4.2.A Peripherie ansprechen in C/I2C-Bus – Arbeitsblatt

Aufgabe 1: Einfache Kommunikation auf dem I²C-Bus

Wenn man die Taktfrequenz sehr niedrig einstellt, kann man auf dem RN-Board die Kommunikation mit den LEDs an PC1 (SDA) und PC0 (SCL) beobachten.

a) Wie muss man die Register TWBR und TWSR einstellen, damit man eine möglichst niedrige Taktfrequenz bekommt?

b) Wie hoch ist die niedrigste Taktfrequenz bei $f_{CPU} = 16\,\text{MHz}$?

c) Mit Hilfe der Fuse-Bits (siehe M 0.4) kann man statt des externen Quarzes den internen RC-Oszillator mit $f_{CPU} = 1\,\text{MHz}$ benutzen. Wie hoch ist die niedrigste Taktfrequenz dann?

d) Stellen Sie die Taktfrequenz so niedrig wie möglich. Dazu können Sie entweder die Fuse-Bits nehmen oder einen externen Quarz mit niedrigerer Frequenz. Starten Sie nun `src/rn32/twi1.c` und versuchen Sie das Leuchten der LEDs zu interpretieren. Welche LED muss an SCL liegen? Welche Schritte können Sie erkennen?

Aufgabe 2: Kommunikation mit dem Temperatur-Sensor LM 75

Das Programm `src/rn32/LM75a.c` liest bereits den Sensor aus.

a) Im Datenblatt sind in den Abbildungen 7 bis 12 mehrere Übertragungsprozeduren gezeigt. Welche wird hier benutzt?

b) In `src/rn32/twi2.c` sind die Befehle zu jeweils einer Aktion (Init, Start, Stop, Write, Read mit Ack, Read ohne Ack) in einer Funktion zusammengefasst:

```
1 void i2c_init(void);
2 void i2c_start(void);
3 void i2c_stop(void);
4 void i2c_write(unsigned char daten);
5 unsigned char i2c_read_ack();
6 unsigned char i2c_read_nak();
```

Sie sollen nun diese Funktionen übernehmen und aus `lm75a.c` ein Programm `lm75b.c` machen, das diese Funktionen benutzt.

c) Bis jetzt haben Sie nur *eine* der Übertragungsprozeduren aus den Abbildungen 7 bis 12 im Datenblatt realisiert. Immerhin kann man damit jetzt die Temperatur ermitteln. Für den Schaltausgang sind aber weitere Prozeduren nötig.

Deshalb wird jetzt sinnvollerweise eine zweite Ebene von Funktionen angelegt, von denen jede eine bestimmte Prozedur abarbeitet:

```
1 void lm75_write8(unsigned char reg, unsigned char val);//7
2 void lm75_read8(unsigned char reg, unsigned char *val);//8
3 void lm75_read8preset(unsigned char *val);//9
4 void lm75_write16(unsigned char reg, unsigned char hi, unsigned char lo);//10
5 void lm75_read16(unsigned char reg, unsigned char *hi, unsigned char *lo);//11
6 void lm75_read16preset(unsigned char *hi, unsigned char *lo);//12
```

Als Beispiel ist hier einmal `lm75_read16preset()` aufgeführt:

```
1 void lm75_read16preset(unsigned char *hi, unsigned char *lo)
2 {                                 /* Fig. 12 */
3     i2c_start();                  /* S       */
4     i2c_write(LM75A+READ);        /* Adresse */
5     *hi=i2c_read_ack();           /* Daten 2 */
6     *lo=i2c_read_nak();           /* Daten 1 */
7     i2c_stop();                   /* P       */
8 }
```

Damit brauchte die Hauptschleife unseres Programms also nur noch ein oder zwei Zeilen. Legen Sie die restlichen Funktionen entsprechend an und vereinfachen Sie `main()`! Das neue Programm soll `lm75c.c` heißen.

d) Nun haben Sie die Möglichkeit, die Register für den Schaltausgang zu setzen. Setzen Sie Register HYST auf 25° und TOS auf 26°. Nun versuchen Sie, durch Erwärmen des Sensors die LED einzuschalten und durch Abkühlen die LED auszuschalten. Sie können auch HYST auf die aktuelle Temperatur ϑ setzen und TOS auf $\vartheta + 1°$.

Aufgabe 3: Ausgabe auf einem LCD

Nun ist die Ausgabe auf einem LCD (ersatzweise auf einem LED-Display oder über die serielle Schnittstelle) an der Reihe.

a) Legen Sie für dieses Projekt ein kleines Verzeichnis an!

b) Teilen Sie Ihr LM-75-Programm auf in `lm75.c`, `lm75.h` und `main75.c`! Testen Sie, ob Ihr neues, modulares Programm funktioniert!

c) Teilen Sie Ihr LCD-Programm auf in `lcd.c`, `lcd.h` und `mainlcd.c`! Testen Sie nun Ihr modulares LCD-Programm!

d) Nun legen Sie `main75.c` und `mainlcd.c` zusammen zu einem gemeinsamen Programm `main.c` und realisieren (zunächst) eine ganzzahlige Ausgabe.

e) Zum Schluss können Sie versuchen, die Ausgabe des oder der Nachkomma-Bits zu realisieren. Vermeiden Sie Gleitkommazahlen und `printf()`-Befehle, sie fressen nur Ihr wertvolles RAM!