Versuch 4: I²C

In diesem Versuch ergänzen wir unser bisheriges Programm und verwenden die I²C-Schnittstelle. Über die I²C-Schnittstellle werden die Messdaten eines Temperatursensors ausgelesen. Außerdem wird über die I²C- Schnittstelle eine LCD-Anzeige zur Visualisierung des Temperatur-Messwertes angesteuert.

I²C:
I²C (gesprochen "I-Quadrat-C") ist eine Zwei-Draht-Kommunikationsschnittstelle. Zahlreiche Bausteine (Sensoren, Aktoren, Speicher, …) verwenden diese Schnittstelle und können damit an den STM32-Mikrocontroller angeschlossen werden. Die Schnittstelle ist ein serieller Bus und besteht aus einer Takt- und einer Datenleitung, sie erlaubt ca. 100 Teilnehmer. Jeder Teilnehmer hat eine 7-Bit Adresse. Nach erfolgreicher Adressierung können Daten bidirektional im synchronen Halbduplex-Verfahren mit 100 kBit/s ausgetauscht werden.

Informieren Sie sich über I²C und über das Übertragungsprinzip. Eine kurze Beschreibung zum Thema I²C finden Sie in den Unterlagen zum Versuch sowie im Internet unter:

https://www.youtube.com/watch?v=6IAkYpmA1DQ

http://www.i2c-bus.org/

https://www.mikrocontroller.net/articles/I%C2%B2C

In diesem Versuch werden wir mit einer Bibliothek auf die I²C-Schnittstelle zugreifen, Sie brauchen dafür keine Kenntnisse über die I/O-Register des I²C-Teils des Prozessors (falls es Sie doch interessiert: siehe Seite 759 des Reference Manuals). Die I²C-Bibliothek besteht aus den Dateien `I2C.c` und `I2C.h`, es wird der I²C-Kanal 1 verwendet mit SCL an Port B6 und SDA an Port B7. Die dort gegebenen Funktionen sind in Anhang 1 nochmals kurz zusammengefasst. Bei der Verwendung der Routine `I2CInit()` wird die eigene Geräteadresse unseres Mikroprozessors übergeben; diese Adresse ist frei wählbar, muss aber niedriger sein, als alle anderen Geräteadressen; Sie können z.B. 0x08 verwenden.

Am Ende sollten Sie ein funktionierendes Projekt haben, im dem alle wesentlichen Punkte der Versuche 1 bis 4 integriert sind!

Aufgabe 4.1: I²C- Temperatur
Als Temperatursensor wird der Baustein MCP9808 mit der Adresse 0x18 verwendet. Das Datenblatt finden Sie in den Unterlagen, für das Verständnis der Kommunikation mit dem Baustein ist das Kapitel 5 (Seite 16) wichtig. Der MCP9808 besitzt 9 Funktions-Register und ein Auswahl-Register (Registerpointer), das die Adresse (0…8) des ausgewählten Funktionsregisters enthält. Wenn man also auf eines der Funktionsregister zugreifen will, muss vorher der Registerpointer beschrieben werden.
Für uns interessant ist das Ambient Temperature Register (Register 5), ein 16-Bit Register, das neben einigen Statusbits die Temperatur in Grad Celsius mit 4 binären Nachkommastellen enthält (siehe Seite 24 im Datenblatt). Zusätzlich wird noch das Resolution Register verwendet (Register 8, Seite 29), hier werden in Bit 0 und 1 verschiedene Temperaturauflösungen eingestellt. Es kann zunächst die Voreinstellungen des Bausteins mit der höchsten Auflösung von 0,0625 °C verwendet werden.

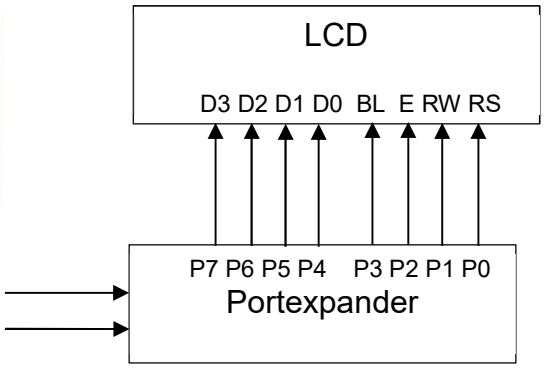
Aufgabe: Fügen Sie in Ihr Projekt die C-Module `I2C.c` und `Temperatur.c` ein. Verwenden Sie die Funktion `I2CInit(Adr)` zur Initialisierung der I²C-Schnittstelle, wobei `Adr` die eigene Adresse des Mikrocontrollers ist (z.B. 8). Ergänzen Sie die Funktionen im Modul `Temperatur.c` so, dass die Temperatur mit der Funktion `TempRead` ausgelesen und in 1/16 Grad zurückgegeben wird, sowie mit der Funktion `TempSetRes` die Auflösung geändert werden kann. Die von `I2C.c` zur Verfügung gestellten Funktionen sind in Anhang 1 aufgelistet.

Die `ExecCmd`-Funktion im Hauptprogramm soll um die Kommandos `tr` und `ts` ergänzt werden. Das Kommando `tr` liest die Temperatur vom Sensor aus und zeigt die Temperatur mittels putty an, das Kommando `tsx` setzt die Auflösung der Temperaturmessung (x=0 → 0,5°C, x=1 → 0,25°C, etc.).

Analysieren Sie das I²C-Protokoll mit Hilfe des Oszilloskops.

Aufgabe 4.2: I²C LCD-Anzeige

Die LCD-Anzeige besteht aus zwei Modulen, einem sog. Portexpander und der eigentlichen 2-zeiligen LCD-Anzeige.



Der Portexpander basiert auf dem Baustein PCF8574 und ist ein digitaler 8-Bit-Port, der über die I²C-Schnittstelle angesprochen wird. Die LCD-Anzeige verwendet den Baustein HD44780 und ist an den digitalen Port angeschlossen.
Die Leitungen P4-P7 werden zur Datenübertragung verwendet. P0 ist mit dem Pin RS, P1 mit dem Pin RW und P2 mit dem Enable-Eingang des LCD-Bausteins verbunden. Über P3 wird die Hintergrundbeleuchtung angesteuert (1 = an).

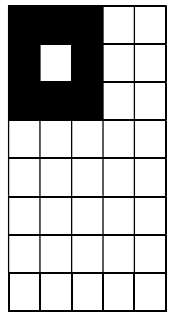
Übertragungsschema:
RS = 0 → es wird ein Kommando übertragen, RS = 1 → es werden Daten übertragen.
RW = 0 → Write, RW = 1 → Read
Der LCD-Baustein übernimmt die Daten mit der fallenden Flanke des Signals E, d.h. die Daten müssen in 2 Schritten übertragen werden. Im 1. Schritt ist P2 (E) gesetzt, im zweiten Schritt mit identischen Daten liegt P2 auf 0. Damit wird die fallende Flanke an E realisiert. Nachdem nur 4 Datenleitungen verwendet werden, muss ein Byte in zwei Teilen übertragen werden. Zuerst werden die oberen 4 Bit (D4-D7), dann die unteren 4 Bit (D0-D3) übertragen.

Aufgaben:

4.2.1 Binden Sie die Datei `LCD.c` und `LCDInit.o` in Ihr Projekt ein. Verwenden Sie die Funktion `LCDInit()` zum einmaligen Initialisieren des Displays. Das Modul `LCD.c` enthält die Funktionen `LCDSendCmd` und `LCDSendData` zum Ansteuern des Displays. Ergänzen Sie die Funktion `LCDSendData` entsprechend dem obigen Übertragungsschema.
Fügen Sie zum Testen der Übertragung in die Funktion `ExecuteCmd` die Kommndos `lc[cmd]` und `ld[data]` ein, wobei `[cmd]` und `[data]` für eine Hexzahl stehen. `lc` soll ein Kommando und `ld` Daten übertragen.
Geben Sie einige Zeichen als Daten aus und testen Sie ein LCD-Kommando laut Datenblatt „LCD1602“.

4.2.2 Schreiben Sie die Funktion `void LCDClear(void)`, die durch Senden des entsprechenden Kommandos (siehe Datenblatt) das Display löscht. Nach dem Absenden des Kommandos muss etwas gewartet werden, da das Kommando vom Display relativ langsam abgearbeitet wird und ein zu schnelles Senden der nächsten Daten/Kommandos zur Fehlfunktion des Displays führen kann. Ermitteln Sie im Datenblatt, wie lange gewartet werden muss und nutzen Sie zum Warten die vorgegebene Funktion `voidDelayMs(int ms)`.

4.2.3 Schreiben Sie nun eine Funktion `void LCDDefineDegree(void)`, um das Symbol „Grad“ (entsprechend der rechts stehenden Grafik) im Zeichensatz des Displays an ASCII-Position 0 abzuspeichern. Rufen Sie diese Funktion einmalig in `main()` auf und testen Sie die Ausgabe des selbstdefinierten Symbols mit geeigneten Kommandos aus 4.2.1. Sinnvollerweise wird zum Abschluss der Funktion das Display mit der Routine aus 4.2.2 gelöscht, da sonst alle weiteren Ausgaben evtl. „außerhalb“ des Displays landen und damit nicht zu sehen sind. Siehe Dokument „LCD1602“.



Aufgabe 4.3: Temperatur anzeigen

Nun soll zyklisch die Temperatur abgefragt und im Display angezeigt werden. Verwenden Sie dazu die Funktion `sprintf`, um den Ausgabetext in einem String vorzubereiten und geben Sie dann den String mit `LCDSendString(*text)` auf dem Display aus. Nutzen Sie dazu das selbstdefinierte Symbol „°“ und ein dahinter gestelltes „C“

Der zyklische Aufruf erfolgt am einfachsten, wenn Sie im Handler des Systick-Timers, der bereits für den Schrittmotor benötigt wird, eine globale Variable Ticks hochzählen und im Hauptprogramm diese Variable abfragen. Die Temperaturmessung mit Anzeige soll im Hauptprogramm einmal pro Sekunde aufgerufen werden.

Zusätzlich soll der Schrittmotor als analoge Temperaturanzeige verwendet werden. Dazu müssen Sie aus dem Temperaturwert die Sollposition des Schrittmotors berechnen und die Skalierung so bemessen, dass der Temperaturbereich von 20 Grad Celsius bis 30 Grad Celsius auf eine 180 Grad-Drehung des Schrittmotors abgebildet wird.

Anhang 1: I2C- Bibliothek

Folgende Funktionen werden für die *I2C-Bus Kommunikation* zur Verfügung gestellt:

- *i2c.c:*

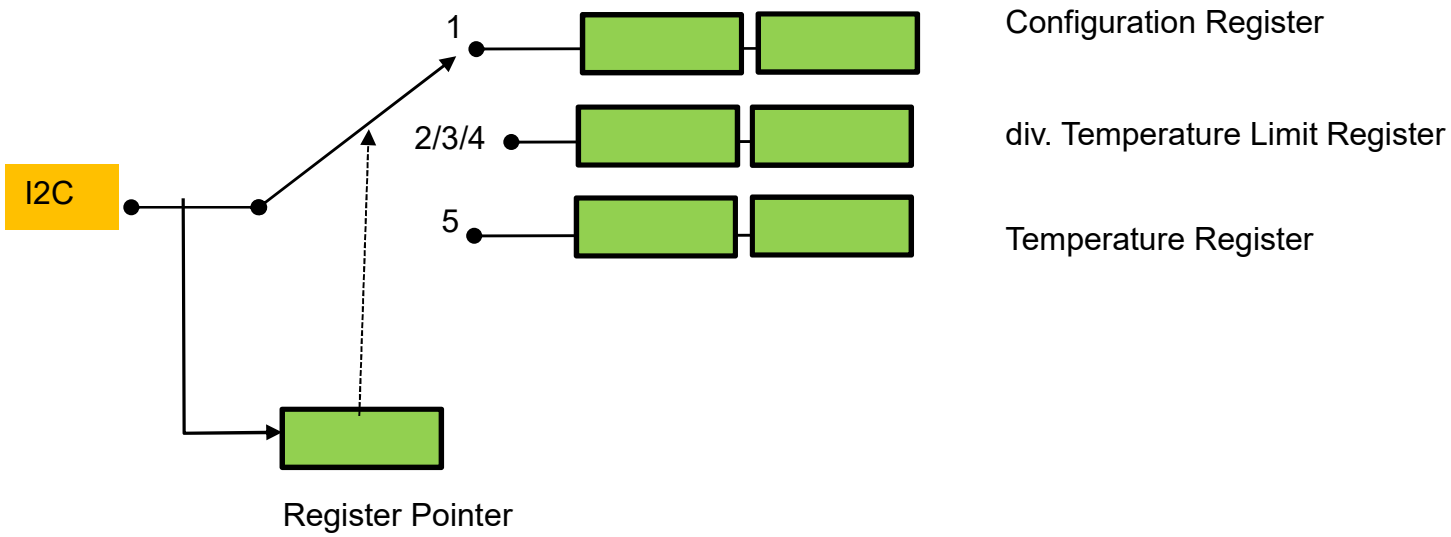
```
/* ================================================================
   I2C Init Routine
   ================================================================ */
void I2CInit(
        unsigned char address               // 7-Bit I2C Adresse des Masters (µC)
)

/* ================================================================
   I2C Write n Byte From Buffer Routine
   ================================================================ */
void I2CWrite(
        const unsigned char * buf           // Puffer der zu sendenden Byte
        ,unsigned int n                     // Anzahl zu sendender Byte
        ,unsigned char dest                 // 7-Bit I2C Adresse des Busteilnehmers
)

/* ================================================================
   I2C Read n Byte To Buffer Routine
   ================================================================ */
void I2CRead(
        unsigned char * buf                 // Puffer für eingelesene Byte
        ,unsigned int n                     // Anzahl zu lesender Byte
        ,unsigned char dest                 // 7-Bit I2C Adresse des Busteilnehmers
)
```

Typische Werte:

I²C-Adresse des Busmasters (µC): 0x08 (d.h., niedriger als jeder Slave)
I²C-Adresse der Busteilnehmer: aus den Datenblättern entnehmen, muss die *7-Bit Adresse* sein!
Puffer buf: ausreichend lang für Daten und Befehle machen, mindestens n Byte!

Hinweise

Bei der Verwendung des Register-Pointers im Temperatursensor hilft die Vorstellung, den Register-Pointer wie einen Umschalter zu verwenden.



Um also die Temperatur auszulesen muss vorher der Register Pointer mit 5 beschrieben werden. Anschließend können die 2 Bytes des Temperaturregisters gelesen werden