

## Inhaltsverzeichnis

1	Aufgabenstellung .....	3
1.1	Beschreibung der Aufgabenstellung .....	3
1.2	Verwendung von Informationen aus dem Internet .....	3
1.3	Schaltplan z.B. mit KiCad .....	7
1.4	Source Code Control .....	7
1.5	Programmentwurf und Dokumentation .....	7
1.6	Quellcode.....	9
1.6.1	KISS .....	9
1.6.2	Benennung der Variablen ähnlich der ungarischen Konvention .....	9
1.6.3	Header von Funktionen .....	10
1.6.4	Versionsnummer im Quellcode.....	10
1.7	Lessons-Learned .....	11
1.8	Doku des Ergebnisses, .....	12
2	Musterlösung – Notizen zum Thema „Inbetriebnahme des I2C 1602 LCD Displays“ .....	13
2.1	Aufgabenstellung.....	13
2.2	Informationssammlung .....	14
2.2.1	Informationen zum Display .....	14
2.2.2	Treiber .....	15
2.3	Schaltplan .....	17
2.4	Setup der SFRs (PxSEL, PxSEL2) zur Verwendung von Pins für den Anschluß des I2C-Displays.....	17
2.5	Probleme bei der IBN der Hardware .....	19
2.5.1	Kontrasteinstellung des LCD-Displays .....	19
2.5.2	Fehlende Pullup Widerstände an SCL und SDA.....	19
2.5.3	LaunchPad: Jumper an P1.6 .....	20
2.5.4	FOC .....	20
2.6	Struktogramm.....	21
2.7	Quellcode.....	23
2.8	Speed enhancement.....	27
2.9	Fotogalerie .....	28
2.10	Lessons learned.....	29
2.10.1	Probleme bei der IBN des Displays .....	29
2.10.2	Verwendung des Struktogrammeditors.....	29

2.10.3	Verwendung von CodeComposerStudio .....	29
2.10.4	Source Code Control.....	30

# 1 Aufgabenstellung

## 1.1 Beschreibung der Aufgabenstellung

Beschreiben Sie die Aufgabenstellung mit tabellarischer Übersicht

Aufgabe	Notwendig MSP	Notwendig STM
Einlesen eines analogen Werts in eine Variable	X	X
Darstellung des Werts auf dem LCD-Display	X	X
Wenn ein Schwellwert unterschritten wird, sollen die gelbe und die weiße LED leuchten	X	X
Programmablaufsteuerung: Polling, Software		X
Programmablaufsteuerung: ISR, Hardware	X	

Aufgabe	Beispielhafte Aufgabenstellung	Status
A	Wenn der Schwellwert wieder überschritten wird, soll die gelbe LED sofort ausgehen. Die weiße LED soll noch eine bestimmte Zeit (5 s) nachleuchten.	
A	Die restliche Nachleuchtzeit soll auf dem Display angezeigt werden	
B	Wenn es dunkel wird, soll das Fenster durch den Servo geschlossen werden	
B	Wenn es hell wird, soll das Fenster durch den Servo geöffnet werden	

### 1. Aufgabe bis 18. Dezember 2024:

Erstellen Sie ein individuelles Anforderungsdokument (Lastenheft, d.h. Weiterführen der obigen Tabelle). Diese Tabelle ist sehr dynamisch: Fügen Sie Anforderungen, die Sie erst während der Programmerstellung erkennen unten in die Tabelle ein.

Erstellen Sie ein Dokument, in dem sie Ihre Vorgehensweise dokumentieren.

**Wichtig:** Dieses Dokument soll Ihre Arbeit **skizzenhaft** dokumentieren. Sie brauchen keine Zeit verschwenden, um das Dokument besonders schön zu gestalten. Es soll **kurz, schlicht** und **zweckmäßig** sein.

## 1.2 Verwendung von Informationen aus dem Internet

Verwenden Sie das Internet um Informationen zur Aufgabenstellung zu finden.

Verwenden Sie auch eine AI wie ChatGPT.

Dokumentieren Sie Ihre Erfahrungen.

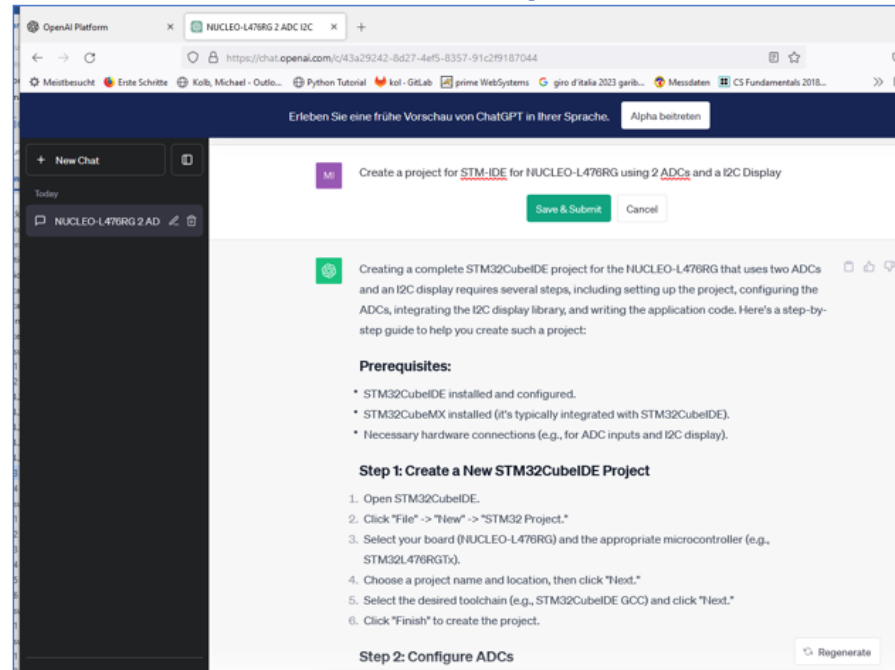
Beispiel:

21.7.4 2023-10-04 ChatGPT

21.7.4.1 *Anfrage an ChatGPT*

“Create a project for STM-IDE for NUCLEO-L476RG using 2 ADCs and a I2C Display”

21.7.4.2 *Antwort von ChatGPT mit meinen Erweiterungen*



Creating a complete STM32CubeIDE project for the NUCLEO-L476RG that uses two ADCs and an I2C display requires several steps, including setting up the project, configuring the ADCs, integrating the I2C display library, and writing the application code. Here's a step-by-step guide to help you create such a project:

```
### Prerequisites:
- STM32CubeIDE installed and configured.
- STM32CubeMX installed (it's typically integrated with STM32CubeIDE).
- Necessary hardware connections (e.g., for ADC inputs and I2C display).
```

```
### Step 1: Create a New STM32CubeIDE Project
```

1. Open STM32CubeIDE.
2. Click "File" -> "New" -> "STM32 Project."
3. Select your board (NUCLEO-L476RG) and the appropriate microcontroller (e.g., STM32L476RGTx).
4. Choose a project name and location, then click "Next."
5. Select the desired toolchain (e.g., STM32CubeIDE GCC) and click "Next."

#### MEINE BEOBACHTUNGEN:

Der Quellcode muß noch ein bisschen angepasst werden:

- Die Namen der ADC `Init()`-Funktionen stimmen nicht mit denen von Chat-GPT überein.
- Die Namen der Treiberfunktionen für das Display müssen angepasst werden.
- Im `main` von ChatGPT wird keine `I2C Init()`-Funktion aufgerufen

```
void SystemClock_Config(void);

int main(void) {
    HAL_Init();
    SystemClock_Config();

    HAL_ADC_MspInit(&hadc1);
    HAL_ADC_MspInit(&hadc2);

    HAL_ADC_Init(&hadc1);
    HAL_ADC_Init(&hadc2);

    SSD1306_Init(); // Initialize I2C OLED display

    uint16_t adc1_value = 0;
    uint16_t adc2_value = 0;

    while (1) {
        HAL_ADC_Start(&hadc1);
        HAL_ADC_PollForConversion(&hadc1, HAL_MAX_DELAY);
        adc1_value = HAL_ADC_GetValue(&hadc1);

        HAL_ADC_Start(&hadc2);
        HAL_ADC_PollForConversion(&hadc2, HAL_MAX_DELAY);
        adc2_value = HAL_ADC_GetValue(&hadc2);

        char buffer[20];
        sprintf(buffer, "ADC1: %4d ", adc1_value);
        SSD1306_GotoXY(0, 0);
        SSD1306_Puts(buffer, &Font_7x10, SSD1306_COLOR_WHITE);
    }
}
```

```
102
103 /* Initialize all configured peripherals */
104 MX_GPIO_Init();
105 MX_USART2_UART_Init();
106 MX_ADC1_Init();
107 MX_ADC2_Init();
108 MX_I2C1_Init();
109 /* USER CODE BEGIN 2 */
```

```
149 // USER CODE BEGIN 3
150 }
151 /* IICED CODE END 3 */
```

#### ### Step 7: Build and Flash

1. Build your project in STM32CubeIDE.
2. Connect your NUCLEO-L476RG board to your computer via USB.
3. Click the "Run" menu to flash the code to your microcontroller.

This is a basic example to get you started. You may need to customize it further based on your specific hardware and requirements. Additionally, ensure that you've correctly set up the I2C display library and made the necessary modifications to it to match your display module.

Make sure to refer to the datasheets and reference manuals of your microcontroller, ADCs, and I2C display for detailed information on pin configurations, registers, and functions.

#### 21.7.4.3 Durchführung und Ergebnis

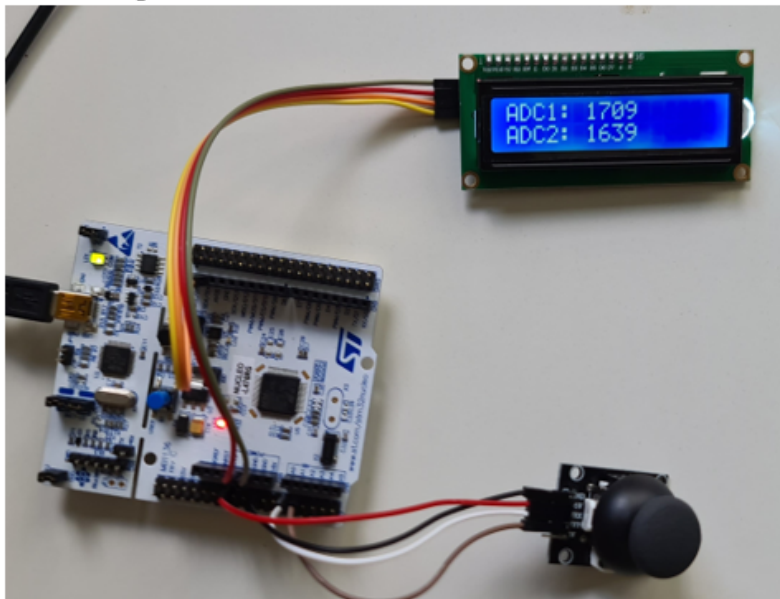
Name des Projekts: CG\_01-01\_Voltmeter\_with\_LCD-Display

J:\ RO\03 STM32\STM32CubeIDEWorkspaces\ADC\220123\_ADC\CG\_01-01\_Voltmeter\_with\_LCD-Display

FOC.

Zeitraum von der ChatGPT Nachfrage bis zum fertigen Programm: Ca. 1,5 h

Es ist zu beachten, daß ich das Projekt mit ausreichend Vorwissen und bereits angepassten Treibern begonnen habe.

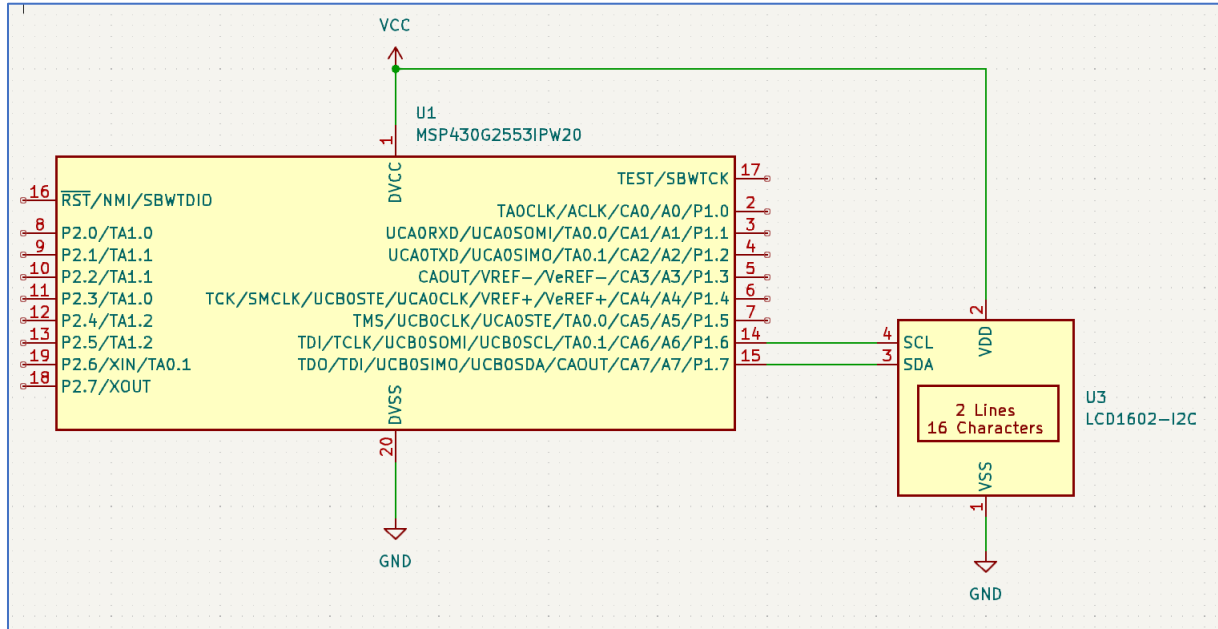


J:\ RO\03 STM32\Doku\EigeneBilder\CG\_01-01 ADC mit Display 20231004\_small.jpg

### 1.3 Schaltplan z.B. mit KiCad

Erstellen Sie einen Schaltplan zu Ihrer Lösung.

Das Symbol für das LCD-Display habe ich selbst erzeugt. Das war nicht besonders aufwändig.



### 1.4 Source Code Control

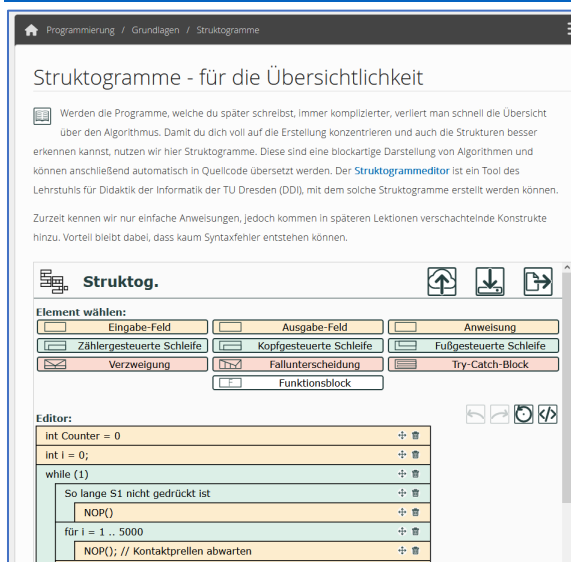
Sparen Sie Nerven und Stress und verwenden Sie ein SCC Programm wie SVN oder GIT.

Legen Sie dort nicht nur den Quellcode ab, sondern auch die Dokumentation und im Projekt verwendete Datenblätter.

### 1.5 Programmwurf und Dokumentation

Die Programmwurf und die -dokumentation soll graphisch erfolgen.

Ein Tool hierzu findet man z. B. bei <https://dditools.inf.tu-dresden.de/ovk/Informatik/Programmierung/Grundlagen/Struktogramme.html>



draw.io für UML / PAP / Struktogramme

[https://de.wikipedia.org/wiki/Unified\\_Modeling\\_Language#Interaktionen](https://de.wikipedia.org/wiki/Unified_Modeling_Language#Interaktionen)



## 1.6 Quellcode

### 1.6.1 KISS

Denken Sie beim Programmieren bitte stets an die wichtigste Regel zur Softwareerstellung, den **KISS** Grundsatz:

**Keep it simple and stupid.**

Weil kompliziert wird es dann eh von allein, oder um es mit Wilhelm Busch zu sagen:  
"Stets findet Überraschung statt da, wo man's nicht erwartet hat."

### 1.6.2 Benennung der Variablen ähnlich der ungarischen Konvention

Die Variablen sollen angelehnt an die ungarische Konvention benannt werden

[https://de.wikipedia.org/wiki/Ungarische\\_Notation](https://de.wikipedia.org/wiki/Ungarische_Notation)

Einfache aber hilfreiche Konventionen  
Prefixe für verschiedene Datentypen

⊕ Modifier

Prefix	C - Typ	Beschreibung
G		Globale Variable
p	Zeiger	
sz	char[]	Zero terminated string
A	Array	Array von Variablen

Datentypen

Prefix	C - Typ	Beschreibung
b	selbst definiert	boolsche Variable (FALSE oder TRUE)
c	char	8 bit character ANSI
u8		unsigned 8-Bit Zahl
u16		unsigned 16-Bit Zahl
u32		unsigned 32-Bit Zahl
s8		signed 8-Bit Zahl
s16		signed 16-Bit Zahl
s32		signed 32-Bit Zahl
f32	float	32-Bit Fließkommazahl
f64	double	64-Bit Fließkommazahl
e	enum	Aufzählungsdattentyp
v	void	
h	handle	Zusammengesetzter Datentyp unbekannten Inhalts
fp	FILE*	Filepointer

Beispiele

```
bSchalterGeschlossen // Kann wahr oder falsch sein
u16AnzahlDerTomaten  // Unsigned Integer
szUeberschrift        // String mit Stringendezeichen '\0'
f32Messwert           // 32 Bit (float) Fließkommazahl
pf64Messwert          // Zeiger auf 64Bit (double) Fließkommazahl
Af32Messwerte         // Array von float Variablen
```

Defines

Ausdrücke nach #define werden immer großgeschrieben

Beispiel:

```
#define MAXIMUM_SPEED 200
```

Namen von Funktionen

Anfangsbuchstabe großschreiben

Beispiel:

```
int BerechneAbstandZurBar (float *pf32Durststrecke)
```

### 1.6.3 Header von Funktionen

In den Headern bei Funktionen ähnlich nach einen Schema einfügen

Schönes Beispielprojekt von Olimex:

```
69  /*
70  Name:
71      Buzzer_Buzz
72  Input:
73      Time - period of buzzing
74      Freq - delaying between toggling the outputs
75  Output:
76      nothing
77  Description:
78      Simple toggling the buzzer's outputs with delay between them
79  */
80  void Buzzer_Buzz (unsigned long int Time, unsigned int Freq)
81  {
82      while (Time-->0)
83      {
84          P1OUT = P1OUT ^ (BUZZER_OUT0_PIN | BUZZER_OUT1_PIN);
85          Delay (Freq);
86      }
87      return;
88  }
```

Wer will, kann sich auch am Standard von Doxygen orientieren:

<https://www.doxygen.nl/manual/docblocks.html>

### 1.6.4 Versionsnummer im Quellcode

Das kann den besten Programmierer passieren: Man sucht mit dem Debugger nach einem Fehler im Programm, und merkt dabei gar nicht, daß man nicht die aktuelle Version des Programms auf den Controller geladen hat.

Um hier Linderung zu schaffen, hat es sich bewährt, im Quellcode ein **#define** mit der aktuellen Version zu unterhalten:

```

MK-KS_main_05_01.c  *MK-KS_main_05_02.c x  stdio.h  lcd.c  i2c.c  lcd.h
1  /*****
2  * Aufgabe: Ausgabe eines AD-Werts an ein LCD-Display.
3  *
4  * Author: Michael.Kolb@th-rosenheim.de
5  * Version:
6  * 231212a - Ausgabe an LCD - FOC
7  * 231212b - Einlesen eines ADs (polling) - FOC
8  * 231212c - Wenn AI > Schwellwert --> LEDS ON - FOC
9  * 231213a - Wenn AI < Schwellwert --> Timer LED White via Polling - FOC
10 * 231213b - Wenn AI < Schwellwert --> Timer LED White via Timer - FOC
11 * 231215a - Abhilfe für Problem: LCD schaut beim Update schrecklich aus - ONGOING
12 *
13 *****/
14
15 #define VERSION "V_2023-12-15a"
16
17 #include "msp430.h"
18 #include "lcd.h"
19 #include "stdio.h"
20 #include "msp430g2553.h"
21 #include <intrinsics.h> // intrinsische Funktionen
22

```

Diese Versionsnummer kann dann z. B. am Programmstart ausgegeben werden:

```

81
82 // Ausgabe des Willkommensbildschirms
83 LcdSetPosition(1,1);
84 LcdWriteString("Analog In");
85 LcdSetPosition(2,1);
86 LcdWriteString(VERSION);
87

```



## 1.7 Lessons-Learned

Good judgement comes from experience.

Experience comes from bad judgement.

„Der Kluge lernt aus allem und von jedem, der Normale aus seinen Erfahrungen und der Dumme weiß alles besser.“ (Sokrates)

Was ist besonders gut gelaufen und sollte in weiteren Projekten wieder verwendet werden?  
Was ist schlecht gelaufen, wo muss in Zukunft mit einer anderen Strategie gearbeitet werden

### 1.8 Doku des Ergebnisses,

Fotografien, Screenshots vom Oszi, evtl. ein kleiner Film der Anwendung

Abzugeben sind:

Bericht (max. 20 Seiten, siehe Musterlösung in Kap.2), entweder im Learning Campus oder per E-Mail oder im Postkasten der Hochschule im Foyer (Im Falle der E-Mail bestätigt der Prüfer den Erhalt.)

Boards: MSP430 und Breakout-Boards bis zum **22. Januar 2024**.

Haus-Bausätze, STM32 bis zum **4. April 2025**

Bewertungskriterien für den Bericht

Item	Punkte
Äußere Form (d.h. Gliederung, Literaturverzeichnis, Einleitung, Zusammenfassung, 2 Punkte)	2
Dokumentation der Aufgabenstellung (2 Punkte)	2
Schaltplan (2 Punkte)	2
Programmablaufpläne (2 Punkte)	2
Programmierstil (Kommentare, Unterprogramme, 4 Punkte)	4
Dokumentation des Ergebnissen mit Hilfe eines Screenshots oder Video	2

(Die Punkte beziehen sich jeweils auf Umfang (d.h. angemessener Umfang) und Inhalt (d.h. richtiger, sinnvoller Inhalt)

Summe: 14 Punkte

**Termin für die Abgabe des Berichts: 31. März 2025**

## 2 Musterlösung – Notizen zum Thema „Inbetriebnahme des I2C 1602 LCD Displays“

### 2.1 Aufgabenstellung

Das 1602LCD Display soll am LaunchPad EXP430G2 in Betrieb genommen werden.

#	Aufgabe	Status
1	Informationen Sammeln: Wie funktioniert das Display? Siehe <a href="#">2.2</a> .	231215
2	Finden von Treibern für das Display. Siehe <a href="#">2.2.2</a> .	231215
3	Anschlußbelegung finden und dokumentieren	231215
4	Ausgabe eines Zählers auf dem Display	231215
4	Ausgabe von verschiedenen Zeichen auf dem Display	231218
5	Testen von Funktionen wie: Cursor On/Off, Backlight on/off, Display on/off	231219 ONGOING
6	Weiter geht's im Kapitel <a href="#">2.7</a> .	231218
7	Fotogalerie. Siehe <a href="#">2.9</a> .	ToDo 231219
8	Lessons learned. Siehe <a href="#">2.10</a> .	ToDo 231219
9	Speicher dieser Datei und der CCS-Projekts im Learning Campus	ToDo 231218
10	Programm in Unterfunktionen aufteilen. Beispiel in <a href="#">1.6.3</a> einfügen.	ToDo 231218
11		
12		

## 2.2 Informationssammlung

### 2.2.1 Informationen zum Display

#### 2.2.1.1 Digikey

<https://www.digikey.de/de/products/detail/sunfounder/CN0295D/18668612>



[www.sunfounder.com](http://www.sunfounder.com)

### I2C LCD1602 Module

#### Introduction

As we all know, though LCD and some other displays greatly enrich the man-machine interaction, they share a common weakness. When they are connected to a controller, multiple IOs will be occupied of the controller which has no so many outer ports. Also it restricts other functions of the controller. Therefore, LCD1602 with an I2C bus is developed to solve the problem.

I2C bus is a type of serial bus invented by PHILIPS. It is a high performance serial bus which has bus ruling and high or low speed device synchronization function required by multiple host system. I2C bus has only two bidirectional signal lines, Serial Data Line (SDA) and Serial Clock Line (SCL). The blue potentiometer on the I2C LCD1602 is used to adjust backlight to make it easier to display on the I2C LCD1602.



- **GND**: Ground
- **VCC**: Voltage supply, 5V.
- **SDA**: Serial data line. Connect to VCC through a pull\_up resistor.
- **SCL**: Serial clock line. Connect to VCC through a pull\_up resistor.

#### I2C Address

The default address is basically 0x27, in a few cases it may be 0x3F.

Quellenangabe:

<J:\ RO\07 MSP430\LCD I2C\CN0295D.pdf>

#### 2.2.1.2 Allaboutcircuits

**Sehr gute Seite**, bei der man fast alles Notwendige findet:

<https://forum.allaboutcircuits.com/ubs/msp430-i2c-lcd1602-interface.1822/>

#### 2.2.1.3 Weitere Datenblätter

<J:\ RO\07 MSP430\LCD I2C\COM-LCD16X2 ANLEITUNG 2022-03-07.pdf>

<J:\ RO\07 MSP430\LCD I2C\COM-LCD16X2 DATENBLATT 2022-03-07.pdf>

<J:\ RO\07 MSP430\LCD I2C\TC1602A-01T.pdf>

#### 2.2.1.4 YouTube Filme

How to Program MSP430 using Code Composer Studio to Interface 16x2 LCD Display:

<https://www.youtube.com/watch?v=QNRDj04RnJE>

## 2.2.2 Treiber

### 2.2.2.1 Github (nicht verwendet)

[https://github.com/Sanjeev7/LCD\\_I2C\\_MSP430F2553](https://github.com/Sanjeev7/LCD_I2C_MSP430F2553)

```
LCD      GPIO PIN ON CONTROLLER/BOARD
VCC --> 5V
GND --> GND
SDA --> P1.7
SCL --> P1.6
```

- Usage:

- Copy and paste LCD\_I2C.H and LCD\_I2C.C at the directory of your main function.
- An example program is provided to illustrate the available functions.

The available functions are:

- void lcd\_init(Slave Address); //Initialize the LCD with its I2C Address as per the datasheet.
- void lcd\_print(String); //Print the String on the LCD.
- void lcd\_setcursor(row,column); //Place the cursor to the values defined as per the arguments "row" and "column".
- void set\_backlight(); //Set backlight of the LCD by software. If backlight is not required, simply don't use this function :-).

[J:\RO\07 MSP430\LCD\\_I2C\LCD\\_I2C\\_MSP430F2553-main](J:\RO\07 MSP430\LCD_I2C\LCD_I2C_MSP430F2553-main)

**231206:** Wegen ungelöster Probleme bei der IBN wurde diese Lösungsmöglichkeit nicht weiterverfolgt

### 2.2.2.2 Github (von mir verwendet)

<https://github.com/andrei-cb/I2C-Display-and-MSP430>

Product Solutions Open Source Pricing

Search or jump to...

andrei-cb / I2C-Display-and-MSP430 Public

Code Issues Pull requests Actions Projects Security Insights

master 1 branch 0 tags

Go to file Code

Local Codespaces

Clone ?

HTTPS GitHub CLI

<https://github.com/andrei-cb/I2C-Display-and-MSP430>

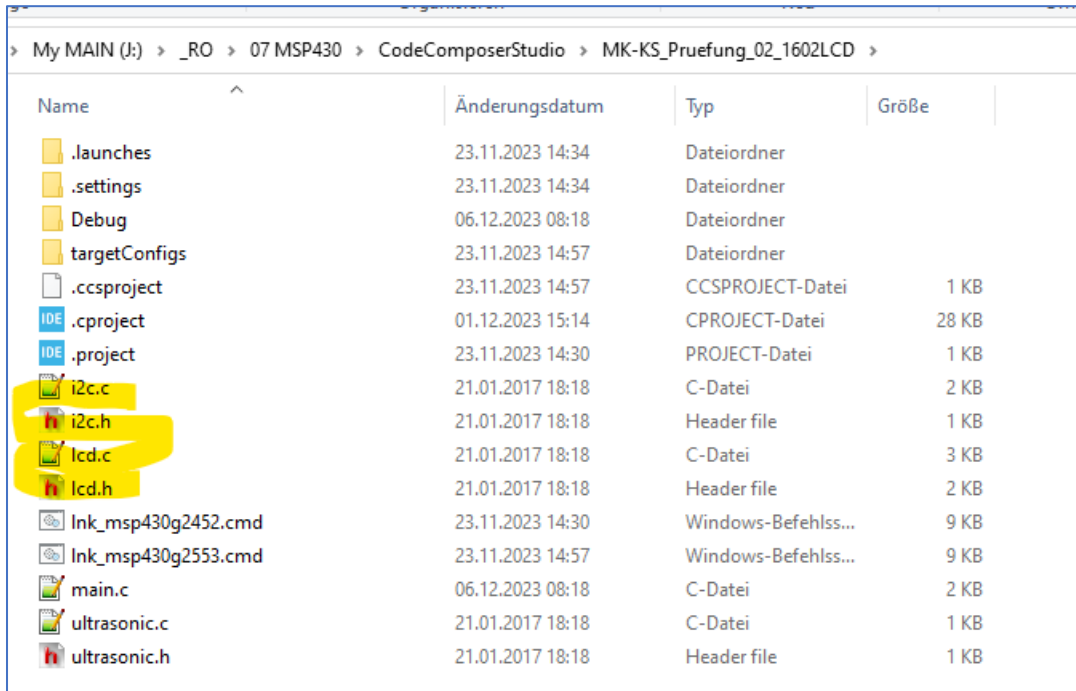
Use Git or checkout with SVN using the web URL.

Open with GitHub Desktop

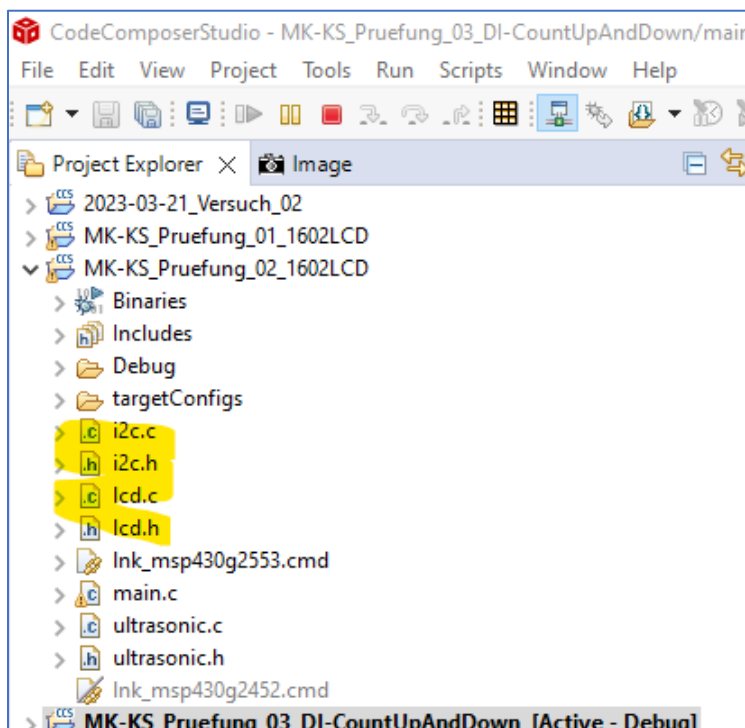
andrei-cb updated comments

.gitignore	multiple fixes
LICENSE	first commit
README.md	first commit
i2c.c	improved comments
i2c.h	multiple fixes

## Einbinden der Treiberdateien:



Name	Änderungsdatum	Typ	Größe
.launches	23.11.2023 14:34	Dateiordner	
.settings	23.11.2023 14:34	Dateiordner	
Debug	06.12.2023 08:18	Dateiordner	
targetConfigs	23.11.2023 14:57	Dateiordner	
.ccsproject	23.11.2023 14:57	CCSPROJECT-Datei	1 KB
.cproject	01.12.2023 15:14	CPROJECT-Datei	28 KB
.project	23.11.2023 14:30	PROJECT-Datei	1 KB
i2c.c	21.01.2017 18:18	C-Datei	2 KB
i2c.h	21.01.2017 18:18	Header file	1 KB
lcd.c	21.01.2017 18:18	C-Datei	3 KB
lcd.h	21.01.2017 18:18	Header file	2 KB
Ink_msp430g2452.cmd	23.11.2023 14:30	Windows-Befehlss...	9 KB
Ink_msp430g2553.cmd	23.11.2023 14:57	Windows-Befehlss...	9 KB
main.c	06.12.2023 08:18	C-Datei	2 KB
ultrasonic.c	21.01.2017 18:18	C-Datei	1 KB
ultrasonic.h	21.01.2017 18:18	Header file	1 KB

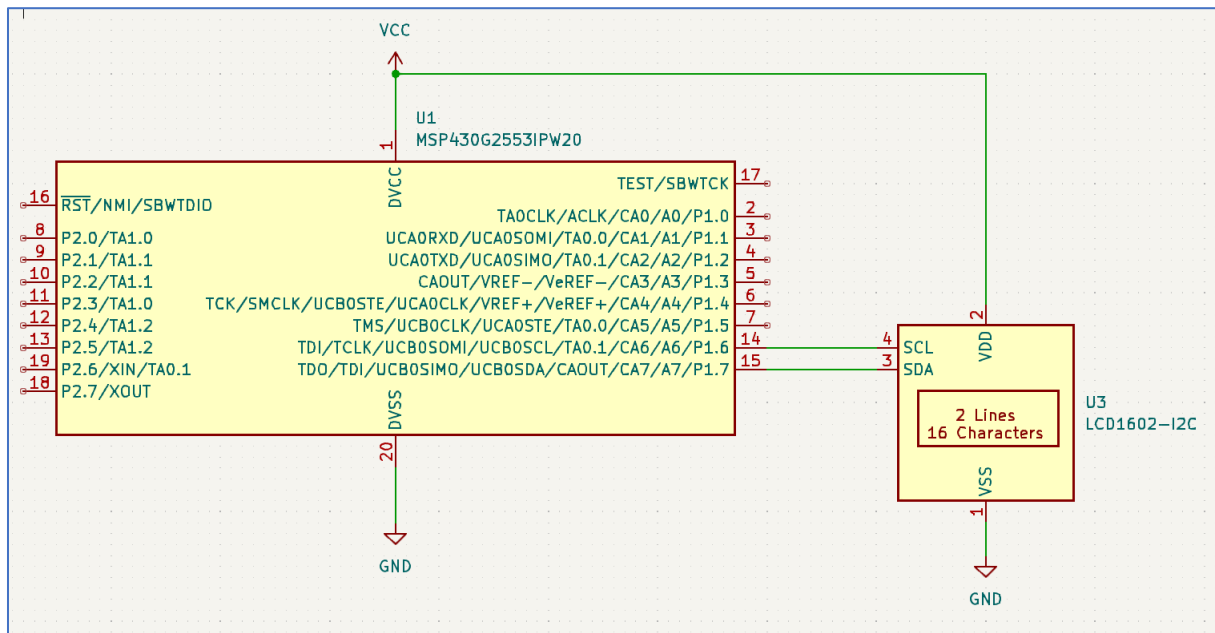


## Achtung:

Die Ausgabe an das LCD-Display war brutal langsam. Deshalb habe ich den Code analysiert. Im Kommando `LcdWriteCommand()` habe ich eine lange Warteschleife entdeckt. Ich habe die Zeit verkürzt. Siehe hierzu [2.8](#).



## 2.3 Schaltplan



[J:\ RO\09 Keystudio Smart Home\KiCad Dateien\231215 16x2 LCD-Display\231215a 16x2 LCD-Display\231215a 16x2 LCD-Display.kicad\\_pro](J:\ RO\09 Keystudio Smart Home\KiCad Dateien\231215 16x2 LCD-Display\231215a 16x2 LCD-Display\231215a 16x2 LCD-Display.kicad_pro)

## 2.4 Setup der SFRs (PxSEL, PxSEL2) zur Verwendung von Pins für den Anschluß des I2C-Displays

[slau144k.pdf](#)

### 8.2.5 Function Select Registers PxSEL and PxSEL2

Port pins are often multiplexed with other peripheral module functions. See the device-specific data sheet to determine pin functions. Each PxSEL and PxSEL2 bit is used to select the pin function - I/O port or peripheral module function.

Table 8-1. PxSEL and PxSEL2

PxSEL2	PxSEL	Pin Function
0	0	I/O function is selected.
0	1	Primary peripheral module function is selected.
1	0	Reserved. See device-specific data sheet.
1	1	Secondary peripheral module function is selected.

Setting PxSELx = 1 does not automatically set the pin direction. Other peripheral module functions may require the PxDIR bits to be configured according to the direction needed for the module function. See the pin schematics in the device-specific data sheet.

#### Note

##### Setting PxREN = 1 When PxSEL = 1

On some I/O ports on the MSP430F261x and MSP430F2416/7/8/9, enabling the pullup/pulldown resistor (PxREN = 1) while the module function is selected (PxSEL = 1) does not disable the logic output driver. This combination is not recommended and may result in unwanted current flow through the internal resistor. See the device-specific data sheet pin schematics for more information.

**Table 19. Port P1 (P1.5 to P1.7) Pin Functions**

PIN NAME (P1.x)	x	FUNCTION	CONTROL BITS AND SIGNALS <sup>(1)</sup>					
			P1DIR.x	P1SEL.x	P1SEL2.x	ADC10AE.x INCH.x=1 <sup>(2)</sup>	JTAG Mode	CAPD.y
P1.5/ TA0.0/ UCB0CLK/ UCA0STE/ A5 <sup>(2)</sup> / CA5 TMS Pin Osc	5	P1.x (I/O)	I: 0; O: 1	0	0	0	0	0
TA0.0		1	1	0	0	0	0	
UCB0CLK		from USCI	1	1	0	0	0	
UCA0STE		from USCI	1	1	0	0	0	
A5		X	X	X	1 (y = 5)	0	0	
CA5		X	X	X	0	0	1 (y = 5)	
TMS		X	X	X	0	1	0	
Capacitive sensing		X	0	1	0	0	0	
P1.6/ TA0.1/ UCB0SOMI/ UCB0SCL/ A6 <sup>(2)</sup> / CA6 TDI/TCLK/ Pin Osc	6	P1.x (I/O)	I: 0; O: 1	0	0	0	0	0
TA0.1		1	1	0	0	0	0	
UCB0SOMI		from USCI	1	1	0	0	0	
UCB0SCL		from USCI	1	1	0	0	0	
A6		X	X	X	1 (y = 6)	0	0	
CA6		X	X	X	0	0	1 (y = 6)	
TDI/TCLK		X	X	X	0	1	0	
Capacitive sensing		X	0	1	0	0	0	
P1.7/ UCB0SIMO/ UCB0SDA/ A7 <sup>(2)</sup> / CA7 CAOUT TDO/TDI/ Pin Osc	7	P1.x (I/O)	I: 0; O: 1	0	0	0	0	0
UCB0SIMO		from USCI	1	1	0	0	0	
UCB0SDA		from USCI	1	1	0	0	0	
A7		X	X	X	1 (y = 7)	0	0	
CA7		X	X	X	0	0	1 (y = 7)	
CAOUT		1	1	0	0	0	0	
TDO/TDI		X	X	X	0	1	0	
Capacitive sensing		X	0	1	0	0	0	

Device-specific data sheet: [J:\\\_RO\07 MSP430\Datasheets\msp430g2553.pdf](J:\_RO\07 MSP430\Datasheets\msp430g2553.pdf)

## 2.5 Probleme bei der IBN der Hardware

### Problembeschreibung:

Wenn das LCD-Display direkt an das LaunchPad angeschlossen wurde, konnten keine Spannungen an den Pins P1.6 und P1.7 gemessen werden.

Wenn die Testplatinen aus dem Praktikum angeschlossen wurden, wurden Spannungen ausgegeben.

Nach intensiver Fehlersuche, wurden drei Fehlerquellen entdeckt und beseitigt:

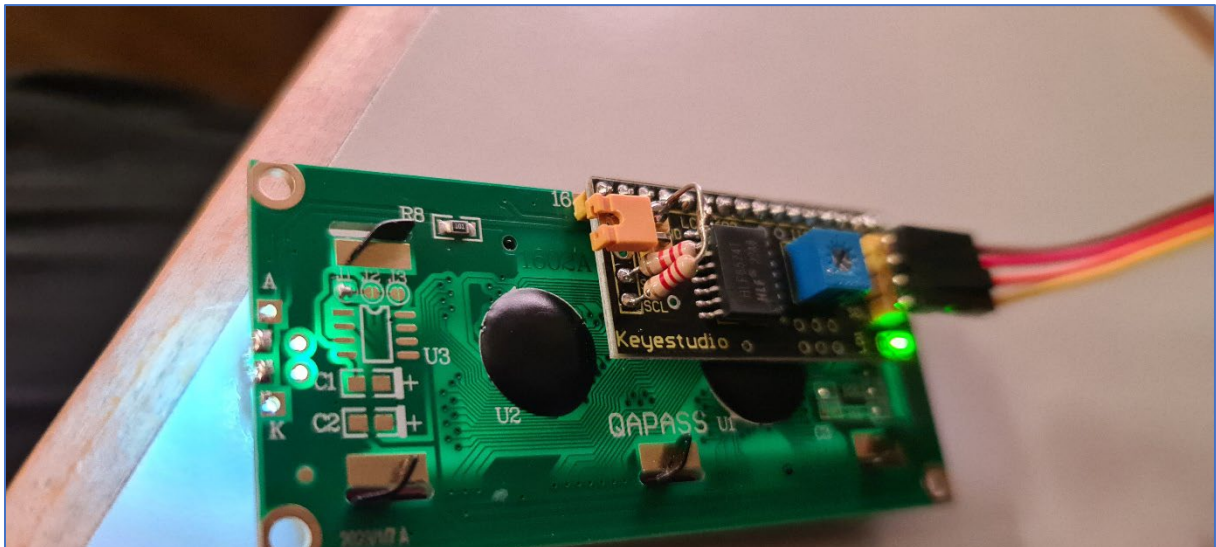
### 2.5.1 Kontrasteinstellung des LCD-Displays

Der Kontrast des LCD-Displays muß über das blaue Poti an seiner Rückseite korrekt eingestellt werden.

### 2.5.2 Fehlende Pullup Widerstände an SCL und SDA

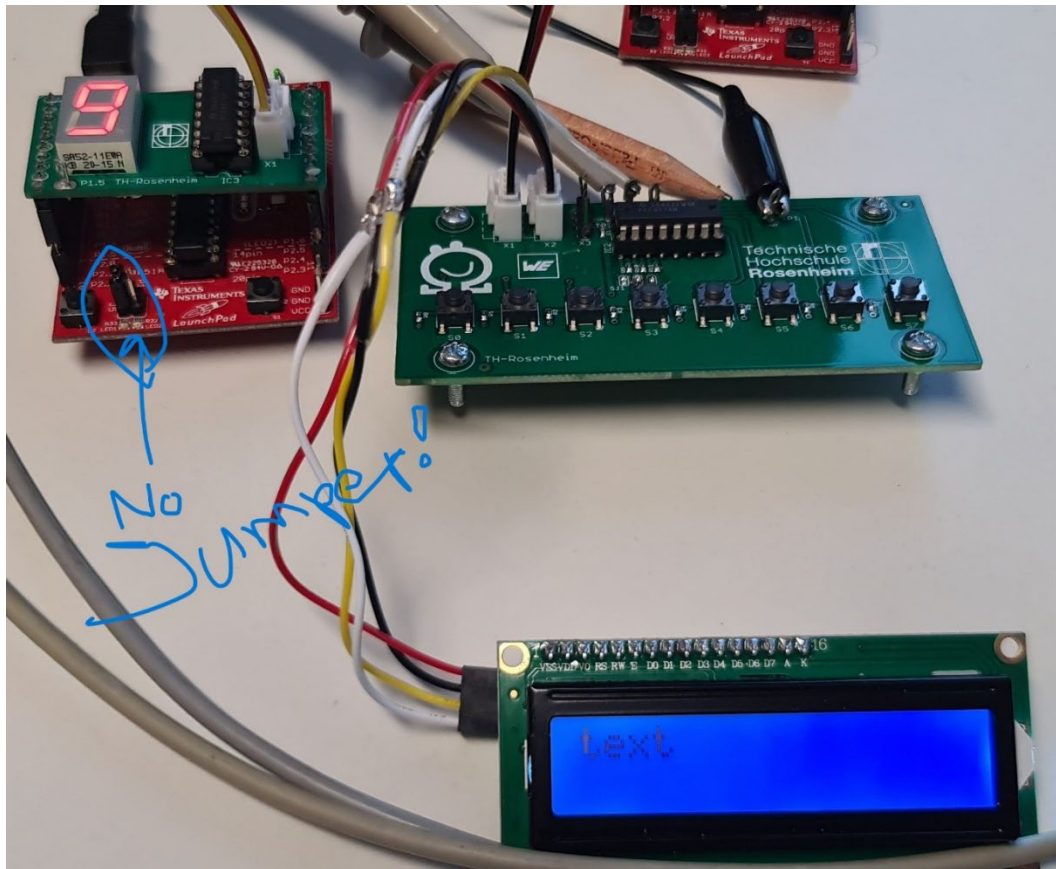
Enhancement der LCD-Platinen, um sie direkt an das LaunchPad anschließen zu können:

Je ein Pullup Widerstand mit 33kOhm an SCL und SDA:



### 2.5.3 LaunchPad: Jumper an P1.6

Auf dem LaunchPad muß der Jumper an P1.6 entfernt werden







### 2.5.4 FOC











Jetzt werden die Daten anständig übertragen und das Display funktioniert.











## 2.6 Struktogramm





































**Struktog.**

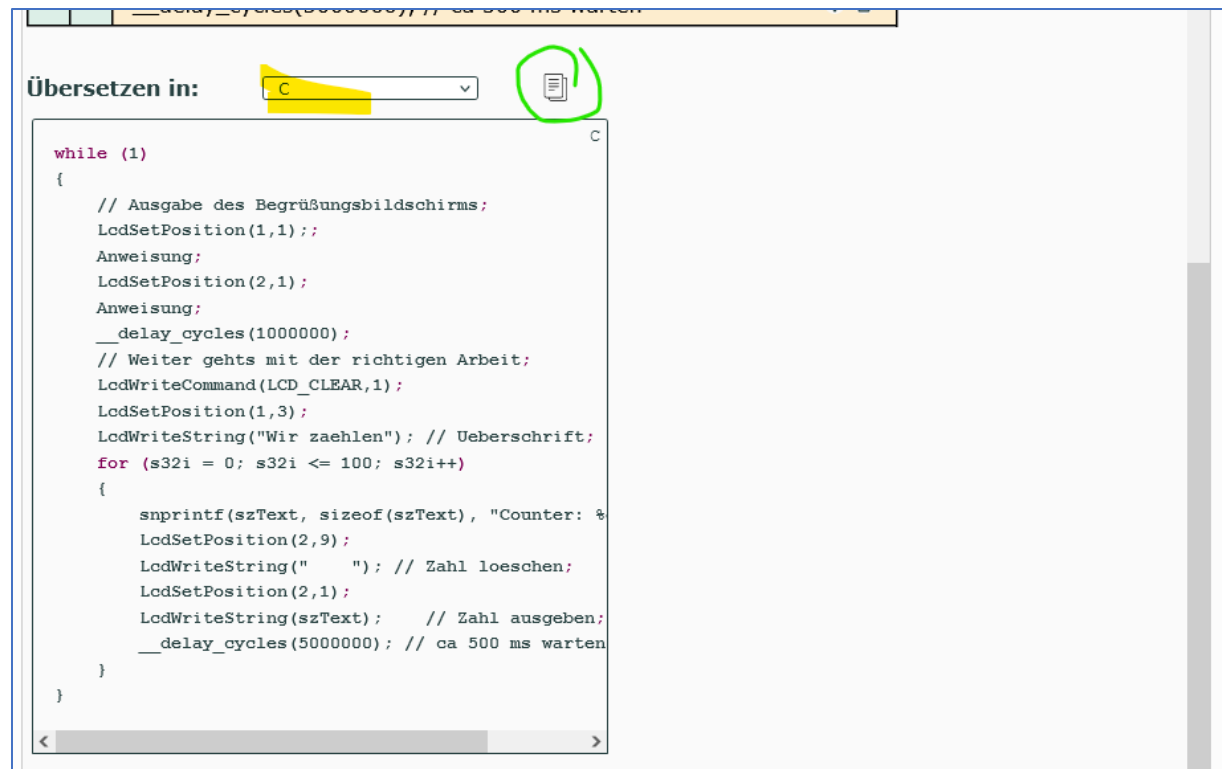
**Element wählen:**

 Eingabe-Feld	 Ausgabe-Feld	 Anweisung
 Zählergesteuerte Schleife	 Kopfgesteuerte Schleife	 Fußgesteuerte Schleife
 Verzweigung	 Fallunterscheidung	 Try-Catch-Block
 Funktionsblock		

**Editor:**

**Editor:**

1		 
	// Ausgabe des Begrüßungsbildschirms	 
	LcdSetPosition(1,1);	 
	Anweisung	 
	LcdSetPosition(2,1)	 
	Anweisung	 
	__delay_cycles(1000000)	 
	// Weiter gehts mit der richtigen Arbeit	 
	LcdWriteCommand(LCD_CLEAR,1)	 
	LcdSetPosition(1,3)	 
	LcdWriteString("Wir zaehlen"); // Ueberschrift	 
	s32i = 0; s32i <= 100; s32i++	 
	sprintf(szText, sizeof(szText), "Counter: %d", s32i)	 
	LcdSetPosition(2,9)	 
	LcdWriteString(" "); // Zahl loeschen	 
	LcdSetPosition(2,1)	 
	LcdWriteString(szText); // Zahl ausgeben	 
	__delay_cycles(5000000); // ca 500 ms warten	 



J:\ RO\09 Keystudio Smart Home\Struktogramme\LCD-Display\_001\_2023-12-15.json



## 2.7 Quellcode

J:\\_RO\07 MSP430\CodeComposerStudio\MK-KS Pruefung 02 1602LCD

```
main.c x i2c.c
1 /*****
2 * Aufgabe: Inbetriebnahme des 1602LCD-Displays.
3 *
4 * Author: Michael.Kolb@th-rosenheim.de
5 * Version:
6 * 231212a - Ausgabe an LCD - FOC
7 * 231212b - Ausgabe der Zahlen von 0 .. 200 - FOC
8 * 231218a - LCD ON/OFF - FOC
9 *
10 *****/
11
12 #define VERSION "V 2023-12-18a"
13
14 #include "lcd.h"
15 #include "stdio.h"
16 #include "msp430g2553.h"
17
18 int main()
19 {
20     char szText[17]; // Puffer für die Textausgabe
21     signed int s32i; // Counter
22
23     WDTCTL = WDTPW + WDTHOLD; // Stop watchdog
24     _EINT(); // enable interrupts (wird in i2c.c benötigt)
25
26     LcdInit(); // initialize LCD display
27
28     while (1)
29     {
30         // Ausgabe des Begrüßungsbildschirms
31         LcdSetPosition(1,1);
32         LcdWriteString(" LCD Demo");
33         LcdSetPosition(2,1);
34         LcdWriteString(VERSION);
35         delay_cycles(1000000);
36
37         // Weiter gehts mit der richtigen Arbeit
38         // LCD aus und wieder einschalten
39         LcdWriteCommand(LCD_CLEAR,1);
40         LcdSetPosition(1,3);
41         LcdWriteString("LCD_OFF"); // Ueberschrift
42         delay_cycles(250000); // ca 250 ms warten
43         LcdWriteCommand(LCD_OFF,1);
44         delay_cycles(1500000); // ca 1500 ms warten
45         LcdSetPosition(1,3);
46         LcdWriteString("LCD_ON "); // Ueberschrift
47         delay_cycles(250000); // ca 250 ms warten
48         LcdWriteCommand(LCD_ON,1);
49         delay_cycles(1500000); // ca 1500 ms warten
50         // Zahlenausgabe
51         LcdWriteCommand(LCD_CLEAR,1);
52         LcdSetPosition(1,3);
53         LcdWriteString("Wir zaehlen"); // Ueberschrift
54
55         for (s32i = 0; s32i <= 100; s32i++)
56         {
57             snprintf(szText, sizeof(szText), "Counter: %d", s32i);
58             LcdSetPosition(2,9);
59             LcdWriteString(" "); // Zahl loeschen
60             LcdSetPosition(2,1);
61             LcdWriteString(szText); // Zahl ausgeben
62             delay_cycles(500000); // ca 500 ms warten
63         }
64     }
65     return 0;
66 }
```

```
main.c  lcd.c  i2c.c  isr_trap.asm  lcd.h  i2c.h X
1 #ifndef I2C_H
2 #define I2C_H
3
4 #define SDA_PIN BIT7
5 #define SCL_PIN BIT6
6 #define PRESCALE 12
7
8 void I2cTransmitInit(unsigned char slaveAddress);
9 void I2cTransmit(unsigned char slaveAddress, unsigned char byte);
10
11 unsigned char I2cNotReady();
12
13 #endif
14
```

```
main.c  lcd.c  i2c.c X  isr_trap.asm
1 #include "i2c.h"
2 #include "msp430g2553.h"
3
4 unsigned char byteToTransmit;
5 unsigned char byteSent = 0;
6
7 void I2cTransmitInit(unsigned char slaveAddress)
8 {
9     P1SEL    |= SDA_PIN + SCL_PIN;           // Assign I2C pins to USCIB0
10    P1SEL2    |= SDA_PIN + SCL_PIN;
11    UCB0CTL1  = UCSWRST;                       // Enable SW reset
12    UCB0CTL0  = UCMST + UCMODE_3 + UCSYNC;      // I2C Master, synchronous mode
13    UCB0CTL1  = UCSSEL_2 + UCSWRST;           // Use SMCLK, keep SW reset
14    UCB0BR0   = PRESCALE;                     // Set prescaler - SMCLK = ~1048Khz/12 = 87.3Khz
15    UCB0BR1   = 0;
16    UCB0I2CSA = slaveAddress;                 // Set slave address
17    UCB0CTL1  &= ~UCSWRST;                     // Clear SW reset, resume operation
18    UCB0I2CIE = UCNACKIE;                     // Interrupt on slave NACK
19    IE2       = UCB0TXIE;                     // Enable TX interrupt
20 }
21
22 void I2cTransmit(unsigned char slaveAddress, unsigned char byte)
23 {
24     I2cTransmitInit(slaveAddress);
25     byteToTransmit = byte;
26     byteSent = 0;
27     UCB0CTL1 |= UCTR + UCTXSTT; // Generate start condition
28 }
29
30 unsigned char I2cNotReady()
31 {
32     return (UCB0STAT & UCBBUSY); // Check if I2C bus is busy
33 }
34
35 #pragma vector = USCIB0TX_VECTOR
36 __interrupt void USCIB0TX_ISR(void)
37 {
38     if (!(IFG2 & UCB0RXIFG))
39     {
40         if (byteSent == 1) // If the byte was sent...
41         {
42             UCB0CTL1 |= UCTXSTP; // Generate stop condition
43             IFG2 &= ~UCB0TXIFG;
44         }
45         else
46         {
47             UCB0TXBUF = byteToTransmit; // Send the byte
48             byteSent = 1; // Modify the variable accordingly
49         }
50     }
51 }
52
```



```

main.c  lcd.c  i2c.c  isr_trap.asm
1 #include "i2c.h"
2 #include "lcd.h"
3 #include "msp430g2553.h"
4
5 void LcdInit()
6 {
7     LcdWriteCommand(LCD_INIT_BYTE, 0);
8     __delay_cycles(80000);
9     LcdWriteCommand(LCD_INIT_BYTE, 0);
10    LcdWriteCommand(LCD_INIT_BYTE, 0);
11
12    LcdWriteCommand(LCD_BUS_WIDTH_4BIT, 0);
13    LcdWriteCommand(LCD_4BITS_2LINES_5x8FONT, 1);
14    LcdWriteCommand(LCD_DISPLAY_OFF_CURSOR_OFF_BLINK_OFF, 1);
15    LcdWriteCommand(LCD_CLEAR, 1);
16    LcdWriteCommand(LCD_INCREMENT_NO_SHIFT, 1);
17    LcdWriteCommand(LCD_DISPLAY_ON_CURSOR_OFF, 1);
18 }
19
20 void LcdWriteCommand(unsigned char data, unsigned char cmdtype)
21 {
22     unsigned char byte;
23
24     byte = (HI_NIBBLE(data) | LCD_BL | LCD_EN;
25     I2cTransmit(0x27, byte);
26     while(I2cNotReady());
27
28     byte = (HI_NIBBLE(data) | LCD_BL) & ~LCD_EN;
29     I2cTransmit(0x27, byte);
30     while(I2cNotReady());
31
32     // cmdtype = 0 -> One write cycle
33     // cmdtype = 1 -> Two write cycles (4 bit mode)
34
35     if (cmdtype)
36     {
37         byte = (LO_NIBBLE(data) | LCD_BL | LCD_EN;
38         I2cTransmit(0x27, byte);
39         while(I2cNotReady());
40         byte = (LO_NIBBLE(data) | LCD_BL) & ~LCD_EN;
41         I2cTransmit(0x27, byte);
42         while(I2cNotReady());
43     }
44
45     __delay_cycles(80000);
46 }
47 void LcdWriteChar(unsigned char data)
48 {
49     unsigned char byte;
50
51     byte = (HI_NIBBLE(data) | LCD_BL | LCD_RS | LCD_EN;
52     I2cTransmit(0x27, byte);
53     while(I2cNotReady());
54
55     byte = (HI_NIBBLE(data) | LCD_BL | LCD_RS) & ~LCD_EN;
56     I2cTransmit(0x27, byte);
57     while(I2cNotReady());
58
59     byte = (LO_NIBBLE(data) | LCD_BL | LCD_RS | LCD_EN;
60     I2cTransmit(0x27, byte);
61     while(I2cNotReady());
62
63     byte = (LO_NIBBLE(data) | LCD_BL | LCD_RS) & ~LCD_EN;
64     I2cTransmit(0x27, byte);
65     while(I2cNotReady());
66 }
67
68 void LcdWriteString(char *s)
69 {
70     while (*s != '\0')
71         LcdWriteChar(*s++);
72 }
73 void LcdSetPosition(unsigned char row, unsigned char column)
74 {
75     switch(row)
76     {
77     case 1:
78         LcdWriteCommand(LCD_LINE1 + (column - 1), 1);
79         break;
80     case 2:
81         LcdWriteCommand(LCD_LINE2 + (column - 1), 1);
82         break;
83     case 3:
84         LcdWriteCommand(LCD_LINE3 + (column - 1), 1);
85         break;
86     case 4:
87         LcdWriteCommand(LCD_LINE4 + (column - 1), 1);
88         break;
89     default:
90         LcdWriteCommand(LCD_LINE1 + (column - 1), 1);
91     }
92 }
93

```

```
main.c  lcd.c  i2c.c  isr_trap.asm  lcd.h X
1 #ifndef LCD_H
2 #define LCD_H
3
4
5 /* bits 0-3 -> control signals
6 * b0 -> RS (reset)
7 * b1 -> RW (read/write)
8 * b2 -> EN (enable)
9 * b3 -> BL (backlight)
10 * bits 4-7 -> data
11 */
12
13 #define LCD_WAIT_DELAY 2
14 #define LCD_BL 0x08
15 #define LCD_EN 0x04
16 #define LCD_RW 0x02
17 #define LCD_RS 0x01
18
19 #define LCD_INIT_BYTE          0x30 // 0011 0000
20 #define LCD_BUS_WIDTH_4BIT    0x20 // 0010 0000
21 #define LCD_BUS_WIDTH_8BIT    0x30 // 0011 0000
22 #define LCD_CLEAR              0x01 // 0000 0001
23 #define LCD_HOME               0x02 // 0000 0010
24 #define LCD_ON                 0x0C // 0000 1100
25 #define LCD_OFF                0x08 // 0000 1000
26 #define LCD_LINE1             0x80 // 1000 0000
27 #define LCD_LINE2             0xC0 // 1100 0000
28 #define LCD_LINE3             0x94 // 1001 0100
29 #define LCD_LINE4             0xD4 // 1101 0100
30 #define LCD_CURSOR_OFF        0x0C // 0000 1100
31 #define LCD_UNDERLINE_ON      0x0E // 0000 1110
32 #define LCD_BLINK_CURSOR_ON   0x0F // 0000 1111
33 #define LCD_MOVE_CURSOR_LEFT  0x10 // 0001 0000
34 #define LCD_MOVE_CURSOR_RIGHT 0x14 // 0001 0100
35 #define LCD_SHIFT_LEFT        0x18 // 0001 1000
36 #define LCD_SHIFT_RIGHT       0x1E // 0001 1100
37
38 #define LCD_DISPLAY_ON_CURSOR_OFF 0x0C
39 #define LCD_DISPLAY_OFF_CURSOR_OFF_BLINK_OFF 0x08
40 #define LCD_4BITS_2LINES_5x8FONT 0x28
41 #define LCD_INCREMENT_NO_SHIFT 0x06
42
43 #define LO_NIBBLE(b) (((b) << 4) & 0xF0)
44 #define HI_NIBBLE(b) ((b) & 0xF0)
45
46 void LcdInit();
47 void LcdWriteCommand(unsigned char data, unsigned char cmdtype);
48 void LcdWriteChar(unsigned char data);
49 void LcdWriteString(char *s);
50 void LcdSetPosition(unsigned char row, unsigned char column);
51
52 #endif
```

## 2.8 Speed enhancement

2023-12-11:

Die Ausgabe an das LCD-Display war brutal langsam. Deshalb habe ich den Code analysiert. Im Kommando `LcdWriteCommand()` habe ich eine lange Warteschleife entdeckt. Ich habe die Zeit verkürzt

Und siehe da, es geht jetzt viel schneller:

```
19
20 void LcdWriteCommand(unsigned char data, unsigned char cmdtype)
21 {
22     unsigned char byte;
23
24     byte = (HI_NIBBLE(data) | LCD_BL) | LCD_EN;
25     I2cTransmit(0x27, byte);
26     while(I2cNotReady());
27
28     byte = (HI_NIBBLE(data) | LCD_BL) & ~LCD_EN;
29     I2cTransmit(0x27, byte);
30     while(I2cNotReady());
31
32     // cmdtype = 0 -> One write cycle
33     // cmdtype = 1 -> Two write cycles (4 bit mode)
34
35     if (cmdtype)
36     {
37         byte = (LO_NIBBLE(data) | LCD_BL) | LCD_EN;
38         I2cTransmit(0x27, byte);
39         while(I2cNotReady());
40         byte = (LO_NIBBLE(data) | LCD_BL) | ~LCD_EN;
41         I2cTransmit(0x27, byte);
42         while(I2cNotReady());
43     }
44
45     // _delay_cycles(80000);
46     _delay_cycles(10000);
47 }
48 void LcdWriteChar(unsigned char data)
```

## 2.9 Fotogalerie

Ausgabe eines Strings auf dem Display:



## 2.10 Lessons learned

### 2.10.1 Probleme bei der IBN des Displays

Ich hatte das Display in der Vergangenheit bereits mit einem **STM32** und einem **Raspberry Pi** ohne Probleme in Betrieb genommen. Hier wurden keine Pull-Up Widerstände benötigt.

Im Datenblatt von Sunfounder.com steht ausdrücklich, daß Pull-Up Widerstände zu verwenden sind.



[www.sunfounder.com](http://www.sunfounder.com)

### I2C LCD1602 Module

#### Introduction

As we all know, though LCD and some other displays greatly enrich the man-machine interaction, they share a common weakness. When they are connected to a controller, multiple IOs will be occupied of the controller which has no so many outer ports. Also it restricts other functions of the controller. Therefore, LCD1602 with an I2C bus is developed to solve the problem.

I2C bus is a type of serial bus invented by PHILIPS. It is a high performance serial bus which has bus ruling and high or low speed device synchronization function required by multiple host system. I2C bus has only two bidirectional signal lines, Serial Data Line (SDA) and Serial Clock Line (SCL). The blue potentiometer on the I2C LCD1602 is used to adjust backlight to make it easier to display on the I2C LCD1602.



- **GND**: Ground
- **VCC**: Voltage supply, 5V.
- **SDA**: Serial data line. Connect to VCC through a pull up resistor.
- **SCL**: Serial clock line. Connect to VCC through a pull up resistor.

#### I2C Address

The default address is basically 0x27, in a few cases it may be 0x3F.

Ich habe diese Information überlesen, bzw. dachte, daß sich auch beim MSP controller-intern vorhanden sind, wenn ich die Pins auf I2C konfiguriere, das war falsch.

**Betriebsblindheit** war mit eine Ursache, daß ich die Information des Datenblatts überlas.

Mit der Hilfe des Oszis und etwas Glück konnte ich das Problem lösen.

### 2.10.2 Verwendung des Struktogrammeditors

Verwendetes Tool: Siehe [2.6](#) und [1.5](#).

Das Erzeugen des C-Codes aus dem Struktogrammeditor ist ein nettes Feature für sehr einfache Programme. Leider ist es kompliziert, Änderungen zurück zu pflegen.

Ich würde in Zukunft nicht mehr versuchen, lauffähigen Code aus dem Struktogrammeditor zu erzeugen.

Außerdem ist die Bedienung des Tools etwas hakelig, vielleicht wäre ein anderes Tool besser geeignet.

### 2.10.3 Verwendung von CodeComposerStudio

Ich muß mir eine bessere Strategie zum Ablegen meiner Projekte überlegen.

Durch die wirre Ablage des Projekts und der dazugehörigen Dokumente, erschien es mir auch zu kompliziert, eine SCC zu verwenden (siehe [2.10.4](#)).

#### 2.10.4 Source Code Control

Ich war zu faul ein SCC wie SVN zu verwenden. Unentschuldbar, sollte in Zukunft nicht mehr passieren.