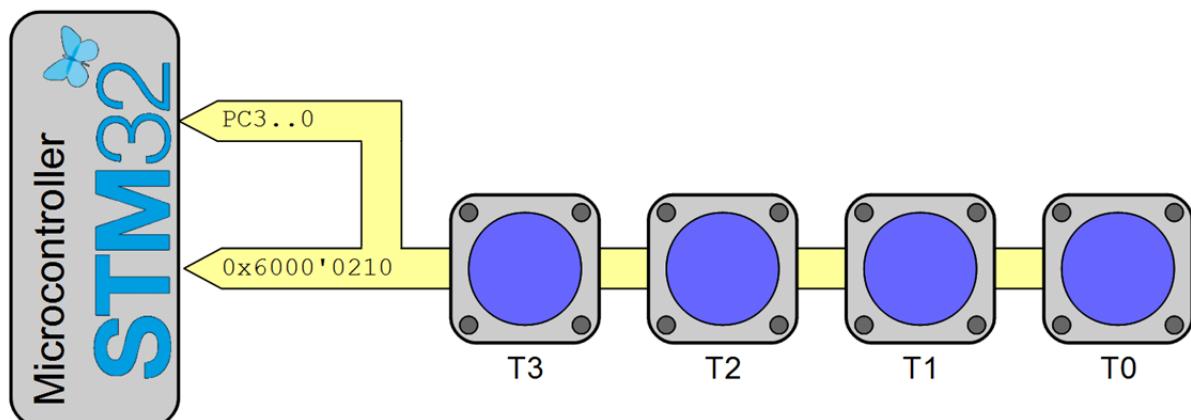
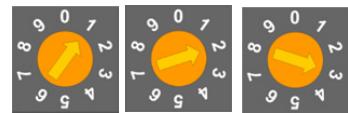


Abbildung 23: Buttons

9.3 HEX Switch

Hinweis: Für diese Funktion muss das CT Board in *Modus 1*, *Modus 2* oder *Modus 3* betrieben werden!



Basisadresse: **0x6000'0000**

Zugriffsart: **read**

Offset	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x211			Reserved				P11	

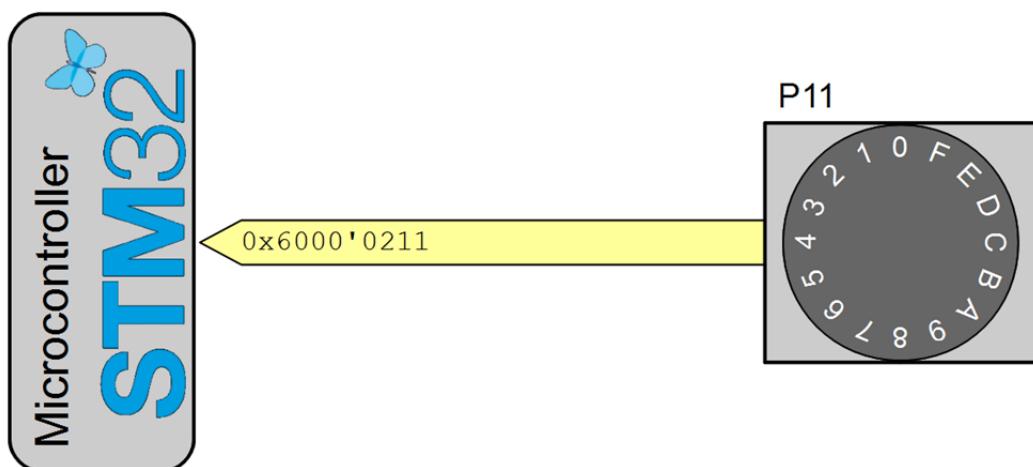


Abbildung 24: HEX Switch

9.4 Potentiometer

Das Potentiometer ist direkt mit dem Microcontroller verbunden. Es ist an Pin 6 des Ports F angeschlossen (Siehe Abbildung 25).

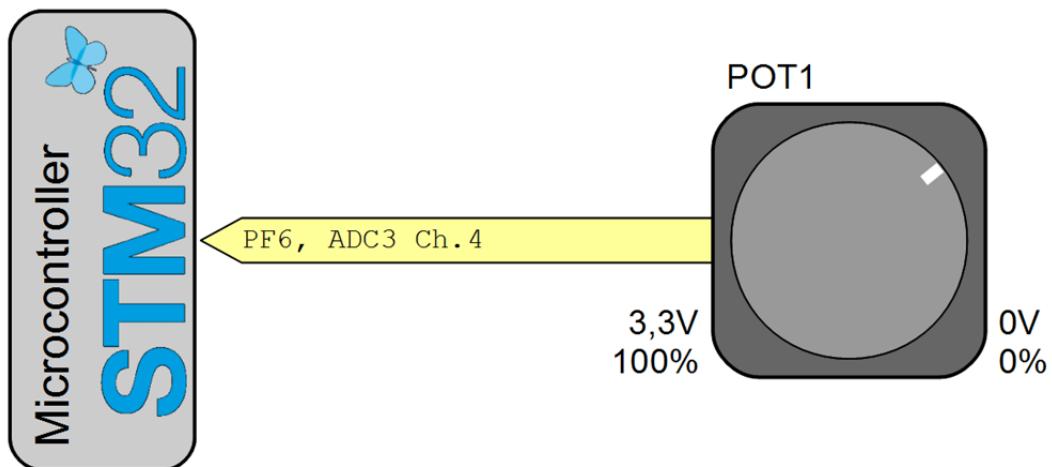
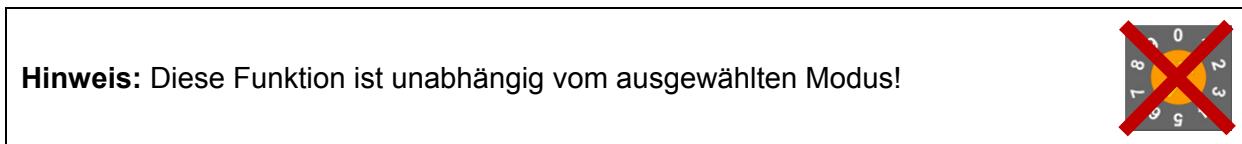
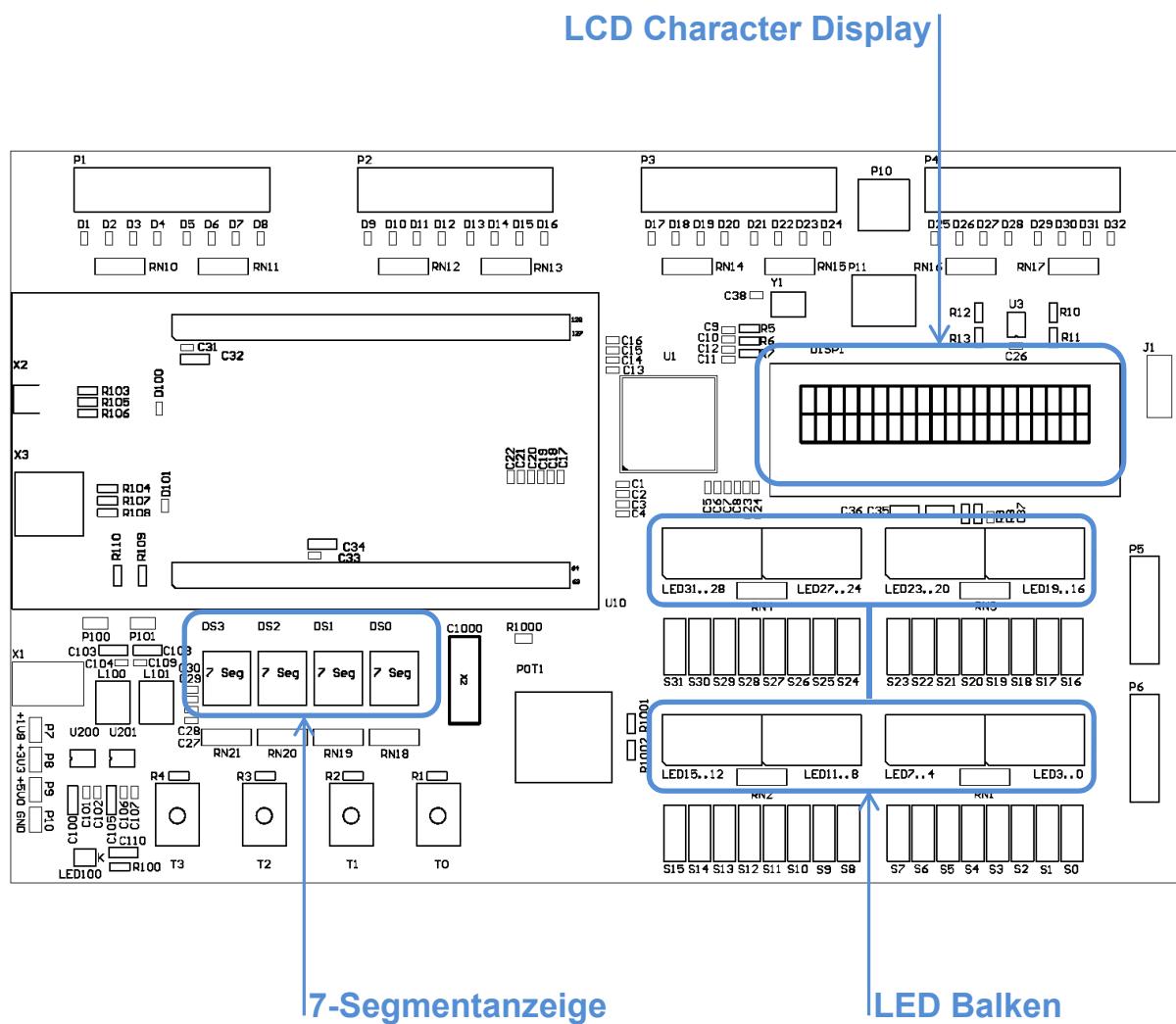


Abbildung 25: Potentiometer

10 Ausgabemöglichkeiten

Für die Ausgabe von Daten stellt das CT Board folgende Möglichkeiten bereit:

- 4x 7-Segmentanzeigen
- 1x LCD Character Display (2x20 Characters)
- 32x LEDs



10.1 LED Balken

Auf dem CT Board sind 8 LED Balken (à 4 LEDs) vorhanden. Die einzelnen LEDs können einzeln angesteuert werden und sind „active-high“. D.h. sie leuchten, wenn eine „1“ geschrieben wird.

Hinweis: Für diese Funktion muss das CT Board in *Modus 1*, *Modus 2* oder *Modus 3* betrieben werden!



Basisadresse: **0x6000`0000**

Zugriffsart: **read / write**

Offset	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x100 Reset Value	LED7..4				LED3..0			
	0	0	0	0	0	0	0	0
0x101 Reset Value	LED15..12				LED11..8			
	0	0	0	0	0	0	0	0
0x102 Reset Value	LED23..20				LED19..16			
	0	0	0	0	0	0	0	0
0x103 Reset Value	LED31..28				LED27..24			
	0	0	0	0	0	0	0	0

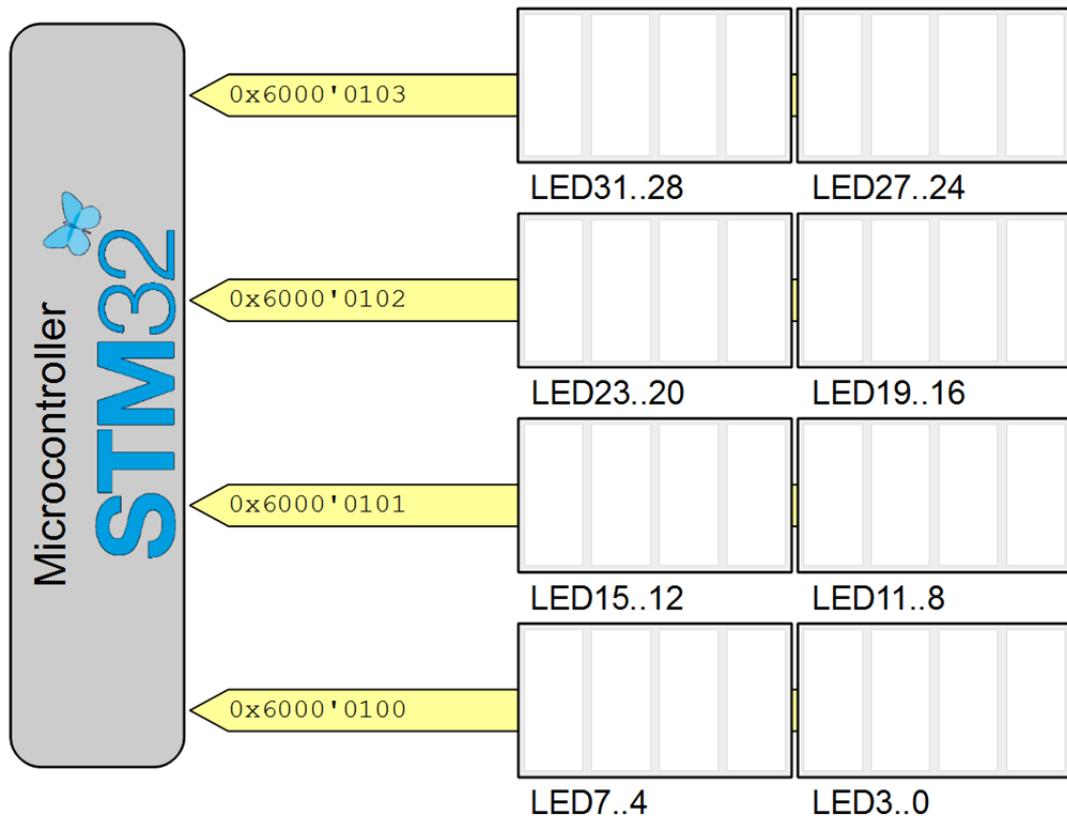


Abbildung 26: LED Balken

10.2 7-Segmentanzeigen

Das CT Board verfügt über vier 7-Segmentanzeigen. Die Anzeige kann über zwei verschiedene Adressbereiche angesprochen werden. Je nachdem auf welchen der beiden Adressbereiche geschrieben wird, werden die Daten entsprechend interpretiert und ausgegeben. Eine detaillierte Beschreibung befindet sich in den Unterkapiteln „Segment Control“ und „Hex Display“.

Hinweis: Für diese Funktion muss das CT Board in *Modus 1*, *Modus 2* oder *Modus 3* betrieben werden!



Segment Control

Über diese Adressen können die Segmente der 7-Segmentanzeigen einzeln angesteuert werden. Die Zuordnung der Bits zu den einzelnen Segmenten ist unten abgebildet. Die einzelnen Segmente der Anzeige sind „active-low“. D.h. sie leuchten, wenn eine logische „0“ geschrieben wird.

Basisadresse: **0x6000`0000**

Zugriffsart: **read / write**

Offset	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	<i>dp</i>	<i>g</i>	<i>f</i>	<i>e</i>	<i>d</i>	<i>c</i>	<i>B</i>	<i>a</i>
0x110	DS0							
Reset Value	1	1	1	1	1	1	1	1
0x111	DS1							
Reset Value	1	1	1	1	1	1	1	1
0x112	DS2							
Reset Value	1	1	1	1	1	1	1	1
0x113	DS3							
Reset Value	1	1	1	1	1	1	1	1

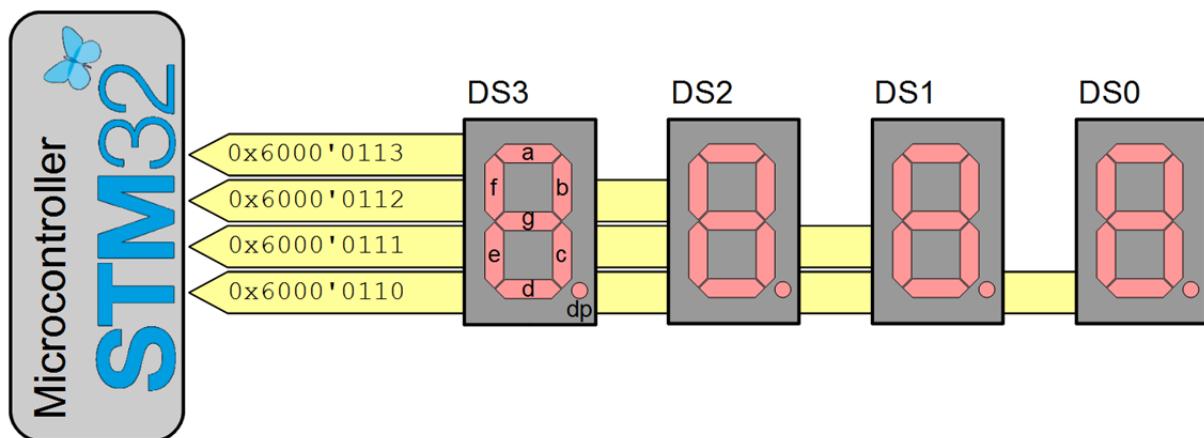


Abbildung 27: 7-Segmentanzeige (1)

Hex Display

Die 7-Segmentanzeigen können auch verwendet werden, um einen 16 Bit Datenwert in hexadezimaler Darstellung anzuzeigen. Um die Anzeige dunkel zu steuern verwenden Sie den Modus „Segment Control“.

Hinweis: Sollen mehr als 16 Bit dargestellt werden, kann auch das LCD Display verwendet werden (siehe Kapitel 10.3). Mit der Datenanzeige des LCD Displays lassen sich insgesamt 128 Bit ausgeben.

Basisadresse: **0x6000'0000**

Zugriffsart: **read / write**

Offset	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x120	Hex-Darstellung auf DS1 / DS0							
Reset Value	0	0	0	0	0	0	0	0
0x121	Hex-Darstellung auf DS3 / DS2							
Reset Value	0	0	0	0	0	0	0	0

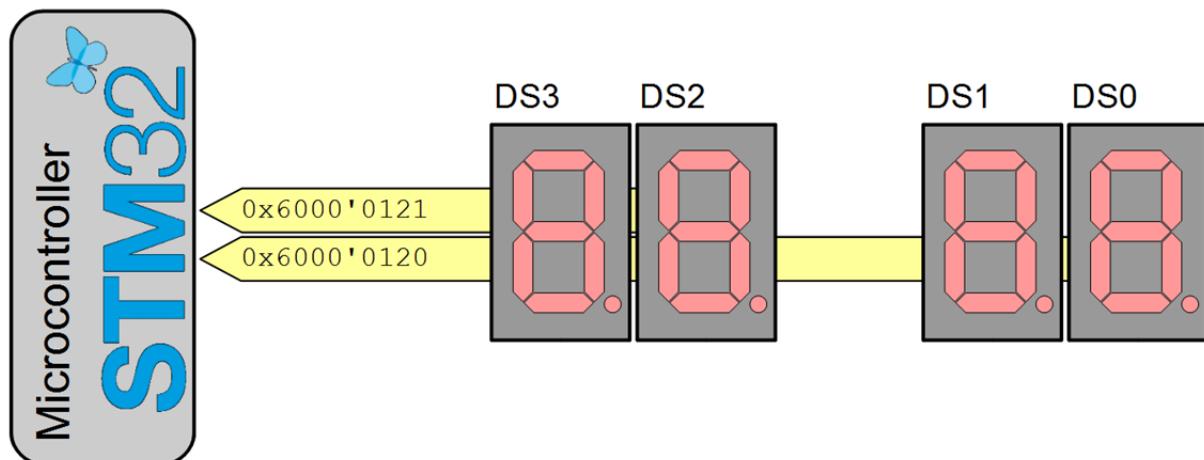


Abbildung 28: 7-Segmentanzeige (2)

10.3 LCD Character Display

Das LCD Character Display kann auf drei Arten benutzt werden:

- Über einen einfachen LCD Controller können beliebige ASCII Zeichen auf dem Display dargestellt werden.
- Das Binär Interface kann maximal 128 Bit in hexadezimaler Form darstellen.
- Das Display kann auch direkt über den I²C Master angesprochen werden. In diesem Fall muss das Protokoll (siehe Datenblatt LCD Display) selbst implementiert werden.

Die Hintergrundbeleuchtung wird über einen 16 Bit PWM Controller gesteuert.

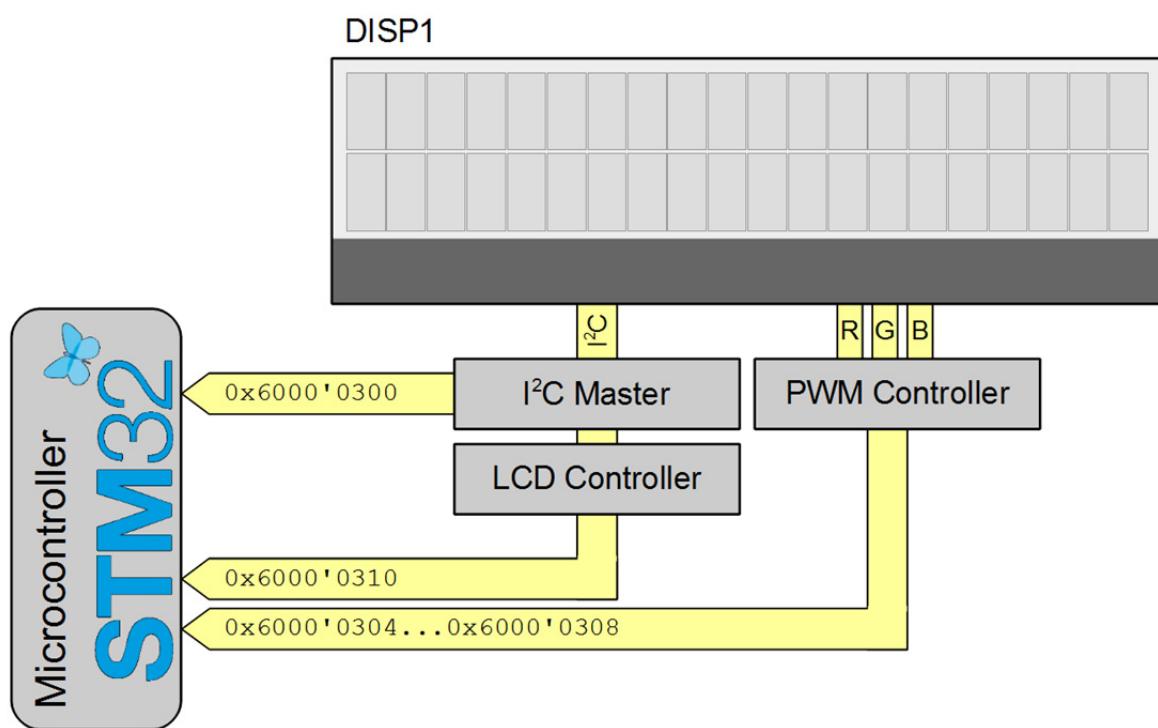
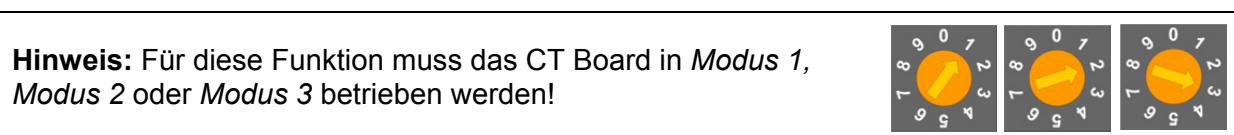


Abbildung 29: LCD Display

ASCII Interface

Die Ansteuerung des Displays kann über den integrierten LCD Controller erfolgen. Zugriffe auf diese Schnittstelle erfolgen zwingend immer mit Half Words (16 Bit Zugriffe).

Basisadresse: **0x6000`0000**

Zugriffsart: read / write

Offset	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x310	ASCII Zeichen							
Reset Value	0	0	0	0	0	0	0	0
0x311	Position auf LCD Display							
Reset Value	0	0	0	0	0	0	0	0

Wurde ein Zeichen an den LCD Controller geschickt, dauert es eine Weile bis die Übertragung abgeschlossen ist. Solange das BSY Bit gesetzt ist (eine logische „1“) kann der LCD Controller kein neues Zeichen entgegen nehmen. Warten Sie solange, bis der Controller das Bit zurücksetzt (eine logische „0“).

Beispiel in C:

```
/* Zeichen 0x61 an Position 01 schreiben */
write_halfword(0x60000310, 0x0161);

/* Auf Abschluss der Übertragung warten */
while(read_halfword(0x60000310) & 0x8000);
```

Pos.:	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f	
Pos.:	10	11	12	13													
	40	41	42	43	44	45	46	47	48	49	4a	4b	4c	4d	4e	4f	50
	51	52	53														

Abbildung 30: LCD Display Positionen

Binär Interface

Über das Binär Interface können auf dem LCD Display Binärdaten in hexa-dezimaler Form angezeigt werden. Es können Zwei mal 32 Bit Binärdaten angezeigt werden.

Hinweis: Bevor erneut auf das Display geschrieben werden kann, muss das BSY Bit abgefragt werden. Vergleiche Beispiel in vorherigem Unterkapitel.

Basisadresse: **0x6000`0000**

Zugriffsart: **read / write**

Offset	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x320	Binärdaten Pos. 0							
Reset Value	0	0	0	0	0	0	0	0
0x321	Binärdaten Pos. 1							
Reset Value	0	0	0	0	0	0	0	0
0x322	Binärdaten Pos. 2							
Reset Value	0	0	0	0	0	0	0	0

...

0x32e	Binärdaten Pos. 14							
Reset Value	0	0	0	0	0	0	0	0
0x32f	Binärdaten Pos. 15							
Reset Value	0	0	0	0	0	0	0	0

	f	f	f	f		f	f	f	f		f	f	f	f
Pos.:	15	14			13	12			11	10			9	8
	f	f	f	f		f	f	f	f		f	f	f	f
Pos.:	7	6			5	4			3	2			1	0

Abbildung 31: Positionen Binär Interface

I²C Master

Über dieses Interface, lässt sich der I²C Master direkt ansteuern. Die Ansteuerung des Displays im Modul utils_ctboard ist über diese Schnittstelle gelöst. In diesem Fall muss das Komplette Protokoll des LCD Displays manuell implementiert werden.

Basisadresse: **0x6000`0000**

Zugriffsart: **read / write**

Offset	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x300 Reset Value	Reserviert					ERR	BSY	RW
	0	0	0	0	0	0	0	0
0x301 Reset Value	Res.	I ² C Adresse						
	0	0	1	1	1	1	0	0
0x302 Reset Value	Data out (from µC)							
	0	0	0	0	0	0	0	0
0x303 Reset Value	Data in (to µC)							
	0	0	0	0	0	0	0	0

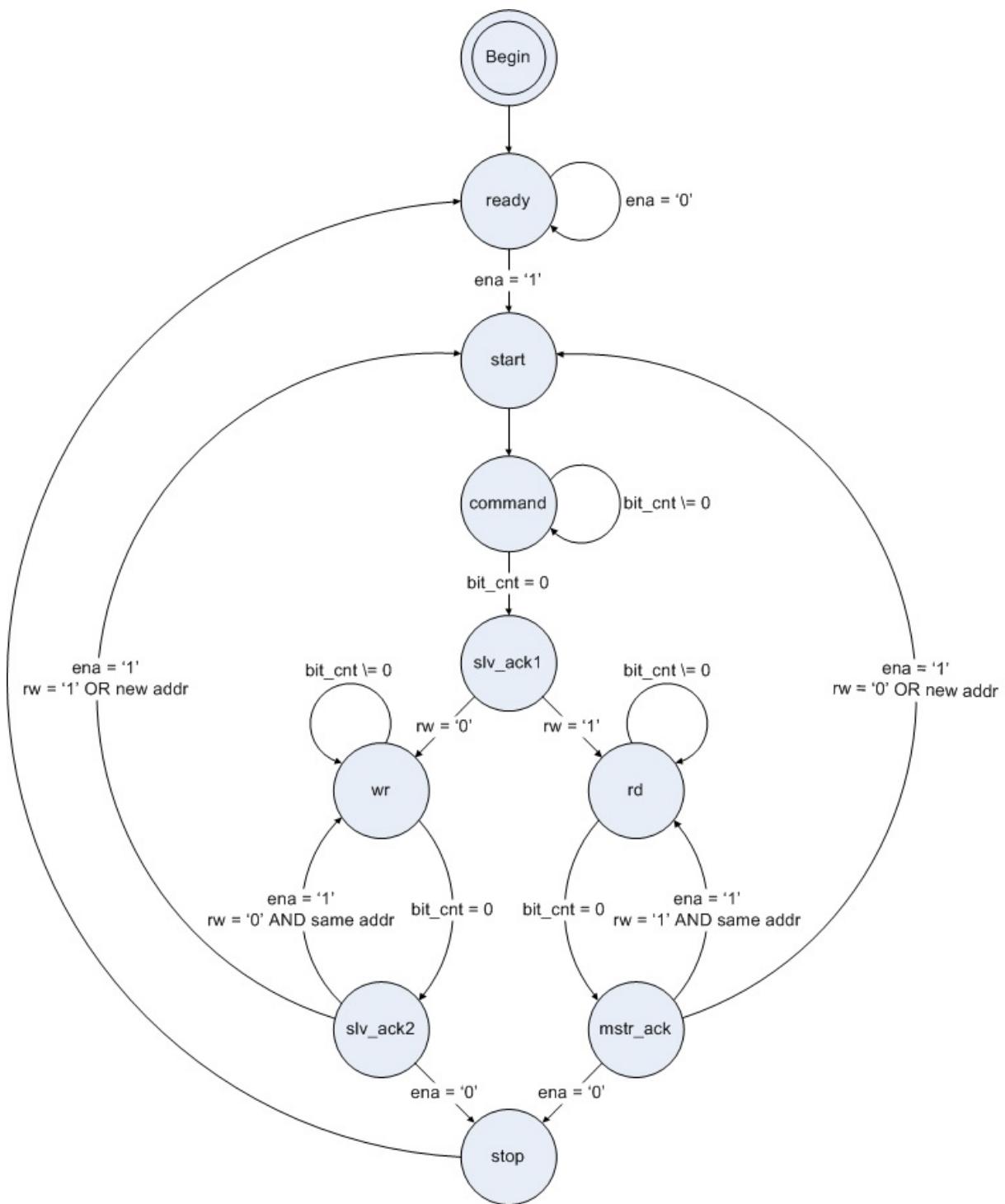
Ansteuerung des I²C Masters

In Abbildung 32 ist die State Machine des I²C Masters abgebildet. Eine mögliche Implementation zur Ansteuerung findet sich am Ende dieses Unterkapitels.

Hinweis: Das Datenblatt des LCD Displays findet sich im Book Tab von Keil µVision.

Im Modul utils_ctboard steht eine Funktion bereit, mit der beliebige Strings an das LCD Display gesendet werden können / das Display gelöscht werden kann:

```
void lcd_write(uint8_t position, char text[]);
void lcd_clear(void);
```

Abbildung 32: State Machine I²C Master

Beispielcode zur Ansteuerung über I²C Master:

```
#define LCD_IS_BUSY (read_byte(0x60000302) & 0x04)

/* Initialisiere Übertragung (Control Byte 1) */
write_byte(0x60000302, 0x80);
write_byte(0x60000300, 0x01);

/* Warten bis I2C Master beginnt */
while(!LCD_IS_BUSY);

/* Sende Character Position (Data Byte 1) */
write_byte(0x60000302, 0x80 | position);
while(!LCD_IS_BUSY); // Übertragung Control Byte 1 fertig
while(LCD_IS_BUSY); // Übertragung Data Byte 1 beginnt

/* Vorbereitung von ASCII Zeichen (Control Byte 2) */
write_byte(0x60000302, 0x40);
while(!LCD_IS_BUSY); // Übertragung Data Byte 1 fertig
while(LCD_IS_BUSY); // Übertragung Control Byte 2 beginnt

/* Übertragung von ASCII Zeichen (Data Byte 2...n) */
write_byte(LCD_ADDR_DATA, text[i]);
while(!LCD_IS_BUSY); // Übertragung Control Byte 2 fertig
while(LCD_IS_BUSY); // Übertragung Data Byte 2 beginnt

/* End transmission */
write_byte(LCD_ADDR_CTRL, LCD_CMD_STOP);
while(!LCD_IS_BUSY); // Übertragung Data Byte 2 fertig
while(LCD_IS_BUSY); // Letzte Übertragung beginnt
```

PWM Controller

Der PWM Controller steuert die Hintergrundbeleuchtung des LCD Displays. Die Helligkeit lässt sich über einen 16 Bit Wert steuern.

Die Leuchtkraft steigt dabei von **0x0000** (0% / aus) bis **0xffff** (100%) linear an. Eine Anpassung an das Helligkeitsempfinden des menschlichen Auges (logarithmische Kennlinie) muss selbst durchgeführt werden.

Basisadresse: **0x6000`0000**

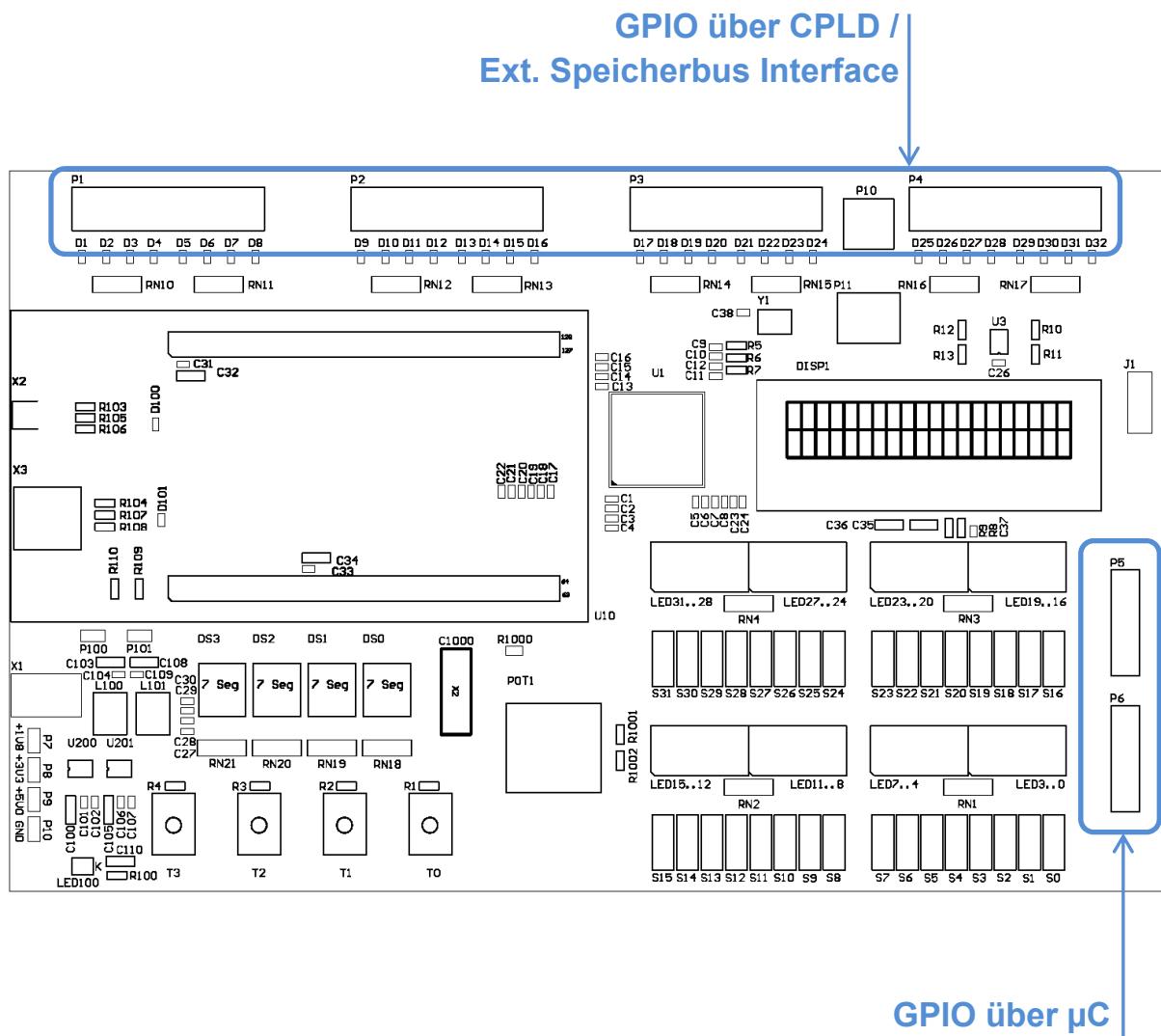
Zugriffsart: read / write

Offset	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x304	PWM Wert Rot Bit 7..0							
Reset Value	0	0	0	0	0	0	0	0
0x305	PWM Wert Rot Bit 15..8							
Reset Value	0	0	0	0	0	0	0	0
0x306	PWM Wert Grün Bit 7..0							
Reset Value	0	0	0	0	0	0	0	0
0x307	PWM Wert Grün Bit 15..8							
Reset Value	0	0	0	0	0	0	0	0
0x308	PWM Wert Blau Bit 7..0							
Reset Value	0	0	0	0	0	0	0	0
0x309	PWM Wert Blau Bit 15..8							
Reset Value	0	0	0	0	0	0	0	0

11 General Purpose Ein- / Ausgabe

Das CT Board stellt folgende General Purpose Funktionen bereit:

- 4x 8 General Purpose Eingänge (Memory Mapped)
- 4x 8 General Purpose Ausgänge (Memory Mapped)
- 2x 16 General Purpose Ein-/Ausgänge direkt über Microcontroller
- Externes Speicherbus Interface



11.1 GPIO über CPLD

Das CT Board verfügt über vier GPIO Schnittstellen mit jeweils 8 Ein- und 8 Ausgabepins. Im Vergleich zu den Mikrocontrollereigenen GPIO Schnittstellen sind diese beträchtlich langsamer, aber für die Laborversuche immer noch schnell genug.

Hinweis: Für diese Funktion muss das CT Board in *Modus 1* betrieben werden!



Die einzelnen Ports sind aufgetrennt in einen Output- und einen Input-Port mit jeweils 8 Bit:

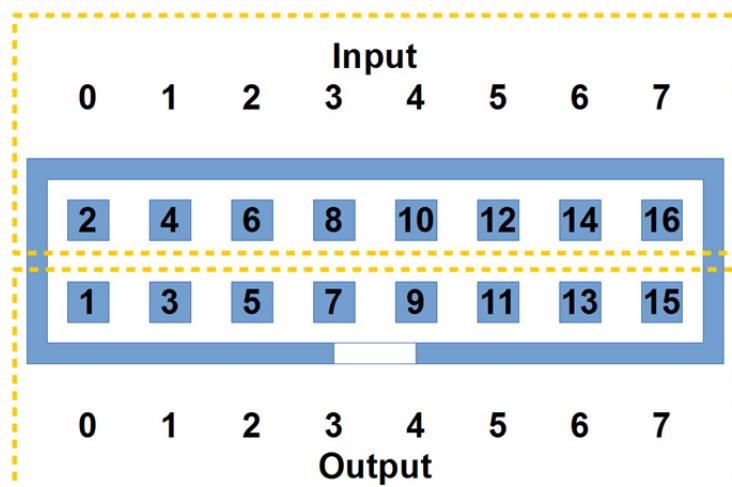


Abbildung 33: GPIO (1)

Output

Basisadresse: **0x6000`0000**

Zugriffsart: read / write

Offset	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x400 Reset Value	P1 Output							
	0	0	0	0	0	0	0	0
0x401 Reset Value	P2 Output							
	0	0	0	0	0	0	0	0
0x402 Reset Value	P3 Output							
	0	0	0	0	0	0	0	0
0x403 Reset Value	P4 Output							
	0	0	0	0	0	0	0	0

Input

Basisadresse: **0x6000`0000**

Zugriffsart: read

Offset	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x410	P1 Input							
	0	0	0	0	0	0	0	0
0x411	P2 Input							
	0	0	0	0	0	0	0	0
0x412	P3 Input							
	0	0	0	0	0	0	0	0
0x413	P4 Input							
	0	0	0	0	0	0	0	0

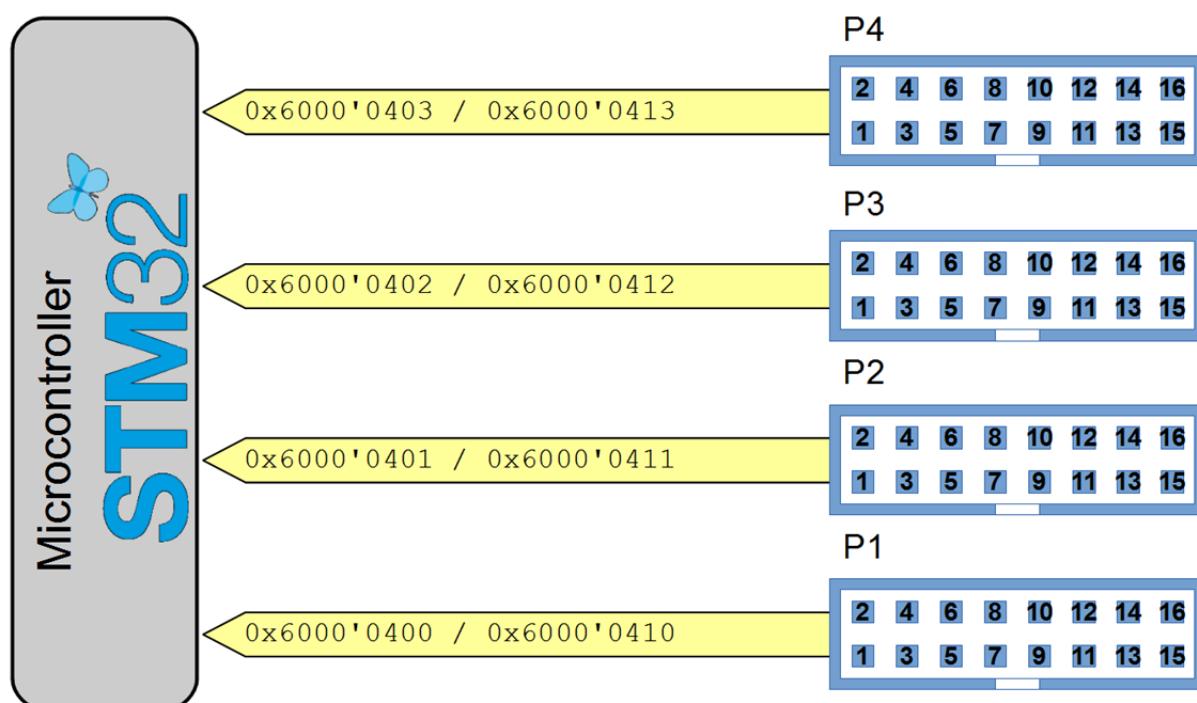


Abbildung 34: GPIO (2)

11.2 GPIO über Mikrocontroller

Diese zwei GPIO Schnittstellen sind direkt mit dem Mikrocontroller verbunden (GPIO Port A bzw. B). Es können also auch alle alternativen Funktionen (UART, ADC, DAC, ...) oder Interrupts die mit diesen zwei Ports verbunden sind (siehe STM32F429 Datenblatt S.71ff) genutzt werden.

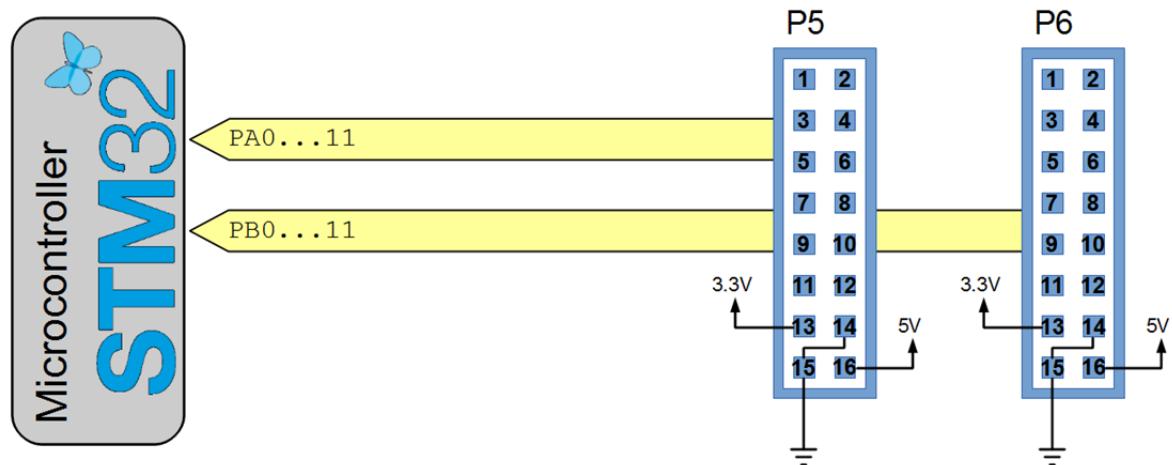
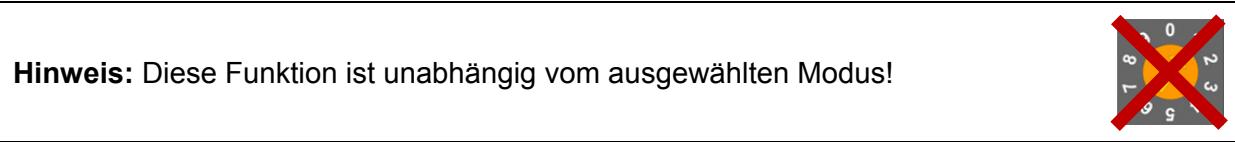


Abbildung 35: GPIO (3)

Port	AF0	AF1	AF2	AF3	AF4	AF5	AF6	AF7	AF8	AF9	AF10	AF11	AF12	AF13	AF14	AF15
SYS	TIM1/2	TIM3/4/5	TIM8/9/ 10/11	I2C1/ 2/3	SPI1/2/ 3/4/5/6	SPI2/3/S AI1	USART6/U ART4/5/7/8	CAN1/2/TIM 12/13/14/ LCD	OTG2_HS/ OTG1_FS	ETH	FMC_SDIO/ OTG2_FS	DCMI	LCD	SYS		
PA0	-	TIM2_ CH1/TIM2_ ETR	TIM5_ CH1	TIM8_ ETR	-	-	USART2_ CTS	UART4_RX	-	-	ETH_MII_ CRS	-	-	-	EVEN_TOUT	
PA1	-	TIM2_ CH2	TIM5_ CH2	-	-	-	USART2_ RTS	UART4_RX	-	-	ETH_MII_ CRS	-	-	-	EVEN_TOUT	
PA2	-	TIM2_ CH3	TIM5_ CH3	TIM9_ CH1	-	-	USART2_ TX	-	-	-	ETH_MDI0	-	-	-	EVEN_TOUT	
PA3	-	TIM2_ CH4	TIM5_ CH4	TIM9_ CH2	-	-	USART2_ RX	-	-	OTG_HS_UPLI_D0	ETH_MII_C0L	-	-	LCD_B5	EVEN_TOUT	
PA4	-	-	-	-	SP13_NSS_12S3_WS	USART2_CK	-	-	-	OTG_HS_SOF	DCMI_HSYNC	LCD_VSYNC	EVEN_TOUT			
PA5	-	TIM2_ CH1/TIM2_ ETR	-	TIM8_CH1	-	SP11_SCK	-	-	OTG_HS_UPLI_CK	-	-	-	-	-	EVEN_TOUT	
PA6	-	TIM1_BKIN	TIM3_CH1	TIM8_BKIN	-	SP11_MISO	-	-	TIM13_CH1	-	-	DCMI_PIXCLK	LCD_G2	EVEN_TOUT		
PA7	-	TIM1_CH1	TIM3_CH2	TIM8_CH1	-	SP11_MOSI	-	-	TIM14_CH1	-	ETH_MII_RX_DV/ ETH_RMII_CRS_DV	-	-	-	EVEN_TOUT	
PA8	MCO1	TIM1_CH1	-	-	I2C3_SCL	-	USART1_CK	-	-	OTG_FS_SOF	-	-	-	LCD_R6	EVEN_TOUT	
PA9	-	TIM1_CH2	-	-	I2C3_SMBA	-	USART1_TX	-	-	-	-	DCMI_D0	-	-	EVEN_TOUT	
PA10	-	TIM1_CH3	-	-	-	-	USART1_RX	-	-	OTG_FS_ID	-	DCMI_D1	-	-	EVEN_TOUT	
PA11	-	TIM1_CH4	-	-	-	-	USART1_CTS	-	CAN1_RX	OTG_FS_DM	-	-	-	LCD_R4	EVEN_TOUT	
PA12	-	TIM1_ETR	-	-	-	-	USART1_RTS	-	CAN1_TX	OTG_FS_DP	-	-	-	LCD_R5	EVEN_TOUT	

Abbildung 36: Alternative Funktionen Port A, B (1)

Port	AF0	AF1	AF2	AF3	AF4	AF5	AF6	AF7	AF8	AF9	AF10	AF11	AF12	AF13	AF14	AF15
	SYS	TIM1/2	TIM3/4/5	TIM8/9/ 10/11	I2C1/ 2/3	SPI1/2/ 3/4/5/6	SPI3/US ART1/2/3	USART6/U ART4/5/7/8	CAN1/2/TIM 12/13/14/ LCD	OTG2_HS /OTG1_ FS	ETH	FMC/Sdio /OTG2_FS	DCMI	LCD	SYS	
PA13	JTMS- SWDI_O	-	-	-	-	-	-	-	-	-	-	-	-	-	-	EVEN TOUT
PA14	JTCK- SWCL_K	-	-	-	-	-	-	-	-	-	-	-	-	-	-	EVEN TOUT
PA15	JTDI	TIM2/ CH1/TIM2_ETR	-	-	SPI1_ NSS/ I2S3_WS	-	-	-	-	LCD_R3	OTG_HS ULPI_D1	ETH_MII/ RXD2	-	-	-	EVEN TOUT
PB0	-	TIM1_ CH2_N	TIM3_ CH3	TIM8_ CH2_N	-	-	-	-	-	LCD_R6	OTG_HS ULPI_D2	ETH_MII/ RXD3	-	-	-	EVEN TOUT
PB1	-	TIM1_ CH3_N	TIM3_ CH4	TIM8_ CH3_N	-	-	-	-	-	-	-	-	-	-	-	EVEN TOUT
PB2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	EVEN TOUT
PB3	JTDO/ TRAC_ESWIO	TIM2_ CH2	-	-	SPI1_ SCK/ I2S3_CK	SPI3_ SCK/ I2S3_CK	-	-	-	-	-	-	-	-	-	EVEN TOUT
PB4	NJTR_ST	-	TIM3_ CH1	-	SPI1_ MISO	I2S3ext_ SD	-	-	-	-	-	-	-	-	-	EVEN TOUT
PB5	-	-	TIM3_ CH2	-	I2C1_ SMBA	SPI1_ MOSI/ I2S3_SD	-	-	CAN2_RX	OTG_HS ULPI_D7	ETH_PPS _OUT	FMC_SDCKE1	DCMI_D10_	-	-	EVEN TOUT
PB6	-	-	TIM4_ CH1	-	I2C1_ SCL	-	-	USART1_ TX	-	CAN2_TX	-	FMC_SDNE1	DCMI_D5_	-	-	EVEN TOUT
PB7	-	-	TIM4_ CH2	-	I2C1_ SDA	-	-	USART1_ RX	-	-	-	FMC_NL	DCMI_VSYNC	-	-	EVEN TOUT
PB8	-	-	TIM4_ CH3	TIM10_ CH1	I2C1_ SCL	-	-	-	CAN1_RX	-	ETH_MII/ TXD3	SDIO_D4	DCMI_D6_	LCD_B6	EVEN TOUT	
PB9	-	-	TIM4_ CH4	TIM11_ CH1	I2C1_ SDA	SPI2_ NSS/I2 S2_WS	-	-	CAN1_TX	-	-	SDIO_D5	DCMI_D7	LCD_B7	EVEN TOUT	
PB10	-	TIM2_ CH3	-	-	I2C2_ SCL	SPI2_ SCK/I2 S2_CK	-	USART3_ TX	-	OTG_HS ULPI_D3	ETH_MII/ RX_ER	-	-	LCD_G4	EVEN TOUT	

Abbildung 37: Alternative Funktionen Port A, B (2)

	AF0	AF1	AF2	AF3	AF4	AF5	AF6	AF7	AF8	AF9	AF10	AF11	AF12	AF13	AF14	AF15
Port	SYS	TIM1/2	TIM3/4/5	TIM8/9/ 10/11	I2C1/ 2/3	SPI1/2/ 3/4/5/6	SPI2/3/ AI1	USART6/U ART1/2/3	USART7/U ART4/5/7/8	CAN1/2/TIM 12/13/14/ LCD	OTG2_HS/ OTG1_FS	ETH	FMC/SDIO /OTG2_FS	DCMI	LCD	SYS
PB11	-	TIM2_ CH4	-	-	I2C2_ SDA	-	-	USART3_ RX	-	-	OTG_HS_ ULPI_D4_	ETH_MII/ TX_EN/ ETH_RMII/ TX_EN	-	-	LCD_G5	EVEN TOUT
PB12	-	TIM1_ BKIN	-	-	I2C2_ SMB	SPI2/ NSS/I2 S2_WS	-	USART3_ CK	-	CAN2_RX	OTG_HS_ ULPI_D5_	ETH_MII/ TXD0/IEETH _RMII/ TXD0	OTG_HS_ID	-	-	EVEN TOUT
Port B	PB13	-	TIM1_ CH1N	-	-	SPI2/ SCK/I2 S2_CK	-	USART3_ CTS	-	CAN2_TX	OTG_HS_ ULPI_D6_	ETH_MII/ TXD1/IEETH _RMII/ TX_D1	-	-	-	EVEN TOUT
PB14	-	TIM1_ CH2N	-	-	TIM8_ CH2N	-	SPI2/ MISO	I2S2ext_ SD	USART3_ RTS	-	TIM12_CH1	-	-	OTG_HS_DM	-	-
PB15	RTC REFIN	TI	M1 CH3N	-	TIM8_ CH3N	-	SPI2/ MOSI/I2 S2_SD	-	-	-	TIM12_CH2	-	-	OTG_HS_DP	-	-

Abbildung 38: Alternative Funktionen Port A, B (3)

11.3 Externes Speicherbus Interface

Über die Anschlüsse P1...P4 kann externer Speicher angeschlossen werden.

Hinweis: Für diese Funktion muss das CT Board in <i>Modus 2</i> betrieben werden!	
--	--

Pinbelegung Stecker P1 bis P4:

Pin 2	Pin 4	Pin 6	Pin 8	Pin 10	Pin 12	Pin 14	Pin 16	P1
A17	A19	A21	A23	A25	0	0	1	
Pin 1	Pin 3	Pin 5	Pin 7	Pin 9	Pin 11	Pin 13	Pin 15	P2
A16	A18	A20	A22	A24	0	1	0	

Pin 2	Pin 4	Pin 6	Pin 8	Pin 10	Pin 12	Pin 14	Pin 16	P2
A1	A3	A5	A7	A9	A11	A13	A15	
Pin 1	Pin 3	Pin 5	Pin 7	Pin 9	Pin 11	Pin 13	Pin 15	P3
A0	A2	A4	A6	A8	A10	A12	A14	

Pin 2	Pin 4	Pin 6	Pin 8	Pin 10	Pin 12	Pin 14	Pin 16	P3
D1	D3	D5	D7	D9	D11	D13	D15	
Pin 1	Pin 3	Pin 5	Pin 7	Pin 9	Pin 11	Pin 13	Pin 15	P4
D0	D2	D4	D6	D8	D10	D12	D14	

Pin 2	Pin 4	Pin 6	Pin 8	Pin 10	Pin 12	Pin 14	Pin 16	P4
NE2	NE4	NWE	NBU	1	1	1	CLK	
Pin 1	Pin 3	Pin 5	Pin 7	Pin 9	Pin 11	Pin 13	Pin 15	P4
NE1	NE3	NOE	NBL	1	1	1	NRST	

11.4 Externes Speicherbus Interface (zur Anzeige am Oszilloskop)

Über die Anschlüsse P1...P4 kann externer Speicher angeschlossen werden.

Hinweis: Für diese Funktion muss das CT Board in <i>Modus 3</i> betrieben werden!	
--	---

Pinbelegung Stecker P1 bis P4:

Pin 2	Pin 4	Pin 6	Pin 8	Pin 10	Pin 12	Pin 14	Pin 16	P1	
GND									
Pin 1	Pin 3	Pin 5	Pin 7	Pin 9	Pin 11	Pin 13	Pin 15		
D15	D14	D13	D12	D11	D10	D9	D8		

Pin 2	Pin 4	Pin 6	Pin 8	Pin 10	Pin 12	Pin 14	Pin 16	P2	
GND									
Pin 1	Pin 3	Pin 5	Pin 7	Pin 9	Pin 11	Pin 13	Pin 15		
D7	D6	D5	D4	D3	D2	D1	D0		

Pin 2	Pin 4	Pin 6	Pin 8	Pin 10	Pin 12	Pin 14	Pin 16	P3	
GND									
Pin 1	Pin 3	Pin 5	Pin 7	Pin 9	Pin 11	Pin 13	Pin 15		
A7	A6	A5	A4	A3	A2	A1	A0		

Pin 2	Pin 4	Pin 6	Pin 8	Pin 10	Pin 12	Pin 14	Pin 16	P4	
GND									
Pin 1	Pin 3	Pin 5	Pin 7	Pin 9	Pin 11	Pin 13	Pin 15		
A10	A9	A8	NOE	NWE	NBL	NBU	CLK		