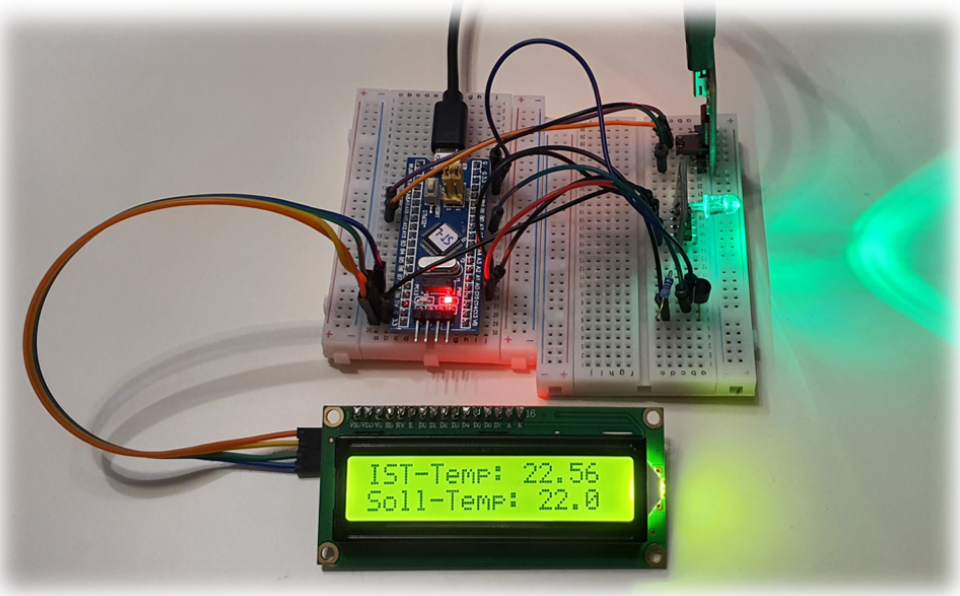
Project Documentation:
Temperature-Measuring and Visualization System

Jonas Berger



Datum: 18. November 2023

Abstract

Diese Dokumentation soll das Projekt "Temperature-Measuring and Visualization System" beschreiben. Dabei werden theoretische Vorkenntnisse, Schaltpläne, die konkrete Implementierung, etwaige Messungen und ein "Lessons Learned" dokumentiert.

Inhaltsverzeichnis

1 Projektspezifikation

Das nachfolgende Blockschaltbild 1 zeigt vereinfacht alle verwendeten Bauteile, Sensoren und Anzeigen mit entsprechenden Bussystemen bzw. Ansteuerungen:

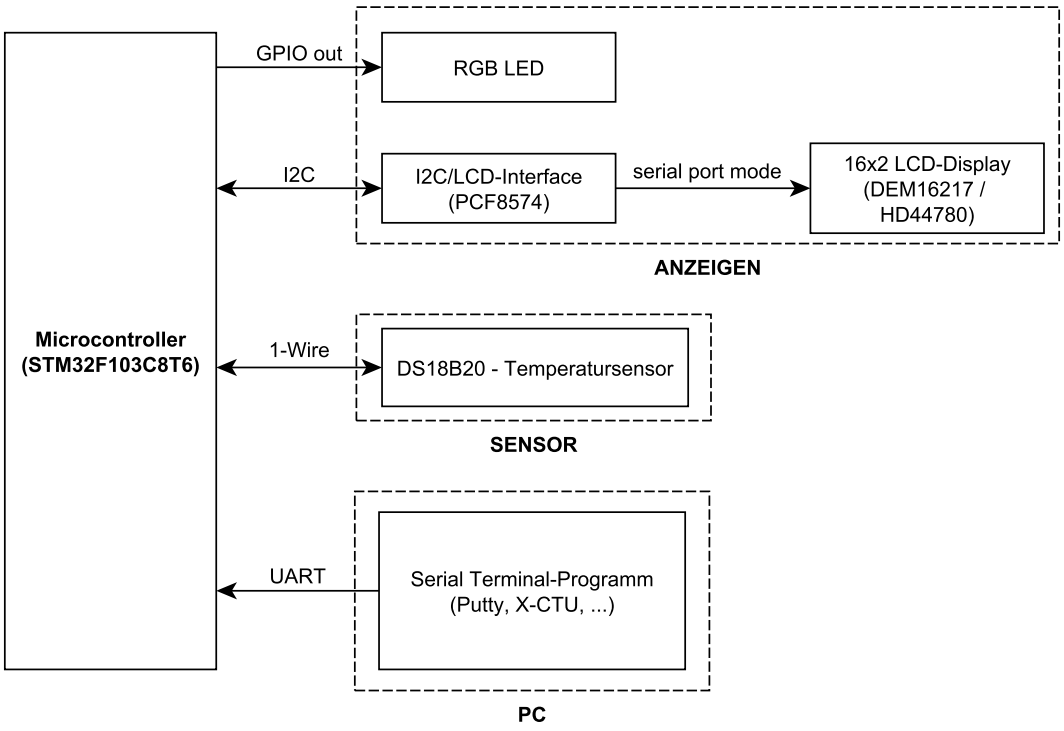


Abbildung 1: Blockschaltbild

Zunächst soll die aktuelle Temperatur mit dem DS18B20-Temperatursensor über das 1-Wire-Protokoll ausgelesen werden. Dieser Wert soll auf ein 16x2 LCD-Display ausgegeben werden.
Jedoch wird der Wert nicht direkt vom Microcontroller an das Display übertragen, sondern wird an den I^2C-to-LCD Interface-Baustein (PCF8574) via I^2C gesendet. Damit wird ein hoher Verbrauch von Portleitungen des Microcontrollers zur Ansteuerung des Displays vermieden.

Ein weiteres Feature ist die Einstellung einer Solltemperatur via UART-Protokoll über einen angeschlossenen PC mit einem Serial-Terminal-Programm. Dieser Wert wird ebenfalls am LCD-Display angezeigt.
Sollte sich die gemessene Temperatur innerhalb einer gewissen Toleranz befinden leuchtet eine RGB-LED grün. Ansonsten werden größere Abweichungen vom Sollwert stufenweise grün-rot bis rot angezeigt.

2 Verwendete Hardware

Folgende Hardware wird in dem Projekt verwendet:

- STM32F103C8T6-Microcontroller
- 16x2 LCD-Display (z.B. HD44780, DEM16217, ...)
- I2C/LCD-Interface Baustein (verwendet PCF8574-Chip)
- RGB-LED (KY-016)
- DS18B20 1-Wire-Temperatursensor
- PC mit Serial-Terminal-Programm (z.B. Putty, X-CTU, ...)

3 Schematic

In folgender Abbildung ist die gesamte Schaltung ersichtlich:

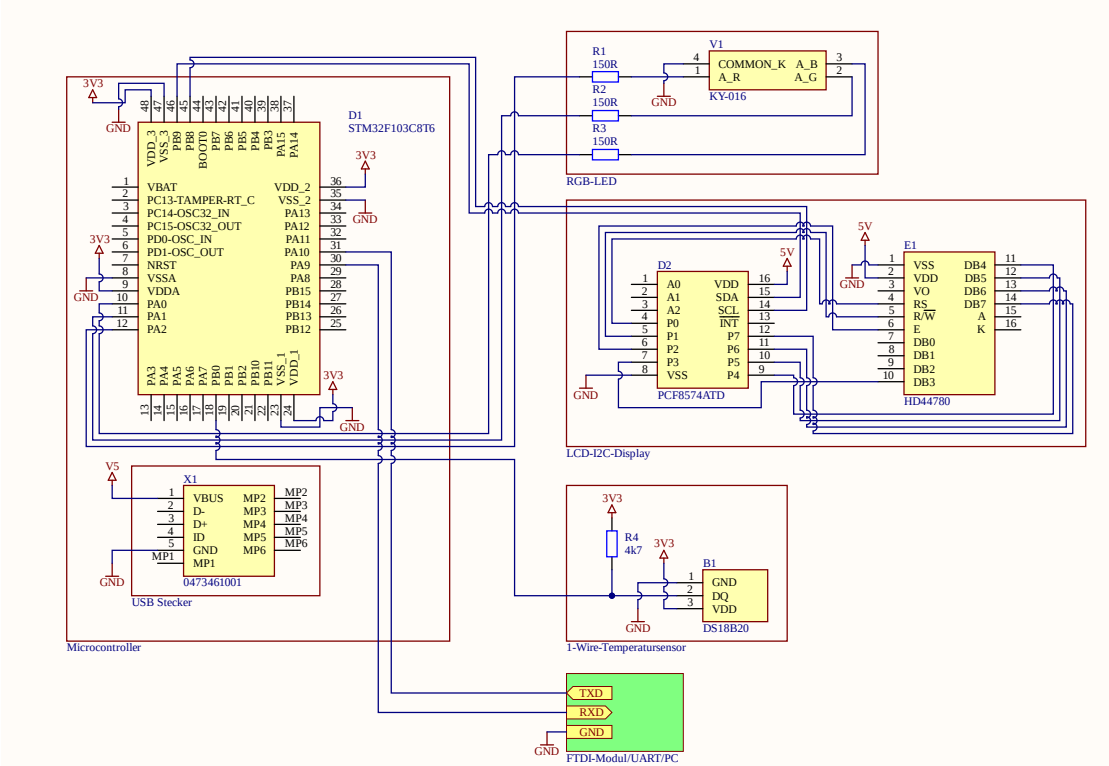


Abbildung 2: Project Schematic

4 Theorie

4.1 Bit banding

Die Cortex-M3-Memory-Map enthält zwei Bitbandbereiche. Jedes Bit im Bitband-Speicherbereich wird einem Word im Alias-Speicherbereich zugeordnet. Wird das Word im Alias-Bereich beschrieben, so hat das den gleichen Effekt wie eine Read-Modify-Write-Operation auf das Zielbit in der entsprechenden Bitbandregion.

Folgende Mapping-Formula wird zur Berechnung der Bit-Band-Adresse verwendet:
$bit\_word\_addr = bit\_band\_base + (byte\_offset \times 32) + (bit\_number \times 4)$

- bit_word_addr is the address of the word in the alias memory region that maps to the targeted bit.
- bit_band_base is the starting address of the alias region
- byte_offset is the number of the byte in the bit-band region that contains the targeted bit
- bit_number is the bit position (0-7) of the targeted bit.

4.2 1-Wire-Protokoll

Das 1-Wire-Protokoll ist ein asynchroner, bidirektionaler und half-duplex serieller Datenbus, der nach dem Master-Slave-Prinzip funktioniert und von der Firma DALLAS Semiconducter Corp. entwickelt wurde. Das Protokoll verwendetet dabei nur eine Leitung für die Datenübertragung und zwingend eine Ground-Leitung. Je nach 1-Wire-Slave wird auch eine eigene Versorgungsleitung benötigt, sonst können viele 1-Wire-Bauteile auch direkt die Datenleitung als Versorgung verwenden. Diesen Betrieb nennt man auch "Parasitären Betrieb".
Da es keine CLK-Leitung gibt, arbeitet dieser Bus mit eine Zeitschlitzverfahren. Die Übertragung eines Bits dauert standardmäßig 60μs (6μs im Overdrive-Modus). Zwischen logisch "0" und "1" wird unterschieden, je nachdem wie lange die DQ-Leitung auf "Low" ist.

Nachstehend ist der OneMaster-MultipleSlaves-Aufbau ersichtlich:

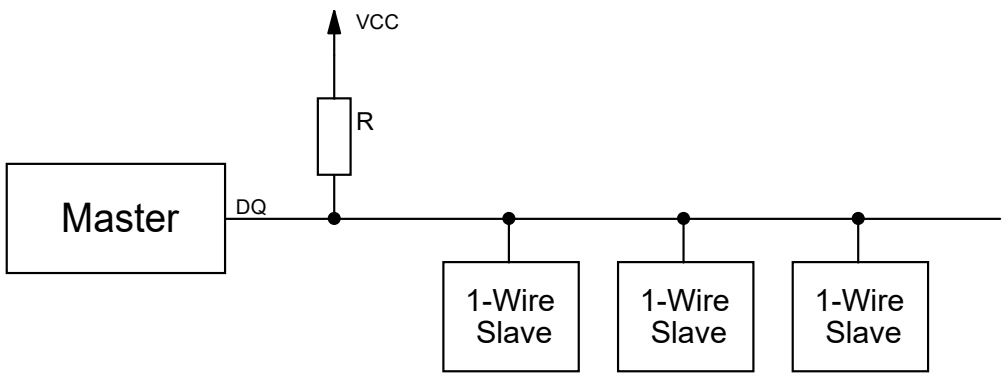


Abbildung 3: 1-Wire Master/Slaves

4.3 I²C-Schnittstelle

Die I²C-Schnittstelle ist eine weitverbreite Microcontroller-Peripherie-Schnittstelle, die von Phillips entwickelt wurde. Sie verwendet zwei Leitungen zur Datenübertragung, die jeweils mit einem Pull-Up Widerstand auf die Versorgungsspannung gezogen werden. Dabei erfolgt über die SDA-Leitung der bidirektionale Datenverkehr und über die SCL-Leitung wird der Takt übertragen.
Einzelne angeschlossene Peripheriebausteine werden über eine 7-Bit Adresse angesteuert, dass zu einer maximalen Slave-Anzahl von 112 führt, da 16 reserviert sind. Weiters verfügt die Schnittstelle über fünf unterschidliche Geschwindigkeitsmodi, nämlich Standard Mode, Fast Mode, Fast Mode Plus, High Speed Mode und Ultra Fast-mode, womit Datenübertragungsraten von 0,1 Mbit/s bis 5,0 Mbit/s möglich sind.

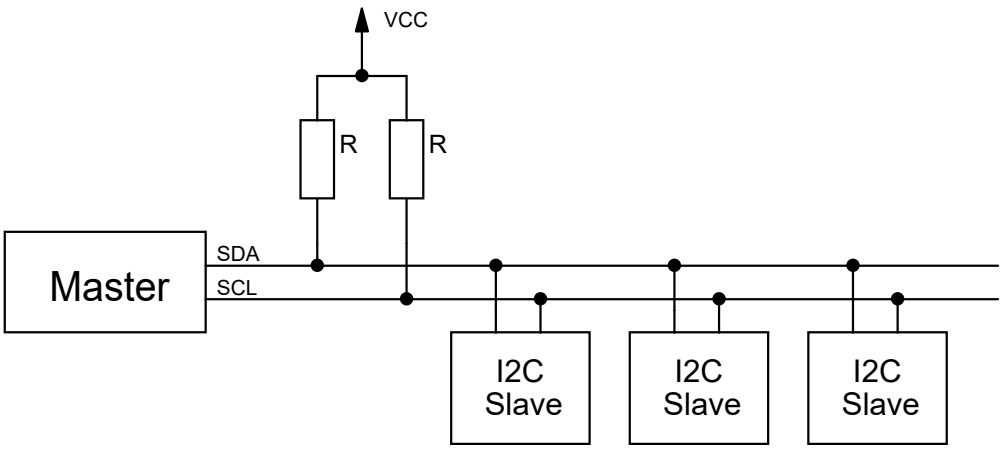


Abbildung 4: I²C Master/Slaves

5 Implementation

5.1 Fluss- und Ablaufdiagramme

Um den Befehlsablauf speziell für die 1-Wire und I^{2}C-Schnittstelle im Code (siehe 5.2) besser nachzuvollziehen, werden diese in Form von Fluss- bzw. Ablaufdiagrammen veranschaulicht.

5.1.1 1-Wire-Temperaturmessung

Der Ablauf der Temperaturmessung mit dem 1-Wire-Temperatursensor ist im nachfolgendem Flussdiagramm ersichtlich:

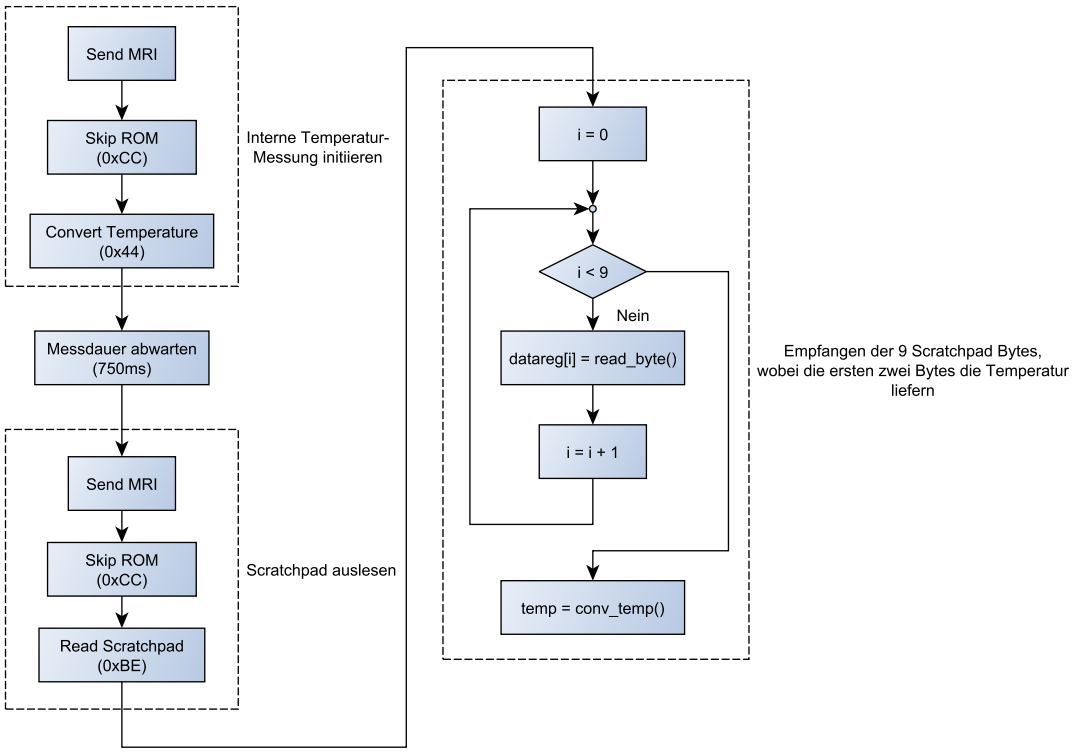


Abbildung 5: Flussdiagramm der Temperaturmessung

5.1.2 I²C/LCD-Display-Ansteuerung

Nachfolgend wird die LCD-Display-Initialisierung über die I²C-Schnittstelle dargestellt:

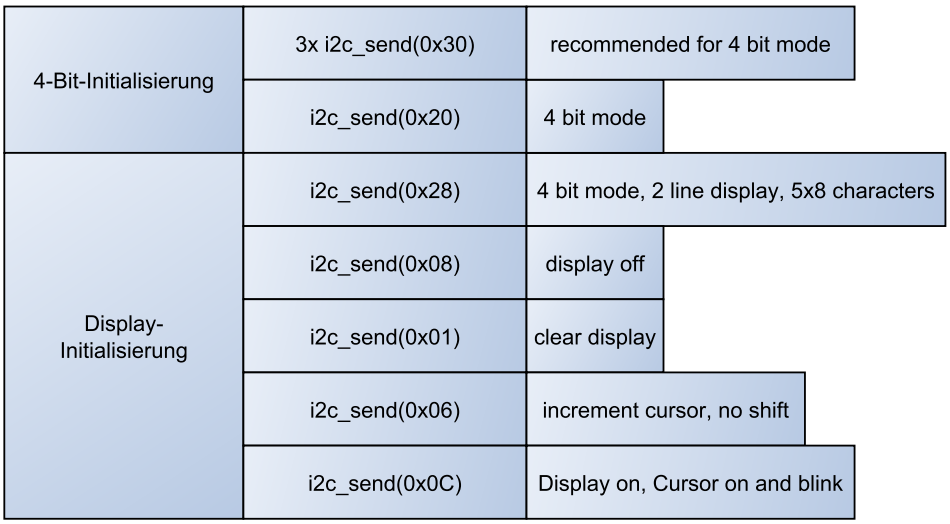


Abbildung 6: Ablaufdiagramm der LCD-Display-Initialisierung

5.2 Code-Implementierung

Es folgen nun Code-Snippets der Temperaturmessung, der LCD-Display-Initialisierung und der Main-Funktion. Dabei bauen die Code-Snippets auf den Diagrammen in 5.1 auf.

5.2.1 1-Wire-Temperatursensor Code

```
float get_temperature()
{
  int i=0;
  float temp=0;

  send_mri();       //send Master Reset Impulse; must return 0x00
  skip_ROM();       //simply calls all slaves instead of an specific address call

  write_byte(CONVERT_TEMP); //Temperaturmessung ausfuehren
  delay_ms(750);            //Messdauer abwarten

  send_mri();
  skip_ROM();

  write_byte(READ_SCRATCHPAD); //Temperaturwert auslesen

  for(i=0;i<9;i++)    //Antwort einlesen: 9 Byte grosße Scratch Pad-Inhalt einlesen
  {
    datareg[i] = read_byte();
  }
  temp = conv_temp();

  return temp;
}
```

Listing 1: Temperaturmessung

5.2.2 LCD-I²C-Display-Initialisierungscode

```
void lcd_i2c_init(void)
{
  //init wait
  delay_init();

  //init I2C1, no pin remapping, no fast mode, APB1 clock is 8 MHz
  i2c_init(I2C1, true, false, 8000000);

  // 4 bit initialisation
  delay_ms(50);            // wait for >40ms
  lcd_i2c_send_cmd(0x30);
  delay_ms(5);             // wait for >4.1ms
  lcd_i2c_send_cmd(0x30);
  delay_ms(1);             // wait for >100us
  lcd_i2c_send_cmd(0x30);
  delay_ms(10);
  lcd_i2c_send_cmd(0x20);   // 4bit mode
  delay_ms(10);

  // display initialisation
  lcd_i2c_send_cmd(0x28); // Function set --> DL=0 (4 bit mode), N = 1 (2 line display) F = 0 (5x8
      characters)
  delay_ms(1);
  lcd_i2c_send_cmd(0x08); //Display on/off control --> D=0,C=0, B=0  ---> display off
  delay_ms(1);
  lcd_i2c_send_cmd(0x01); // clear display
  delay_ms(1);
  lcd_i2c_send_cmd(0x06); //Entry mode set --> I/D = 1 (increment cursor) & S = 0 (no shift)
  delay_ms(1);
  lcd_i2c_send_cmd(0x0C); //Display on/off control --> D = 1, C and B = 0. (Cursor and blink, last two
      bits)
}
```

Listing 2: LCD-Display-Initialisierung

5.2.3 Main-Funktion

```
int main(void)
{
  float temp=0, diff = 0, solltemp = 22.0;
  char buffer[20];

  rgb_led_init();
  uart_init(9600);
  uart_clear();

  lcd_i2c_init(); // PB8 - SCL, PB9 - SDA
  lcd_i2c_clear();
  lcd_i2c_put_curs(0, 0);

  DS18X20_init(); // PB0

  // switch RGB-LED completely off
  LEDr = 0;
  LEDg = 0;
  LEDb = 0;

  while(1)
  {
    if(line_valid == 1) // correct rx uart line received
    {
      solltemp = (float)atof((char *)line_buffer); // if uart sent line convert to new soll-temperature, 0 is
       assigned on failure
    }
    temp = get_temperature();
    lcd_i2c_put_curs(0, 0);
    sprintf(buffer, "IST-Temp:%6.2f", temp);
    lcd_i2c_send_string(buffer);
    lcd_i2c_put_curs(1, 0);
    sprintf(buffer, "Soll-Temp:%5.1f", solltemp);
    lcd_i2c_send_string(buffer);

    diff = solltemp - temp; // absolute difference
    // diff = (solltemp - temp)/temp * 100; // difference in percent
    if(diff < 0) // abs of difference
    {
      diff = diff * -1;
    }

    if(diff <= 3)
    {
      if(diff <= 1)
      {
        LEDr = 0;
        LEDg = 1; // full green
        LEDb = 0;
      }
      else
      {
        LEDr = 1; // mix of red and green
        LEDg = 1;
        LEDb = 0;
      }
    }
    else
    {
      LEDr = 1; // full red
      LEDg = 0;
      LEDb = 0;
    }

    delay_ms(5);
  }
}
```

Listing 3: Main-Funktion

6 Messungen

6.1 1-Wire-Temperatursensor

Messungen mit dem Logic-Analyzer:
Messinitiierung:

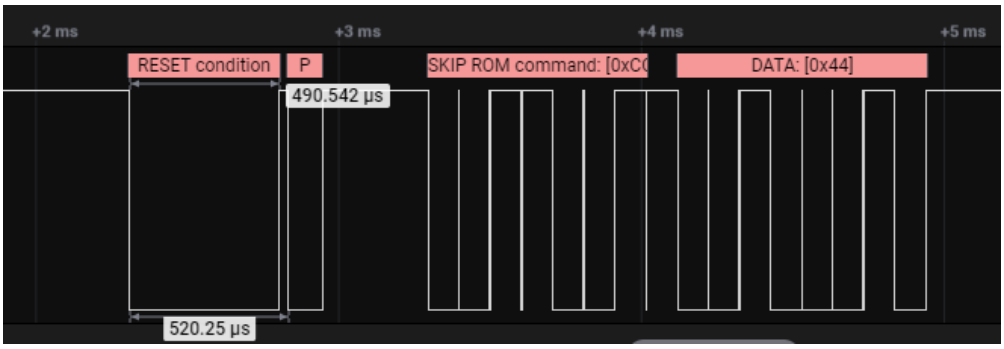


Abbildung 7: Messinitierung des Temperatursensors - Logic Analyzer

Start der 750μs Messdauer:

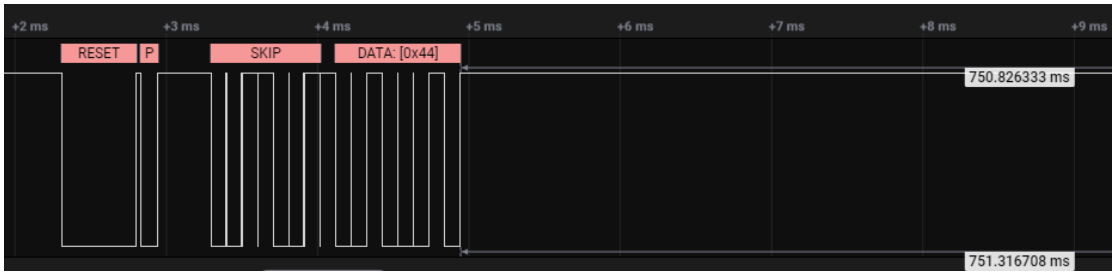


Abbildung 8: Start der Messdauer - Logic Analyzer

Auslesen der Temperatur aus dem Scratchpad:

Abbildung 9: Auslesen der Temperatur aus dem Scratchpad - Logic Analyzer

Messungen mit dem Oszilloskop:
Messinitiierung:

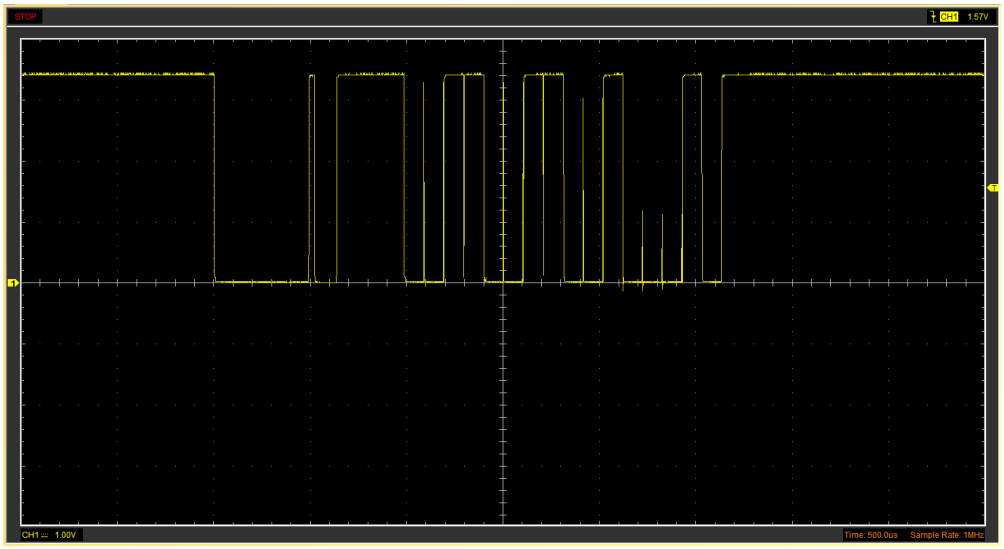


Abbildung 10: Messinitierung des Temperatursensors - Oszilloskop

Auslesen der Temperatur aus dem Scratchpad:

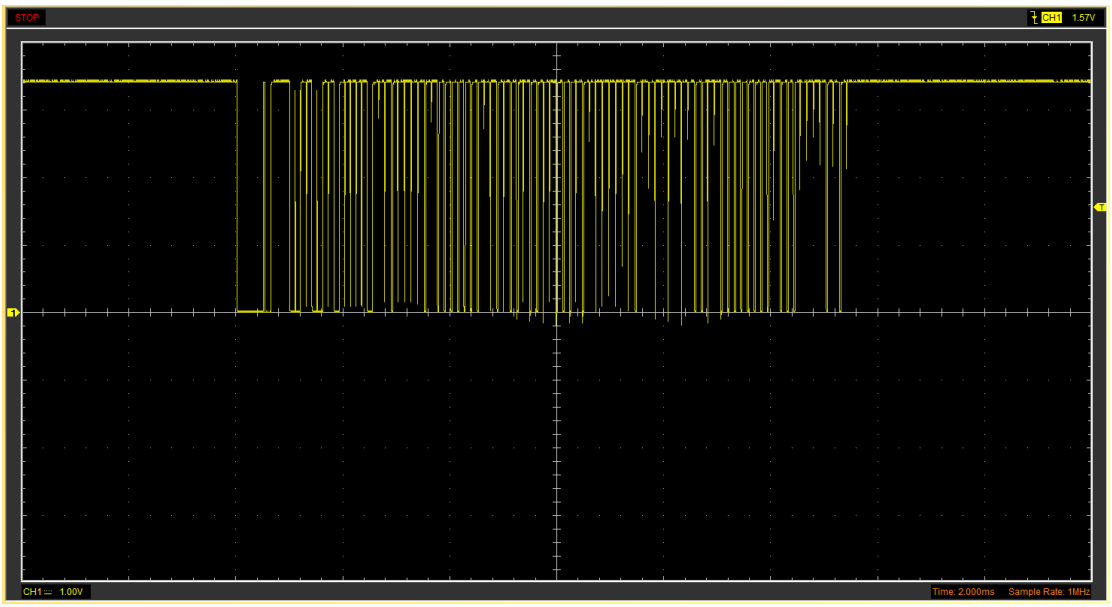


Abbildung 11: Auslesen der Temperatur aus dem Scratchpad - Oszilloskop

6.2 I²C/LCD Interfacing

Bemerkung: Die I²C-Messung wird nur mit dem Logic Analyzer durchgeführt.

Komplette I²C-Initialisierung:

Abbildung 12: I²C-Initialisierung

Teilausschnitt der Initialisierung, Sende den Befehl 0x30 ans Display:

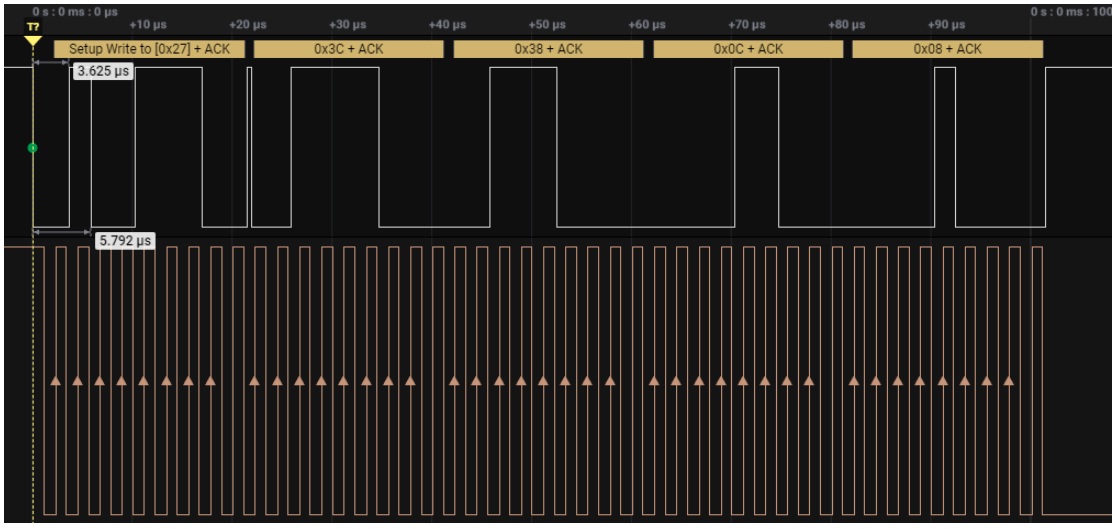


Abbildung 13: Teil-Ausschnitt der I²C-Initialisierung

7 Lessons Learned

Versorgungsproblem:

Nachdem die I²C-Schnittstelle für das LCD-Display fertig programmiert wurde, gab es Probleme mit dem Kontrast bzw. mit der Helligkeit des Displays. Der ausgegebene Text war schlecht erkennbar.
Die Lösung war es den Microcontroller nicht nur über die ST-Link-Schnittstelle anzuschließen, sondern zusätzlich über die USB Schnittstelle zu versorgen. Dadurch konnte die notwendige Stromzufuhr für das LCD-Display bereit gestellt werden.

Unnötige I²C-Pull-Up-Widerstände:

Der erste Hardware-Aufbau der I²C-Schnittstelle wurde mit zwei 10kΩ Widerständen durchgeführt. Eine nähere Begutachtung der Beschaltung des I2C/LCD-Interface Moduls im Datenblatt zeigte, dass dort entsprechende Pull-Up-Widerstände bereits vorhanden sind. Daher konnte auf die Widerstände verzichtet werden.

I²C-Messung mit dem Logic-Analyzer:

Der Logic-Analyzer bietet die Möglichkeit mit einer Auto-Mess-Funktion die übertragenen Daten direkt zu intepretieren. Diese Funktion wird bei der I²C-Messung verwendet, um die Übertragung der SDA- und SCL-Leitung genau nachzuvollziehen. Dabei gab es aber das Problem, dass die Auto-Messung die Daten nicht intepretieren konnte.
Das Problem lag in der Zuweisung der Leitungen: Die SDA- und SCL-Leitung wurden hardwaremäßig vertauscht an den Logic-Analyzer angeschlossen, daher konnte die Messsoftware die Leitung auch nicht interpretieren. Als die Leitungen richtig angeschlossen wurden, konnte die Messung ohne Probleme durchgeführt werden.

Dauer der Temperaturmessung:

Der Temperatursensor liefert zeitweise erst nach ca. 10 Minuten bis 1h nach der ersten Messung einen korrekten Temperaturwert. Daraus kann geschlossen werden, dass der Sensor eher träge auf grobe Temperaturschwankungen reagiert.

Abbildungsverzeichnis

Codelisting