Temperatur-Sensor selbstgebaut

Ein Temperatursensor mit integriertem AD-Wandler an der seriellen Schnittstelle (oder am USB-seriell Adapter):

Zur alten Version der Seite (LM75 am Parallelport; Auslesen unter DOS) geht's hier.

Wie messen wir mit möglichst geringem Aufwand Temperaturen mit dem PC? Bisher hätten wir einen temperaturabhängigen Widerstand als Spannungsteiler an einen AD-Wandler angeschlossen. Das ist aufwändig und muss geeicht und linearisiert werden. Inzwischen beglücken uns die Halbleiter-Hersteller mit Sensoren, in die der AD-Wandler schon integriert ist.

Auf dieser Seite wollen wir uns mit dem LM75 von Texas Instruments (ehemals National Semiconductor) beschäftigen. Der Sensor kann Temperaturen von -55°C bis +125°C bei einer Auflösung von 0,5°C messen. Der interne AD-Wandler benötigt etwas Zeit, ca. 10 Messungen pro Sekunde sind erreichbar. Es gibt eine 3,3V und eine 5V Version. Beide dürfen bei 3,0 - 5,5V betrieben werden. Die Typenbezeichnung gibt lediglich an, bei welcher Versorgungsspannung die Temperatur genau ausgegeben wird. Ich habe die 3,3V Version gewählt. Viele Schnittstellen geben nicht mehr Spannung ab und falls doch, ist es leichter die Spannung zu verringern, als sie zu vergrößern.

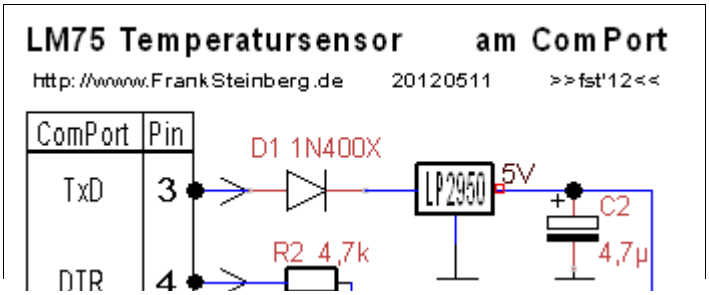
Der Datentransfer geschieht über das I²C-Bus Protokoll. Das erzeugen wir softwaremäßig mit zwei Ausgängen und einem Eingang der seriellen Schnittstelle.

- Keine separate Stromversorgung erforderlich.
- Aufbau mit nur sieben Bauteilen.
- Materialwert gut 2 Euro.

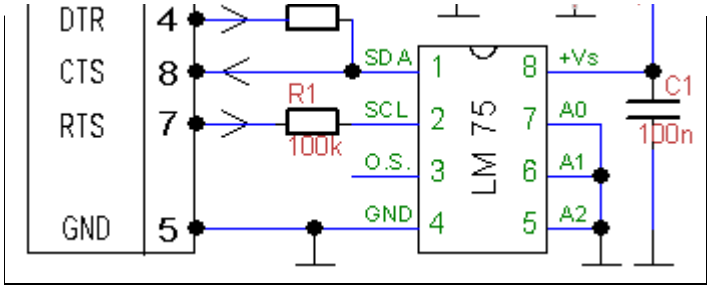
Warnung:

Wie immer: Alles, was nach diesen Veröffentlichungen nachgebaut wird, geschieht auf eigene Gefahr! Insbesondere können wir nicht voraussetzen, dass andere I²C-Bausteine die nötigen internen Schutzbeschaltungen gegen die negativen Spannungen der seriellen Schnittstelle besitzen.

Schaltungsbeschreibung:



Das IC erhält seine Versorgungsspannung (+Vs) aus einem Datenausgang (TxD Pin 3) der seriellen Schnittstelle. D1 schützt die Schaltung vor der negativen Spannung der Schnittstelle. Der Low-Drop Spannungsregler LP2950 reguliert die



Spannung auf 5V. Die minimale Differenz zwischen Ein- und Ausgang beträgt nur 40mV; wenn die Ströme klein sind. Und das sind sie hier. C1 entstört den AD-Wandler und muss so nah wie möglich am LM75 angebracht werden. C2 stabilisiert die Versorgungsspannung.

Über die bidirektionale Datenleitung (SDA = Serial Data) wird auf dem I²C-Bus gelesen und geschrieben. Mittels Pin 4 der seriellen Schnittstelle (DTR) schreibt unser PC auf den Bus. Die Daten vom IC werden über Pin 8 (CTS) in den PC gelesen. Beim I²C-Bus ist die Datenleitung permanent mit einem Pull-Up Widerstand nach +5V gezogen. Ein angeschlossener Baustein (Slave, unser LM75) sendet Daten, indem er die Leitung SDA nach Masse zieht (=logisch 0) oder eben nicht (=logisch 1). Die Pull-Up - Spannung erzeugen wir, indem wir den Ausgang DTR auf HIGH (positive Spannung) setzen, bevor der Slave Daten sendet. R2 reduziert die Pegel auf zulässige Werte.

Der Ausgang RTS (Pin 7) gibt den Takt (SCL = Serial Clock) auf den I²C-Bus. Hier sorgt R1 für die Reduzierung der Pegel. Die internen Schutzdioden des LM75 leiten die negativen Spannungspegel der seriellen Schnittstelle ab.

Je nachdem, wie wir A0-A2 entweder mit Masse oder mit +Vs verbinden, können wir die Adresse des ICs im Bus festlegen. Mit den drei Anschlüssen sind 8 verschiedene LM75 ansprechbar. Hier sind alle Pins mit Masse verbunden, die 3 Adressbits müssen dementsprechend auf 0 gesetzt werden.

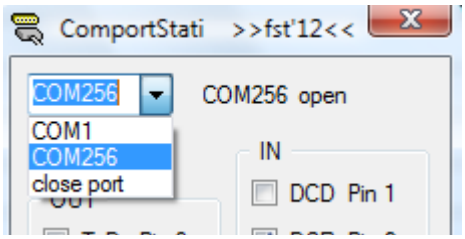
Den Schaltausgang (O.S.) benutzen wir hier nicht.

Bauteileliste mit Reichelt-Bezeichnungen:

Bauteil	Beschreibung	Preis in €
IC	LM 75 SMD 5 Volt-Version (LM 75CIM-5)	1,45
R1	Kohleschichtwiderstand 1/4W, 5%, 100 K-Ohm	-,11
R2	Kohleschichtwiderstand 1/4W, 5%, 4,7 K-Ohm	-,11
C1	X7R-5 100N Vielschicht-Keramikkondensator 100N, 10%	-,06
C2	RAD 105 4,7/100 Elektrolytkondensator, 105°C, RM 2,5mm	-,04
D1	1N 4001 Gleichrichterdiode, DO41, 50V, 1A	-,02
UReg	LP 2950 ACZ5,0 Festspannungsregler, +5,0V, 0,16A, TO-92	-,33
Gesamtpreis		2,12

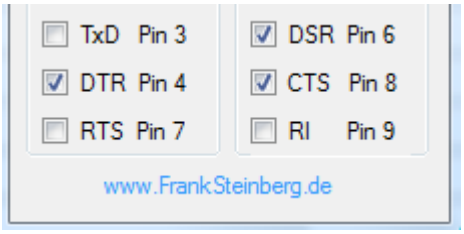
Prüfung:

Für erste Tests können wir das rechts abgebildete Programm benutzen. Damit können wir alle Pins gezielt setzen und auslesen:

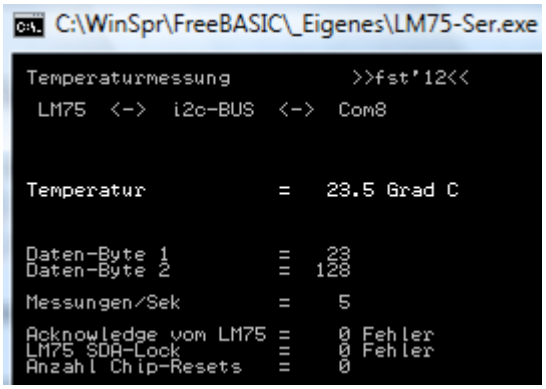


1. TxD aktivieren: An Pin 8 am LM75 müssen 5V anliegen.

Die Spannung eingeschaltet lassen.
2. DTR schalten: Pin 4 am LM 75 wechselt zwischen -0,7- 0V und 3-7,5V
3. RTS schalten: Pin7 am LM 75 wechselt zwischen -0,7- 0V und 3-5V. Das Häkchen CTS muss bei gesetztem RTS mitkommen.



Liegen die Spannungen außerhalb der Limits, können wir die Widerstände anpassen. R1 ist sehr unkritisch, R2 braucht evtl. mehr Zuwendung.



Jetzt starten wir das Beispielprogramm. Wir können als Parameter die Portnummer übergeben, z.B. "LM75-Ser.exe Com8". Andernfalls wird der Port abgefragt.

Das Programm sollte jetzt die Temperatur anzeigen. Bei Zimmertemperatur genügt es, mit dem Finger auf das IC zu tippen, um die Temperatur steigen zu lassen. Arbeitet die Schaltung nicht, wird -0,5 °C angezeigt (beide Datenbytes auf 255). In diesem Fall erstmal den Schaltungsaufbau genau kontrollieren, danach messen.

Im laufenden Betrieb können wir auf SDA und SCL nur schwankende Werte messen, weil dort die Spannung impulsweise anliegt. Mit einem einfachen piezokeramischen Schallwandler ohne Elektronik (z.B Conrad 75 16 59, € 1,51) können wir die Impulse hörbar machen. Ein sehr nützliches Utensil zur Fehleranalyse!

Das Beispielprogramm:

Der *.bas Quelltext ist für FreeBasic (getestet mit Version 0.23) unter Windows vorgesehen. Sie sind als Anregungen für eigene Projekte zu verstehen. Es werden WinAPI Funktionen zur Portansteuerung verwendet. Deshalb funktioniert es auch mit USB-Seriell-Wandlern. Der geringe Geschwindigkeit dieser Wandler im Bitbanging-Betrieb fällt hier kaum auf (5 Messungen/Sek gegenüber 10 am nativen ComPort).

Download Beispielprogramm Deutsche FreeBasic Seite FreeBasic Homepage

Programmierung:

Wer verstehen will, wie der LM75 programmiert wird, sollte sich das Datenblatt (Download im pdf-Format) zu Gemüte führen. Dort ist das Datenprotokoll ausführlich beschrieben. In `LM75-I2C.BAS` werden wir die dort verwendeten Begriffe wiederfinden. Ich habe den Quelltext ausführlich kommentiert. Wie wir die benötigten Pegel PC-seitig an der Schnittstelle erzeugen, steht im Schnittstellen-Tutorial.

Auf die Möglichkeiten, den LM75 zu konfigurieren, habe ich in den Beispielprogrammen verzichtet. Das ist nur dann erforderlich, wenn wir den LM75 als autarken Regler mittels des Steuerausgangs (O.S.) verwenden wollen. Ein Nachteil dabei ist, dass er alle Einstellungen verliert, sobald die

Versorgungsspannung fehlt. Der Beispielquelltext enthält alle Befehlssequenzen des I²C-Bus in separaten Unterprogrammen. Wir können uns zusätzliche I²C-Sequenzen leicht zusammenstellen.

Auf der Datenleitung SDA bedeutet ein high-Pegel (ca. 3-5V) eine logische 1 (Bitwert=1) ein low-Pegel (ca. 0V) eine logische Null (Bitwert=0). Der Pegel auf SDA wird geändert, während die Taktleitung SCL low ist. SDA wird gelesen, während SCL high ist. Der Master (=PC) steuert SCL und SDA. Bevor der Slave sendet, müssen wir SDA auf high setzen. Der Slave sendet seine Daten (-Bits) ja, indem er SDA auf Masse zieht oder high lässt. Im Folgenden ist beschrieben was passieren muss, um einen vollständigen Temperaturwert zu lesen

Master sendet STOP-Bedingung
SCL auf high setzen, dann SDA auf high setzen. Beim ersten Durchlauf hat STOP den Sinn, eine definierte Ausgangssituation zu erzeugen.
Master sendet START-Bedingung
Erst SDA auf low setzen, dann SCL. Damit sind die angeschlossenen Geräte (unser LM75) lesebereit.
Master sendet Adress-Byte
Das Byte muss in unserem Fall binär so aussehen: 1001 0001. 1001 ist der unveränderliche erste Teil der Adresse jedes LM75. Die folgenden drei Bits bestimmen wir damit, wie wir die Leitungen A0/A1/A2 verbunden haben. Laut Schaltplan sind alle mit Masse verbunden, deshalb sind drei Nullen erforderlich. Mit dem letzten Bit, sagen wir dem Slave, ob wir anschließend lesen oder schreiben wollen. Wir wollen vom Slave lesen, also ist der Bitwert 1. Das erste Bit gibt der Master sofort auf den Bus, indem er die SDA auf high setzt. Dann geht SCL auf high, anschließend auf low und das nächste Bit kann übertragen werden. Das ganze wiederholt sich insgesamt acht mal.
Slave sendet ACKNOWLEDGE
Nachdem der Master SCL ein weiteres mal auf high gesetzt, hat können wir der Pegel von SDA lesen. Hat der Slave SDA auf low gezogen, hat er damit den Empfang des Bytes quittiert (bleibt die Leitung high, wird in `LM75-I2C.BAS` eine Variable hochgezählt und angezeigt). Abschließend setzt der Master SCL wieder auf low.
Slave sendet das erste Datenbyte
Dazu folgt acht mal: SCL auf high setzen, SDA lesen (wenn high, Bitwert addieren), dann SCL wieder auf low setzen. Das erste Bit hat den Wert 128, das zweite 64, das dritte 32 usw.
Master sendet ACKNOWLEDGE
SDA auf low, dann SCL auf high, anschließend SCL wieder auf low. Damit quittiert der Master das Datenbyte und signalisiert Bereitschaft für ein weiteres Byte.
Slave sendet das zweite Datenbyte
Hier passiert dasselbe wie unter 5. Wir benötigen jedoch nur das erste (höchstwertige) Bit. Diese stellt den 0,5°C-Anteil dar. Die restlichen 7 Bits sind ohne Bedeutung.
Master sendet NO ACKNOWLEDGE
SDA auf high, SCL auf high, anschließend auf low. Das bedeutet, dass der Master das Byte empfangen hat, jedoch keine weiteren Daten wünscht.
Master sendet STOP-Bedingung
SCL auf high setzen, dann SDA auf high setzen. Jetzt sind wir wieder da, wo wir hergekommen sind.

Mehr Informationen:

Deutsche Beschreibung des LM75: http://www.goblack.de/desy/digitalt/i2c/lm75.html

Der I²C-Bus in der deutschen Wikipedia: http://de.wikipedia.org/wiki/I²C

Viel Erfolg beim Nachbau!
