Gruppe: Leander Stark, Jonas Dollinger Datum: 18.01.2021

Hochschule Aalen

I2C Temperatursensor

Lernzeile:

- Kennenlernen des i2C Busses und Protokols
- Grundverständnisse für die I2C Peripherie
- Analyse des Temperatursensor Datenblattes
    - Registersatz
    - Daten Protokol / Zugriffssequenz zum Lesen/Schreiben der Register
- Implementierung einer für die Aufgabe passende Funktionaltät für das Lesen der Temperatur

Aufgabe: Entwickeln Sie ein Programm um die Temperatur mit dem auf dem Board verbauten Temperatursensor TCN75 gemessen und danach über die serielle Schnittstelle übertragen wird.

- Identifizieren Sie die Schnittstelle im Schaltplan und konfigurieren Sie die I2C Peripherie des Mikrocontrollers
    - Verwenden Sie dabei die typische Übertragsrate welche im Datenblatt des Temperatursensors angegeben wurde.
- Machen Sie sich mit dem Übertragungsprotokol, Register und Datenwerte des TCN75 vertraut, welche zum Auslesen der Temperatur benötigt werden.
    - Es hilft Ihnen für die spätere Programmierung eine komplette Sequenz zu skizzieren, mit:
        - Start, Adresse/RW, Ack, Daten, ... Stop
        - Welcher Teilnehmer was auf den Bus legt.
        - Was wird vom Master (dsPIC) initiiert, bzw. auf was muss reagiert werden.
- Der I2C Datentransfer vom Sensor zum Mikrocontroller kann am einfachsten in einer blockierenden Funktion realisiert werden.
    - Jedoch neigt die I2C Schnittstelle insbesondere im Debug Betrieb zum Blockieren. Im EDA Wiki unter Code Snippets finden sie ein Beispiel um eine blockierte Schnittstelle wieder frei zu bekommen. es empfiehlt sich definitiv dies in der Init Funktion auszuführen.

- Entwickeln Sie eine FSM für den I2C Datentransfer der Temperaturmessung. Diese kann in der Super Loop aufgerufen werden oder aber (besser?) im I2C Interrupt.
- Für den Start oder Ende des Datentransfers benötigen Sie natürlich noch Steuervariablen für die Kommunikation I2C FSM und andere Programmteile.
- Die Tempertur soll zyklisch einmal je Sekunde gemessen werden, da sich die Temperatur nur langsam ändert.
    - Optional: Parametrierbare Zykluszeit,z.B: über Taster oder UART Kommandos
- Nach der Messung soll der Temperaturwert über die serielle Schnittstelle übertragen werden, z.B. "Temperatur: 23,5 °C\n"
- Hinweis: Verwenden Sie das UART Programm (Aufgabe 6) und fügen Sie dort die neue Temperaturfunktion hinzu, da hier schon alle benötigten Timer und UART Funktionen aktiv sind.

Initialisierungsfunktion I2C

```
1.  void initI2C(){
2.      I2C2CONbits.A10M = 0;
3.      I2C2BRG = 245; //100kHz
4.
5.      // Einschalten I2C mit eigenem Workaround, I2C Peripheriemodul kann hier leider
    nicht verwendet werden
6.      I2C_SDA_TRIS = 1;    // Pins wie einen Open-Kollektor-
    Treiber verwenden, d.h. 1 - hochohmig, 0 wird getrieben
7.      I2C_SCL_TRIS = 1;
8.      I2C_SDA = 0;
9.      I2C_SCL = 0;
10.
11.         int j;
12.         for (j=0; j<=9; j++)    // takten bis min 1 Byte
13.         {
14.           I2C_SCL_TRIS = 0; delay_ms(1); // 5 us wÃ¤ren ausreichend ...100 kBaud
15.           I2C_SCL_TRIS = 1; delay_ms(1);
16.         }
17.         // Start Condition senden
18.         I2C_SCL_TRIS = 0;  delay_ms(1);
19.         I2C_SDA_TRIS = 0;  delay_ms(1);
20.         // Stop Condition senden
21.         I2C_SCL_TRIS = 1;  delay_ms(1);
22.         I2C_SDA_TRIS = 1;  delay_ms(1);
23.
24.         // Nun I2C erst anschalten
25.         _MI2C2IF = 0; //Interrupt falls noetig
26.         _MI2C2IE = 0;
27.         I2C2CONbits.I2CEN = 1;
28.
29.     //Sensor Pointer auf TEMP Register setzten
30.     I2C2CONbits.SEN=1; //start
31.     while(I2C2CONbits.SEN==1){}
32.
33.     //Tx Device address + Write bit
34.     I2C2TRN=0b10010000;
35.     while(I2C2STATbits.TRSTAT==1){}
36.
37.     if (I2C2STATbits.ACKSTAT==1){   //if NACK received, generate stop condition and
    exit
38.         I2C2STATbits.ACKSTAT=0;
39.         I2C2CONbits.PEN=1;
40.         while(I2C2CONbits.PEN==1){} //wait for the stop interrupt;
41.         return;
42.     }
43.
44.     //Tx Register Address
45.     I2C2TRN=0b00000000; //Pointer auf TEMP REGISTER setzten
46.     while(I2C2STATbits.TRSTAT==1){}
47.
48.     if (I2C2STATbits.ACKSTAT==1){   //if NACK received, generate stop condition and
    exit
49.         I2C2STATbits.ACKSTAT=0;
50.         I2C2CONbits.PEN=1;
51.         while(I2C2CONbits.PEN==1){} //wait for the stop interrupt;
52.         return;
53.     }
54.
55.     I2C2CONbits.PEN=1; //stop
56.     while(I2C2CONbits.PEN==1){} //wait for the stop interrupt
57. }
```

In dieser Funktion wird als erstes der Baud Rate Generator so gesetzt, dass bei einer Systemfrequenz von 50Mhz ungefähr eine Baudrate von 100kHz erreicht werden (Z.3).

Die Berechnung erfolgt wie folgt:

$$I2CBRG = \left(\left(\frac{1}{F_{SCL}} - 130ns\right) * F_{CY}\right) - 2$$

Danach folgt das Einschalten des I2C-Moduls mit Hilfe des Code-Snippets aus der EDA-Wiki. Dieses sorgt dafür das die Schnittstelle nicht blockiert, bzw. bei einem Neustart wieder freigegeben werden kann. Nach dem das I2C-Modul aktiviert worden ist, erfolgt eine einmalige Einrichtung des Temperatursensors.

Dabei wird der Pointer auf das TEMP Register gesetzt, sodass bei späterem Auslesen das Temperatur-Register angesprochen wird. Die Adresse des Sensor ist dabei „1001000“. Zusätzlich wird noch eine „0“ angehängt, um das Schreiben einzuleiten. Der weitere Verlauf des Protokolls wird im Nachfolgenden weiter erläutert.

```
1.  void *FSM2_Idle(void)
2.  {
3.      static int c = 0;
4.      if (c>=999){
5.          c=0;
6.          return FSM2_Start;
7.      }
8.      c++;
9.      return FSM2_Idle;
10.
11. }
```

Die Finite State Machine wird im Rhythmus einer Millisekunde von der Main Superloop aufgerufen. Da die FSM im ersten Schritt immer zuerst in die obige Idle-Funktion springt, lässt man dort einen Counter auf 1000 zählen, um die Ausführung der weiteren States im Sekundenrhythmus zu ermöglichen.

```
1.  void *FSM2_Start(void)
2.  {
3.      I2C2CONbits.SEN=1; //Start
4.      while(I2C2CONbits.SEN==1){}
5.      return FSM2_Adresse;
6.  }
```

Im ersten State wird die I2C-Schnittstelle angesprochen. Da dieser SEN-Bit bei erfolgreicher Ausführung wieder auf 0 gesetzt wird, steht in Zeile 4 eine while-Schleife welche auf dieses Ereignis wartet, um mit dem Adresse-State fortzufahren.

```
1.  void *FSM2_Adresse(void)
2.  {
3.      //Tx Device address + Read bit
4.      I2C2TRN=0b10010001;
5.      while(I2C2STATbits.TRSTAT==1){} //Warten solange übertragen wird
6.      return FSM2_ACK_Receive;
7.  }
```

Der Adresse-State besitzt die Funktion, den entsprechenden I2C-Slave anzusprechen. Der Slave (Temperatursensor) besitzt in unserem Falle die Adresse in Zeile 4 (+ „1“ für Lesen). Diese Adresse wird über den TRSTAT-Command angesprochen und es wird auf die erfolgreiche Übertragung gewartet (Zeile 5).

```
1.  void *FSM2_ACK_Receive(void)
2.  {
3.      if (I2C2STATbits.ACKSTAT==1){   //if NACK received, generate stop condition and
    exit
4.          I2C2STATbits.ACKSTAT=0;
5.          return FSM2_Stop;
6.      }
7.      return FSM2_Data_Receive;
8.  }
```

Der Slave antwortet auf diese Adresse mit einem Acknowledge / Not Acknowledge, welches in diesem ACK_Receive State empfangen wird. Sollte ein NACK empfangen werden, wird die weitere Ausführung abgebrochen, indem in Zeile 5 den Stop-State gewechselt wird. Sollte ein Acknowledge zurückkommen, werden im nächsten Schritt die Daten empfangen.

```
1.  void *FSM2_Data_Receive(void)
2.  {
3.      int N=2; //2 bytes empfangen
4.      int i;
5.
6.      for(i=0;i<N;i++){
7.          I2C2CONbits.RCEN=1; //Empfangen aktivieren
8.          while(I2C2CONbits.RCEN==1){} //RCEN cleared automatically when SSP1IF goes h
    igh
9.
10.         data[i]=I2C2RCV;
11.
12.         //ACK sequence
13.         if (i<N-1){ I2C2CONbits.ACKDT=0; } //jedes byte mit ACK bestätigen
14.         else {I2C2CONbits.ACKDT=1;} //send NACK if this is the last Byte
15.
16.         I2C2CONbits.ACKEN=1; //start ack/nack sequence
17.         while(I2C2CONbits.ACKEN==1){}
18.
19.     } //end for loop
20.     return FSM2_Stop;
21. }
```

In diesem State werden die 2 Bytes des TEMP-Registers vom Sensor empfangen. Dafür wird in einer for-Schleife das Empfangen für den Master aktiviert und das I2CRCV-Register ausgelesen. Nach dem ersten ausgelesenen Byte sendet der Master ein ACK-Signal. Nach dem letzten ausgelesenen Byte ein NACK-Signal.

Sind beide Bytes ausgelesen worden, wechselt die FSM in den Stop-State.

```
1.  void *FSM2_Stop(void)
2.  {
3.      I2C2CONbits.PEN=1;
4.      while(I2C2CONbits.PEN==1){} //wait for the stop interrupt
5.
6.      float temp = data[0]<<8|data[1];
7.      char str[16];
8.      sprintf(str,"%f",temp/256);
9.      putsUART("Temperatur: ");
10.     putsUART(str);
11.     putsUART("°C");
12.     putsUART("\n");
13.
14.     return FSM2_Idle;
15. }
```

Im letzten State wird ein Stop-Signal ausgesendet. Auch hier wird wieder solange gewartet, bis das Signal erfolgreich versendet wurde. Zuletzt erfolgt hier noch die Berechnung der Temperatur aus den zwei ausgelesene Bytes und das Versenden über die UART-Schnittstelle. Danach wechselt die FSM wieder in den Idle-State.