

Messen von Temperaturen

– mit einem Microcontroller.

Sicherlich sind Temperaturmessungen ein wichtiger Punkt in der MSR (Messen – Steuern – Regeln) – Technik.

Wer z.B. eine Heizungssteuerung realisieren will oder eine Wetterstation aufbauen möchte, der muß auf jeden Fall eine oder mehrere Temperaturen kennen, um daraus etwaige Entscheidungen ableiten zu können.

Bei der Realisierung einer Heizungssteuerung muß man zumindest die Außentemperatur und die Vorlaufemperatur kennen, um eine Anpassung vorzunehmen. Baut man einen Microcontroller in die Steuerung ein, so kann man viel genauer und damit auch viel effektiver regeln. Letztendlich könnte man den Luftdruck mit in die Regelung einbeziehen – oder gar die einzelnen Zimmertemperaturen im Haus messen und eine individuelle Steuerung vornehmen. Die Änderung des Luftdrucks könnte dann als Maß für das zu erwartende Wetter in die Regelung eingebaut werden.

Bei der Wetterstation ist zumindest eine Temperatur von Interesse: die aktuelle Außentemperatur. Wer es jedoch etwas ausführlicher machen will, der kann verschiedene Temperaturen rund um das Haus auswerten. Temperatur in der Sonne, Temperatur im Schatten, Temperatur im Boden, Temperatur an verschiedenen Stellen rund um das Haus.

Wer eine ausgefeilte Ladestation für Akkus aufbauen möchte und dies ganz genau machen will, der sollte auch die Zellentemperaturen bei der Ladung berücksichtigen.

Alles in allem: Temperaturen sind interessante Meßwerte.

Die einfachsten menschlichen Prüfmethode sind unsere Haut und vor allem unsere Finger. Technisch gesehen ist dies aber eine äußerst ungenaue und subjektive Methode. Quantitativ – ob es nun 20 oder 20,4 Grad ist – dürfte keinem Menschen gelingen. Die Technik ist hier genauer und liefert reproduzierbare Werte.

Die klassische Methode ist der Einsatz eines Temperaturfühlers zum Messen der Temperatur. Hier gibt es eine Unzahl verschiedener Sensoren – NTC oder PTC (negativer oder positiver Temperaturkoeffizient), Thermolemente usw. Je nach Güte und je nach Material –letztendlich auch nach Preis- sind diese Sensoren in der Reproduzierbarkeit und damit in der Genauigkeit unterschiedlich.

Wir reden hier über den Einsatz dieser Fühler in der Analogtechnik. Nachteil aller Sensoren ist die mehr oder weniger große Unlinearität bei verschiedenen Temperaturen. Diesem Mangel kann jedoch mit verschiedenen Strategien entgegengewirkt werden. Die elektronische Kompensation mit Hilfe von Operationsverstärkern oder Transistoren ist sehr häufig anzutreffen. Sie hat den Nachteil : je genauer desto teurer.

Die zweite Methode ist die Eichung in Software oder Tabellenform. Man mißt den Nullpunkt mit Hilfe von Eiswasser, mißt die elektrischen Werte und gibt diese in eine Tabelle. Man mißt die Temperatur im kochenden Wasser und liest die elektrischen Werte ab. Hier muß man natürlich noch die Höhe über N.N. berücksichtigen – auch dieser Wert kommt in die Tabelle.

Dann benötigt man ein genaues Thermometer, um alle Zwischenwerte zu notieren. Dieses ist natürlich ein irrsinniger Aufwand, der nur von Enthusiasten zu leisten ist. Doch glücklicherweise gibt es die Eichkurven zu den Sensoren im Datenblatt. Meistens findet man

auch eine Umrechnungstabelle vor, mit der man für viele Anwendungen eine ausreichende Genauigkeit in einem Bereich realisieren kann.

Wer die CControl einsetzt und Temperaturen messen will, der wird schon einmal die dazu angebotenen Temperatursensoren incl. der Kompensationselektronik gesehen haben. Ein solcher direkt anschließbarer Sensor kostet immerhin fast 50 Mark.

Die versprochene Genauigkeit von 0.5 Grad kann ich nicht bestätigen. Zwei identische (ohne daran gefummelt zu haben) Sensoren lieferten Werte von +/- 1.5 Grad Unterschied.

Die Betriebsspannung dieser Sensoren liegt bei 6.5 Volt, die Referenzspannung bei der CControl liegt bei 2.5 Volt.

Der weitere Nachteil beim Einsatz dieser Temperatursensoren ist die Einbindung einer Eichentabelle. Da 255 Werte mit je 2 Byte in einer Integerzahl gespeichert werden, belastet dies die CControl mit 510 Byte. Führt man Eichentabellen für Luftdruck, Schallpegel, Hygrometer u. mehr mit, so hat man einige Kilobytes schon für das Programm selbst vorab aufgezehrt.

Wir werden nachher sehen, dass es auch ganz ohne Tabellen geht.

Wer nun, wie oben beschrieben, mehrere Temperaturen messen will und nicht mit dem einen Sensor in der Hand von A nach B wandern möchte, der kann dann einiges investieren. Vier Temperaturen machen nach Adam Riese glatt 200 Mark.

Wir verlassen jetzt den analogen Sektor und wenden uns dem kostengünstigen digitalen Bereich zu.

Dass es überhaupt digital geht, ist noch ziemlich unbekannt; zumindest machte ich diese Erfahrung.

Und es geht gleich mehrmals digital: Es gibt Sensoren für den Eindrahtbetrieb (in Wirklichkeit sind es zwei, denn Masse muß auch gelegt werden). Diese interessante Methode werde ich später einmal beschreiben.

Es gibt Sensoren für den Zweidraht- und Dreidrahtbetrieb. Jeweils natürlich auch wieder plus eine Leitung für die Masse.

Beschrieben werden soll hier die Zweidrahtmethode über einen seriellen Bus.

Damit ist das Stichwort gefunden. Es ist wieder der schon bekannte I2C – Bus. Auch hier gibt es verschiedene Anbieter (National LM75) und (Dallas DS1621), vielleicht noch andere.

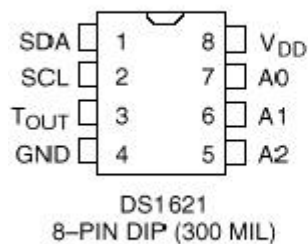
Beide Chips kosten mit ca. 6.- DM und stellen ein komplettes Temperatursystem mit zusätzlichen Features (z.B. Thermostat) dar. Für den Geldbeutel ist die Realisierung von mehreren Temperaturstationen daher besonders freundlich. 4 mal 6 – das geht noch ohne Taschenrechner = 24.- DM.

Beide Chips habe ich ausprobiert und erfolgreich getestet.

Für die Beschreibung hier habe ich den DS1621 favorisiert, da er in 8 DIL (Dual in line - also für Fassungen geeignet) käuflich ist. Den Chip von National habe ich bisher nur als SMD - Version gesehen. SMD bedeutet Surface mounted device.

Für den Betrieb ist die SMD – Version natürlich günstiger, da durch die geringere Masse des Chips schneller auf Temperaturen reagiert wird. Für den Normallötter ist aber die SMD – Technik nicht immer so perfekt zu beherrschen.

Ich hoffe, dass ich da auch Ihren Geschmack treffe und die Bastlerversion (Gehäuse und auswechselbarer Chip) treffe. Wer gut löten kann, der kann natürlich auch die SMD-Version wählen – elektrisch unterscheidet sie sich nicht.



PIN DESCRIPTION

SDA	–	2-Wire Serial Data Input/Output
SCL	–	2-Wire Serial Clock
GND	–	Ground
T _{OUT}	–	Thermostat Output Signal
A0	–	Chip Address Input
A1	–	Chip Address Input
A2	–	Chip Address Input
V _{DD}	–	Power Supply Voltage

Ich habe hier versucht, die Realisierung der Temperaturmessung mit einem I2C-betriebenen Temperatursensor einmal ganz genau zu beschreiben.

Für viele geübte Elektroniker sind hier Dinge beschrieben, die sie sicherlich überspringen können.

Für alle Anfänger, die sich noch nicht so recht mit der Microcontollertechnik auskennen, werde ich Schritt für Schritt vorgehen. Natürlich kann man nicht immer bei Adam und Eva anfangen.

Einige Dinge werden aus Redundanzgründen wiederholt. Ich setze voraus, dass die CControl vorhanden ist und der DS1621 aufgebaut ist. Bei meiner Schaltung (Lallus) ist der Digitalport 8 mit dem data - Signal versehen, Die Clockleitung ist auf dem Digitalport 7.

I2C – das bedeutet immer, dass es eine Systemadresse gibt, auf die der Chip reagiert. Weiterhin sind drei Eingänge codierbar, so dass sich maximal 8 dieser Chips im System ansprechen lassen. Diese Eingänge sind mit A0, A1, A2 bezeichnet.

Damit wäre eigentlich schon alles gesagt.

Diesen Wissensstand hatte ich, als ich mit dem Probieren begann. Ich konnte außer dem Datenblatt keinen dazu befragen, da dieser Chip in der normalen Lötpraxis recht unbekannt ist. Letztendlich habe ich dann doch zwei Abende herumgetüftelt, um die Spezialitäten in das Programm zu formen.

Das ging dann soweit, dass ich aus dem Tiefkühlfach eine Spinatpackung herausholte, um das Verhalten bei Minustemperaturen zu ergründen.

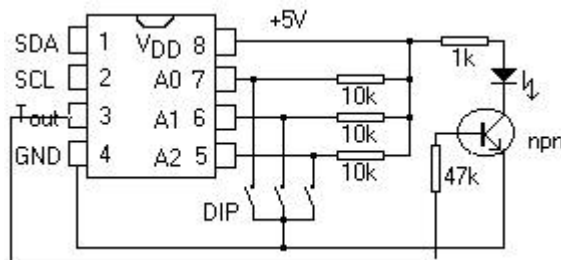
Da das Experimentieren so schön war, dauerte es einige Zeit. Zuerst habe ich noch Temperaturen von -16 Grad gemessen, dann ging der Chip nur noch auf -14 Grad. Am späten Abend dann hatte ich Mühe noch Minuswerte zu messen.

Als ich dann stolz von meinen Erfolgen berichtete und noch erwähnte, dass jetzt nur noch Temperaturen von $+3$ Grad gemessen werden, gab es am nächsten Tag die Quittung: es gab Spinat mit Ei als Mittagessen.

Es kann jetzt losgehen.

Ich kaufte bei Conrad in Köln die Chips – einmal LM75 und 3mal DS1621. Alles verstaute ich auf einer Europakarte mit Lochraster. Da ist dann übrigens noch viel Platz frei.

Der SMD-Chip machte ein bißchen Probleme beim Anlöten, letztlich ging es aber doch ganz gut. Er erhielt (weil es am einfachsten war) die Adressierung 000, also alle drei Adreßbeinchen A0, A1, A2 auf Masse gelegt. Die beiden anderen Chips DS1621 kamen in eine 8-Pin DIL –Fassung. Alle Chips habe ich an den Rand der Platine gesetzt, damit man später Temperaturunterschiede (siehe Spinat) besser ausprobieren kann.



Die beiden DS1621 bekamen zur Adressierung ihrer Adresse kleine Dipschalter. Damit kann man dann bequem experimentieren. Bei offenem Schalter sollten die Adresseingänge mit $10\text{ k}\Omega$ nach plus 5 V gepullt werden. Also kamen pro Chip 3 Widerstände hinzu.

Bei beiden Chiparten gibt es noch einen Ausgang, den man als Thermostat mit Hystereseeigenschaften benutzen kann. Diesem Ausgang habe ich einen Transistor und eine Leuchtdiode spendiert, um beim Experimentieren auch eine optische Anzeige zu haben.

Plus 5 Volt – ist klar, die Masse – ist klar. Und dann noch die beiden seriellen Taktleitungen vom I2C-Bus (SDA und SCL) an die Pins angeschlossen und schon kann es los gehen.

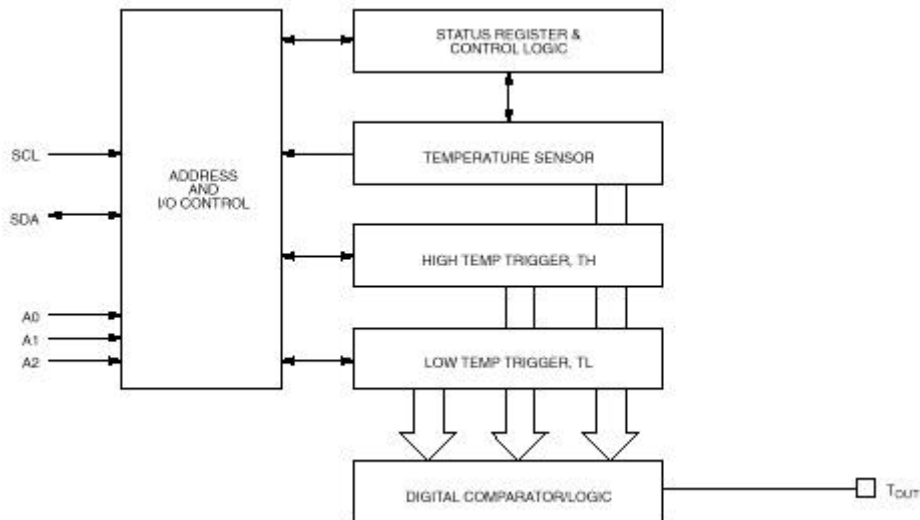
Bei Lallus sind die Digitalports 8 (SDA – vielmehr als data bezeichnet) und der Digitalport 7 (SCL – hier als clock bezeichnet. Diese werden also im Programm zunächst einmal definiert.

P.S: Die Bezeichnungen wurden nicht willkürlich anders als im Datenblatt gewählt. SDA und SCL sind reservierte Befehle, da diese innerhalb der CControl schon für die Kommunikation mit dem seriellen EEPROM 2465 benutzt werden.

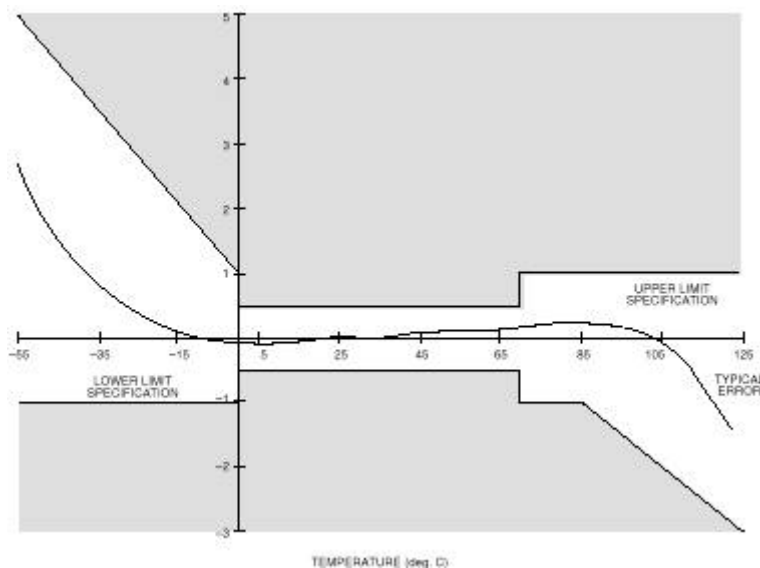
Es gäbe Fehlermeldungen, wollte man sie so bezeichnen. Übrigens für Fortgeschrittene: der I2Cbus ist im System auffindbar (siehe Beschreibung der CControl). Die beiden Ports SDA und SCL liegen im digitalen Datenregister A, das von Basic aus nicht erreichbar ist. Von Assembler aus geht es aber)

In dieser Beschreibung sind einige kleine Programme lauffähig beschrieben. Sie können die Programme markieren und sie direkt in den Editor der CControl bringen. Natürlich müssen die Ports und die Variablen vorher deklariert werden.

Funktionsschema der Temperaturmessung mit dem DS1621



Ein Auszug aus dem Datenblatt zeigt den Verlauf der Genauigkeit an. In dem Bereich zwischen 0 und 70 Grad Celsius sieht es wirklich sehr gut aus.



Deklaration der digitalen Ein- Ausgänge

```
Define data port [8]
Define clock port [7]
```

Damit sind die beiden Digitalports aktiv als Ausgang oder Eingang geschaltet.

Vielleicht ist hier die Zeit gegeben, wo ich schon einmal auf die Funktion `deact port` hinweise. Bei meinen Versuchen tüftelte ich nämlich lange im Trüben herum, bis ich auf diesen Befehl stieß. Ich hatte in der Zwischenzeit sogar den Zweidrahtbus zu einem Dreidrahtbus erweitert. Es lag an dem Befehl `deact`. Wenn ich den Port data z.B. auf 1 oder on schalte, so ist er als Ausgang definiert. Werden dann Polaritätswechsel vom Chip vorgenommen, so hängt er in der aktiven Position und bekommt diese Wechsel nicht mit.

In meiner Not habe ich mir damals damit geholfen, dass ich diese Wechsel über eine Diode an einem weiteren Port gemessen habe – deshalb meine Dreidrahtmethode. Das hat mich sehr geärgert und einige Tage Schweiß gekostet. Dann kam der Befehl deact vor die Augen und alles wurde klar.

So ist es, wenn man die Bedienungsanleitung nicht sorgfältig genug liest. Deshalb hier an dieser Stelle schon einmal mein Erfahrungsbericht. Sie werden später noch sehen, wo dieser deact – Befehl zum Einsatz kommt. Deact – deaktiviert also den Port, so dass er auch als Eingang benutzt werden kann.

Wie oben schon angedeutet, ist eine Spezialität des I2C-Busses die Adressierbarkeit des Chips. Jede Chipfamilie hat eine eigene Adresse. Der I/O – Expander PCF8574 zum Beispiel hat die Adresse hex40 oder dezimal 64.

Die Temperatursensoren von National und von Dallas besitzen die Systemadresse hex 90 oder $9 \cdot 16 = 144$ als dezimale Adresse. Hinzu kommen dann noch die Chipadressen, wie oben angedeutet. 3 Bits sind dafür reserviert u.zwar Bit 3, Bit 2, Bit 1. Bit 0 ist reserviert für die Wahl: Schreiben oder Lesen. Dazu kommen wir später noch.

Ich weiß nicht, wie gut Sie in der Bitmanipulation sind. Deshalb hier ein paar Hilfestellungen, die für alle Ungeübten notwendig sind.

Ein Byte besteht bekanntlich aus 8 Bits.

Sagen wir, das Byte wäre binär so aufgebaut: 1 0 0 1 0 0 1 0.

Was heißt dieses nun in unserer besser zu verstehenden Dezimalsprache? Wir müssen noch wissen, wie wir zählen müssen. Dafür gibt es den Begriff MSB und LSB. In der hier vorliegenden Technologie arbeiten wir mit dem MSB links, dann 6 Datenbits und dann rechts das LSB.

MSB – das ist das most significant Bit und LSB entsprechend das least significant Bit eines Bytes. MSB bedeutet daher die höchste Wertigkeit – also 128. LSB entsprechend das letzte Bit rechts – 1.

Damit ergibt sich die Wertigkeit der 8 Bits eines Bytes

128 64 32 16 8 4 2 1

MSB first bedeutet also bei der Entschlüsselung der Binärzahl: das linke Bit entspricht 128 - wenn es gesetzt ist -.

1 0 0 1 0 0 1 0

In unserem Falle können wir also 128 addieren.

Das nächste Bit nach rechts ist eine 0. Der Wert 64, der dieses Bit hätte, wenn es gesetzt wäre, wird nicht addiert, da eine 0 vorliegt.

Bleibt bei 128.

Das nächste Bit – Wert 32 - ist 0 – es bleibt bei 128.

Das nächste Bit – Wert 16 - ist gesetzt, also $128 + 16 = 144$,

Das nächste Bit (Bit 3, denn man zählt von rechts nach links, beginnend mit 0,1,2,4,8,16,32,64,128) ist wieder 0, Bit 2 ist 0, Bit 1 ist 1 und hat den Wert 2.

Wir waren bei 144, kommt noch 2 dazu – macht 146.

Dieses ist der dezimale Wert unseres Bytes von oben. Hexadezimal bedeutet dies &H92 in der CControlsprache.

```
1 0 0 1 0 0 1 0
128+0+0+16+0+0+2+0 = 146
```

Dieses Beispiel ist auf unseren Chip gut anzuwenden. Die Systemadresse &H90 steckt in den ersten 4 Bits links – die 144 – sie gilt für alle Temperaturchips.

Bit 1 (von rechts gelesen) sitzt auf 1 – hat also den Wert 2.

Dieses bedeutet, dass er erste Chip diese Adresse erhält. Der Schalter 1 am Adresseingang wird geöffnet und liegt über den Widerstand von 10 K an 5 Volt, die beiden anderen Schalter liegen auf 0 Volt.

Zum Verständnis: den Nationalchip, wie oben beschrieben, habe ich komplett mit seinen Adressen auf 0 gelegt. Hier gilt die Adresse 1 0 0 1 0 0 0 0.

Der zweite DS1621 erhält wiederum eine neue Adresse. Sagen wir 1 0 0 1 0 1 0 0.
Der Übung halber machen wir einfach alle Möglichkeiten durch:

```
Chip1 1 0 0 1 0 0 0 0
Chip2 1 0 0 1 0 0 1 0
Chip3 1 0 0 1 0 1 0 0
Chip4 1 0 0 1 0 1 1 0
Chip5 1 0 0 1 1 0 0 0
Chip6 1 0 0 1 1 0 1 0
Chip7 1 0 0 1 1 1 0 0
Chip8 1 0 0 1 1 1 1 0
```

Da wir später die Adresse des Chips eventuell wechseln wollen, definieren wir im Programm die Adresse als Variable.

```
define data    port [8]
define clock  port [7]

define adresse byte
```

```
adresse = &H90 ( oder Adresse = 144) ( Grundadresse bei 3 Eingängen auf 0)
```

Was wir noch wissen müssen: das erste Bit ganz rechts mit dem Wert 1 oder 0 dient zur Signalisierung des Datenweges. Soll geschrieben werden oder soll gelesen werden? In unserem Falle ist es weitgehend das Lesen einer Temperatur, obgleich wir dem Chip auch einiges schreibend mitteilen müssen.

Zum Beispiel schreiben wir die Adresse und schreiben Kommandos für die verschiedenen Register des Chips. Später finden Sie diesen Schreibvorgang unter der Subroutine PUTBYTE. Im Gegensatz dazu existiert die Leseroutine GETBYTE.

Dem Chip wird eine Schreiboption mitgeteilt mit Bit 0 = 0
Dem Chip wird eine Leseoption mitgeteilt mit Bit 0 = 1.

Dazu wieder zwei Beispiele:

1 0 0 1 0 0 1 0 Temperatursensor auf Adresse 144 + 2 soll schreiben

1 0 0 1 0 0 1 1 Temperatursensor auf Adresse 144 + 2 soll lesen

Jetzt kommt der Teil der Datenübertragung. Denn diese Adresse muß dem Chip mitgeteilt werden, damit er überhaupt aufwacht und merkt, dass er etwas tun soll.

Dazu benötigen wir die Routinen des I2C-Protokolls. Dieses Busprotokoll kommt dem einen oder anderen vielleicht etwas spanisch vor, wenn er zum ersten Male damit konfrontiert wird.

Später, nach einer Gewöhnungszeit empfindet man diesen Bus als ideal für die Bastelarbeiten.

Der große Vorteil des I2C-Protokolls liegt im unproblematischen Timing. Wenn ich heute drei Takte auf den Bus gebe und morgen einen und übermorgen wieder einen, so ist dies für den Bus völlig gleichgültig.

Dieser große Vorteil hat auch seinen Preis. Der I2C – Bus ist kein Geschwindigkeitsweltmeister.

Der I2C-Bus – in seiner Standardausführung - schafft 100 kHz. Da man für einen kompletten Lesezyklus fast 30 Takte benötigt kommt man auf maximal 3 kHz. Das wäre z.B. für eine Bildübertragung natürlich viel zu wenig.

Für unsere Anwendung zur Temperaturmessung reicht es völlig aus. Die neuere Generation von I2C-Chips ist übrigens schneller: 400 kHz und die neueste Version soll 3,4 MHz takten können. Soweit nur ein Ausflug in die Technik der I2C-Welt.

Die Datenkommunikation CControl → DS1621 beginnt immer mit einem Startsignal. Dieses ist ein Wechsel auf der Dataleitung von 1 nach 0 bei clock=1. Im Programm kann man sich dies ansehen.

```
Clock = 1
Data   = 1
Data   = 0
Clock  = 0
```

Also : immer ein Startsignal zur Einleitung der Kommunikation.

Was passiert?

Alle I2C-Chips (DS1621 oder LM75 oder auch PCF8574 oder auch I2C EEPROMS 2465) machen die Ohren auf: auf dem Bus ist was los, mal hören, wer und was gemeint ist.

2. Vorgang : Senden des Adressbytes incl. Richtungsbit (Lesen/Schreiben).

Dieses geschieht nun –wie bei einem seriellen Bus zu erwarten- bitweise. Das erste Bit der Adresse, das MSB-Bit ist eine 1

1 0 0 1 0 0 1 0 0

Der Dataport wird also eingeschaltet mit `data=1` oder `data=on`. Jetzt liegt das Bit an.

Ein Wechsel auf der Clockleitung von 1 nach 0 bedeutet die Übernahme der 1 von data in den Temperaturchip DS1621.

Übernahme eines Datenbits mit Wert 1:

```
Data=1 : clock = 1 : clock = 0 : clock=1
```

Die Clockleitung kann auch mit dem Befehl pulse ihren Potentialwechsel vornehmen. Ist clock 1 und es wird gepulst, so wird ein Wechsel von 1 nach 0 für einen kurzen Zeitraum vorgenommen; danach nimmt clock wieder 1 an. Dies aber später.

```
data = 1 : pulse clock
```

Bit 7 ist also geschiftet, der DS1621 hat es übernommen. Jetzt kommt Bit 6 dran. Es ist eine 0. Data wird auf 0 gelegt, ein Puls oder ein Wechsel auf der Clockleitung von 1 nach 0 shiftet Bit 6 raus. 8 Bit machen ein Byte: also wird dieser Vorgang bis zum Bit 0 wiederholt. Der Vorgang kann also so dargestellt werden:

```
Data = 1 : pulse clock 'Systembit mit Wertigkeit 128
Data = 0 : pulse clock 'Systembit mit Wertigkeit 64 = 0
Data = 0 : pulse clock 'Systembit mit Wertigkeit 0 = 0
Data = 1 : pulse clock 'Systembit mit Wertigkeit 16
Data = 0 : pulse clock 'Adressbit mit Wertigkeit 8 = 0
Data = 0 : pulse clock 'Adressbit mit Wertigkeit 4 = 0
Data = 1 : pulse clock 'Adressbit mit Wertigkeit 2
Data = 1 : pulse clock 'Richtungsbit Read / Write
```

Das Adressbyte 1 0 0 1 0 0 1 1 ist gesendet. Der Chip hat die Signale empfangen. Welcher Chip? Klar, der DS1621 mit der Systemadresse 144 und der Eingangsadresse 2 (erster Schalter geöffnet). Alle anderen Chips können sich wieder schlafen legen.

Die nächste Stufe: die CControl möchte gerne wissen, ob alles geklappt hat und ob sie nun weitermachen kann. Ich sagte ja oben, dass der Vorteil im zeitunkritischen Verhalten des Busses liegt. Alle Befehle können im Tagesrhythmus (was natürlich keinen Sinn macht) gegeben werden.

Hierzu gibt es den in der Technik häufig benutzten Begriff: acknowledge. Jetzt wird der Temperaturchip aktiv. Er macht nach dem Empfang von 8 Datenbits ein acknowledge. Da es für die CControl bedeutet, dass sie etwas wissen will habe ich das Unterprogramm GETACK genannt. Im Gegensatz dazu gibt es noch GIVEACK, das wir später noch kennenlernen.

Das acknowledge besteht darin, dass der Temperaturchip die Dataleitung nach 0 zieht. Dafür muß die Dataleitung auf 1 liegen, also `data =1`.

Hier kommt nun der oben schon beschriebene Befehl `deact data` zum Tragen. Die Datenleitung ist mit dem Befehl `data=1` auf Ausgang geschaltet und würde das Herunterziehen nicht mitbekommen.

```
Data = 1
Deact data
```

Jetzt wartet der Prozessor auf diese Aktion. Im Sprachumfang der CControl gibt es den Befehl `wait`, der hier zum Einsatz kommt. Da auf `data = 0` gewartet wird, muß die Wartefunktion mit einem `not` ergänzt werden.

```
data = 1
eact data
```

```
wait not data
```

Für alte Basicprogrammierer: den Befehl `wait` könnte man auch umschreiben, was aber sinnlos ist.

```
#warte  
if data =1 then goto warte
```

Danach wird das `acknowledge` wieder mit einem Clockpuls oder einem Clockwechsel von 1 nach 0 bestätigt.

Damit ist die Unterfunktion `GETACK` fertig, ein `RETURN` zur Rückkehr wird angefügt.

```
#GETACK  
data = 1  
deact data  
wait not data  
pulse clock      oder Clock=1:clock=0  
RETURN
```

Jetzt sind wir erst soweit, dass wir dem Chip irgend etwas vernünftiges an Daten schicken können.

Es kommt das erste Datenbyte. Um es vorwegzunehmen: Wenn es bei einem Datenbyte bleibt, so erfolgt das `STOP` – Kommando und der Zyklus wäre beendet.

An dieser Stelle entscheidet es sich jetzt auch, ob wir lesen (`GETBYTE`) oder schreiben wollen (`PUTBYTE`). Mitgeteilt wurde dies dem Chip ja mit dem ersten Adressbyte mit Bit 0.

Um den Temperaturchip DS1621 in einen bestimmten Modus zu versetzen, muß er am Anfang in einen bestimmten Zustand versetzt werden. Zur genaueren Beschreibung kommen wir später noch.

Sagen wir einfach, wir hätten ihn dazu veranlaßt, dass er die Register so einstellt, dass wir ständig die aktuelle Temperatur auslesen können. (Vorab: wir senden `&HAA` und stellen eine konstante Wandlung ein)

Mit dem Adressbyte haben wir als Bit0 eine 1 gesendet, was ja bedeutet: wir wollen lesen

Gesendet: 1 0 0 1 0 1 1.

Acknowledge ok

Jetzt kommt die Funktion `GETBYTE`, denn wir wollen das Ergebnis in Form eines Temperaturwertes erhalten. Dazu benötigen wir eine neue Variable, die wir `I2Caus` nennen wollen. Sie muß am Anfang mit `define I2Caus byte` deklariert werden. In diese Variable wird der Wert geschrieben. Da die Unterfunktion mehrmals aufgerufen wird, sollte `I2Caus` zunächst einmal definiert auf 0 gesetzt werden.

Jetzt soll das Datenbyte gelesen werden. Durch das Mitsenden von Bit 0 = 1 im Adressbyte weiß der Temperaturchip, dass er aktiv werden muß und die Ergebnisse zur `CControl` liefern muß. Und dieses wieder bitweise und 8 mal, damit ein Byte komplett ist.

Für Anfänger im Programmieren ist der Befehl shl, wie er unten in der Subroutine zu sehen ist, vielleicht etwas gewöhnungsbedürftig. Was muß geschehen? Sie erinnern sich: die Technologie basiert auf MSB first – also zuerst wird das höchstwertige Bit mit der dezimalen Wertigkeit 128 gesendet usw.

Ist bei diesem Vorgang also das erste gesendete Bit auf der Dataleitung eine 1, so müssen wir 128 notieren. Ist das zweite gesendete Bit eine 1, so kommen 64 dazu usw.

Wir könnten dies in der alten Basicmanier primitiv programmieren

```
For i=1 to 8
If i=1 and data =1 then i2caus=i2caus+128
If i=2 and data =1 then i2caus=i2caus+64
If i=3 and data =1 then i2caus=i2caus+32
If i=4 and data =1 then i2caus=i2caus+16
If i=5 and data =1 then i2caus=i2caus+8
If i=6 and data =1 then i2caus=i2caus+4
If i=7 and data =1 then i2caus=i2caus+2
If i=8 and data =1 then i2caus=i2caus+1
Next i
```

Wenn wir in unserem Programm so weitermachen würden, dann wäre der knappe Speicher bald aufgebraucht. Dieses geht auch eleganter. Eben mit dem Befehl shl oder shr.

Shl bedeutet shift left, shr eben shift right – und zwar das anstehende Byte.

Wer Assembler programmiert (dahin werde ich Sie auch noch führen, aber nicht in diesem Artikel), der kennt dieses shiften aus dem Effe. Hier nur einige Ausblicke auf die Assemblersprache des 6505. Es gibt lsl und lsr, was bedeutet: logical shift left accu und logical shift right accu, oder rol und ror: rotate left, rotate right oder asl und asr; arithmetic shift left, arithmetic shift right.

Was bedeuten diese Befehle shl und shr in unserem Falle?

```
1 shl 0 = 2 ^ 0 = 1
1 shl 1 = 2 ^ 1 = 2
1 shl 2 = 2 ^ 2 = 2 * 2 = 4
1 shl 3 = 2 ^ 3 = 2 * 2 * 2 = 8
1 shl 4 = 2 ^ 4 = 2 * 2 * 2 * 2 = 16
1 shl 5 = 2 ^ 5 = 2 * 2 * 2 * 2 * 2 = 32
1 shl 6 = 2 ^ 6 = 2 * 2 * 2 * 2 * 2 * 2 = 64
1 shl 7 = 2 ^ 7 = 2 * 2 * 2 * 2 * 2 * 2 * 2 = 128
```

Das ist das, was wir benötigen.

Wir lassen einen Schleifenzähler 8 mal laufen (von 7 bis 0 mit Dekrementierung step -1)

Das, was wir schreiben wollen, nennen wir **schreibe**, eine Variable, die am Anfang als Byte declariert werden muß.

```
declare schreibe byte
```

```
for i= 7 to 0 step -1
if (schreibe and 1 shl i) = 0 then data = 0
if (schreibe and 1 shl i) <> 0 then data =1
next i
```

Beim shr – Befehl ist dies genau anders. Zuerst wird das rechte LSB – Bit geshiftet, danach kommen die anderen Bits bis zum MSB dran.

Hier wird wieder ein neuer Datentyp eingeführt: das AND, das logische UND. AND bedeutet, dass beide Vergleichswerte eine 1 besitzen müssen, damit der Ausgang ungleich 0 ist.

Das erste Byte schreibe = 0 0 0 0 0 0 0 1 = 1
Der Ausdruck schreibe AND 1 shl 7 = 1 0 0 0 0 0 0 0 = 128

Beide Werte `verANDet` ergibt eine 0, da nicht beide linken Bits 1 sind.

Das erste Byte schreibe = 1 0 0 0 0 0 0 0 = 128
Der Ausdruck schreibe AND 1 shl 7 = 1 0 0 0 0 0 0 0 = 128

Beide Werte `verANDet` ergibt 128, da beide linken Bits 1 sind und die Wertigkeit 128 beträgt.

Das erste Byte schreibe = 1 0 0 0 0 0 0 1 = 129
Der Ausdruck schreibe AND 1 shl 7 = 1 0 0 0 0 0 0 0 = 128

Beide Werte `verANDet` ergibt wieder 128, da beide linken Bits 1 sind und die Wertigkeit des 8. Bits 128 beträgt.

Das erste Byte schreibe = 1 1 1 1 1 1 1 1 = 255
Der Ausdruck schreibe AND 1 shl 7 = 1 0 0 0 0 0 0 0 = 128

Schreibe AND 1 shl 7 ergibt wieder 128.

Natürlich kann man dies mit den anderen Bits genauso darstellen.
Für das Bit 7 z.B.

Das erste Byte schreibe = 1 1 1 1 1 1 1 1 = 255
Der Ausdruck schreibe AND 1 shl 6 = 0 1 0 0 0 0 0 0 = 64

Schreibe AND 1 shl 6 ergibt 64

Das erste Byte schreibe = 1 0 1 1 1 1 1 1 = 191
Der Ausdruck schreibe AND 1 shl 6 = 0 1 0 0 0 0 0 0 = 64

Schreibe AND 1 shl 6 ergibt 0

Wenn Sie mit den Bitmanipulationen noch nicht so erfahren sind, so sollten Sie das einfach an der CControl ausprobieren. Ich gebe hier ein kleines Beispielprogramm:

Sie können das untenstehende Programm einfach markieren und direkt in das Windows-Editierprogramm der CControl kopieren.

```
define wert1 byte
define wert2 byte
define ergebnis byte
define i byte

#main
Print "Wert 1 eingeben "
```

```

Input wert1
For i=7 to 0 step -1
If (wert1 AND 1 shl i)=0 then print "0"; else print "1";
next i
print
Print "Wert2 eingeben "
Input wert2
for i=7 to 0 step -1
if (wert2 AND 1 shl i)=0 then print "0"; else print "1";
next i
print
ergebnis=wert1 AND wert2
Print "Wert1 AND Wert2 ergibt " ; ergebnis
for i=7 to 0 step -1
if (ergebnis AND 1 shl i)=0 then print "0"; else print "1";
next i
print
Goto main

```

Da wir gerade einmal bei den Bitmanipulationen sind, so sollten Sie sich das Ganze einmal mit dem Befehl OR ansehen. Auch dieser Befehl ist sehr wichtig bei vielen Manipulationen.

Er ist quasi das Gegenstück zum AND. Dort wo eine 0 steht, wird eine 1 geschrieben. Also 0 AND 128 ergibt 0 und entsprechend 0 OR 128 ergibt 128.

Im Programm muß nur die Zeile Ergebnis=wert1 AND wert2 in Ergebnis=wert1 OR wert2 geändert werden.

Und dann sollten Sie sich auch noch den nächsten Vergleichsoperator XOR ansehen. Er ist das Gegenteil des OR – Befehls 1 OR 1 ergibt 1 – entsprechend 1 XOR ergibt 0. Die neue Programmzeile sieht dann entsprechend Ergebnis = Wert1 XOR Wert2 aus.

So, nun aber zurück zu unserem Problem. Wenn Sie nun Übung haben, so wird alles leicht verständlich sein. Sie haben in obigem Programm schon gesehen, wie man ein Byte in Bits auflöst.

Die Schleife, die wir vorhin entwickelt haben sah so aus:

```

For i= 7 to 0 step -1
if (schreibe AND 1 shl i) = 0 then data = 0
if (schreibe AND 1 shl i) <> 0 then data =1
next i

```

Sie kann eleganter geschrieben werden, wie sie oben im Programm schon zu sehen war. Die Funktion ELSE kann eingesetzt werden.

```

For i= 7 to 0 step -1
if (schreibe AND 1 shl i) = 0 then data = 0 else data =1
next i

```

Für den I2C-Bus liegen jetzt bitweise die richtigen Datawerte vor. Sie werden dann einfach mit einem Pulsebefehl auf der Clockleitung dem I2C-Chip übermittelt. Also: Daten anlegen, wegpulsen, Daten anlegen, wegpulsen usw.

```

For i= 7 to 0 step -1
if (schreibe AND 1 shl i) = 0 then data = 0 else data =1
pulse clock

```

next *i*

Das Adressbyte ist nun weggesendet und wir kommen zu dem oben schon beschriebenen GETACK.

Vielleicht ist die Erklärung doppelt ausgefallen, doch etwas Redundanz schadet nicht. Wenn der I2C-Chip erfahren hat, dass das Adressbyte gesendet wurde, so wird er selbst aktiv und zieht die Dataleitung des Prozessors nach 0. Dazu gibt es die schon beschriebene Funktion GETACK.

```
#GETACK
data = 1
deact data
wait not data
pulse clock
RETURN
```

Wenn die CControl den Polaritätswechsel der Dataleitung von 1 nach 0 entdeckt hat, weiß die Ccontrol, dass der Chip das Adressbyte verstanden hat, es kann endlich weitergehen. Das Schreibbit LSB=1 hat dem Chip mitgeteilt, dass gelesen werden soll: der Chip also aktiv werden muß.

Der DS1621 legt nun nacheinander die einzelnen Bits des in ihm gespeicherten Datenbytes vor. Und wieder im gewohnten Takt mit der Clockleitung.

Data=x: pulse clock, dann data=x: pulse clock, data=x: pulse clock

Was bedeutet es nun für unser Datenbyte? Sagen wir einmal, dass die Temperatur 23 Grad im Chip gespeichert wäre.

Dann liegt dieses Byte im Temperaturchip bitweise so vor. Bei der Kommunikation schiebt der Chip wieder in der bekannten Methode das MSB zuerst, dann die nächsten – bis zum LSB. Wir müssen nun die einzelnen Wertigkeiten notieren, um an das Ergebnis zu kommen.

128 = 0 64 = 0 32 = 0 16 = 1 8 = 0 4 = 1 2 = 1 1 = 1
16+4+2+1 = 23

Also: MSB 0 0 0 1 0 1 1 1 LSB

Aus dieser Bitfolge müssen wir 23 lesen.

Das Unterprogramm GETBYTE macht dies möglich. Sie werden es jetzt nach dem kurzen Exkurs gut verstehen können.

Zunächst einmal wird die Variable I2Caus auf den Wert 0 gesetzt, damit ein definierter Anfangsstatus gegeben ist.

Danach kommt wieder die bekannte Schleife mit der Decrementierung der Variable *i*.

Das erste Bit (MSB) ist 0, data ist entsprechend 0; 1 shl 7 ist 128. Data ist 0, also wird nichts addiert.

Danach wieder 0 usw.

Die Bits, die Data auf 1 legen werden addiert. Zuerst wäre dies bei *i*=4 mit dem Wert 1 shl 4 =16.

Sind wir beim LSB angekommen, so ist I2Caus auf dem Wert 23.

Nach jedem Bitwechsel kommt dann der bekannte Puls auf der Clockleitung. Der Chip weiß nun, dass er das nächste Bit vorlegen kann.

Eigentlich ganz einfach, oder?

```

#GETBYTE
I2Caus = 0
for i = 7 to 0 step - 1
if data < > 0 then I2Caus = I2Caus + 1 shl i
pulse clock
next

```

Der Vorgang des Lesens von einem Byte ist damit fast abgeschlossen. Wir senden einen Stopbefehl. Dieses ist das Gegenteil des Startbefehls.

```

#START
clock = 1 : data = 1 : data = 0 : clock = 0
return

```

```

#STOP
clock = 1 : data = 0 : data = 1
return

```

Damit ist ein Byte gelesen, wie in unserem Falle die 23.

Der Temperaturchip liefert aber zwei Bytes, wenn man die 0.5 Grad Genauigkeit haben möchte. Dieses ist jetzt schnell erklärt. Wir müssen noch ein Unterprogramm GIVEACK entwickeln.

Dieses wird zwischen das erste gelesene Byte und das zweite zu lesende Byte gesetzt. Nach dem I2C-Busprotokoll sieht dies so aus: die CControl signalisiert nach dem ersten gelesenen Byte: ok, ich habe verstanden, das erste Byte ist angekommen, du kannst weitermachen.

```

#GIVEACK
data = 0 : pulse clock : data = 1
return

```

Damit können wir nun zwei Byte lesen (oder in anderen Applikationen, zB. Beim EEPROM 2465 den gesamten Speicher byteweise nacheinander auslesen)

Unser Lesevorgang sieht also so aus:

```

Gosub start
Schreibe=adresse+1 , für lesen
Gosub putbyte
Gosub getack
Gosub getbyte
Merke den gelesenen Wert I2Caus für das erste Byte
Gosub giveack
Gosub getbyte
Merke den gelesenen Wert I2Caus für das zweite Byte
Gosub stop

```

Wenn wir es nach der genauen Spezifikation des Datenblatts gestalten wollen, so müssen wir nach dem zweiten gelesenen Byte noch ein NO ACKNOWLEDGE senden. Dieses als Unterprogramm sieht dann so aus:

```

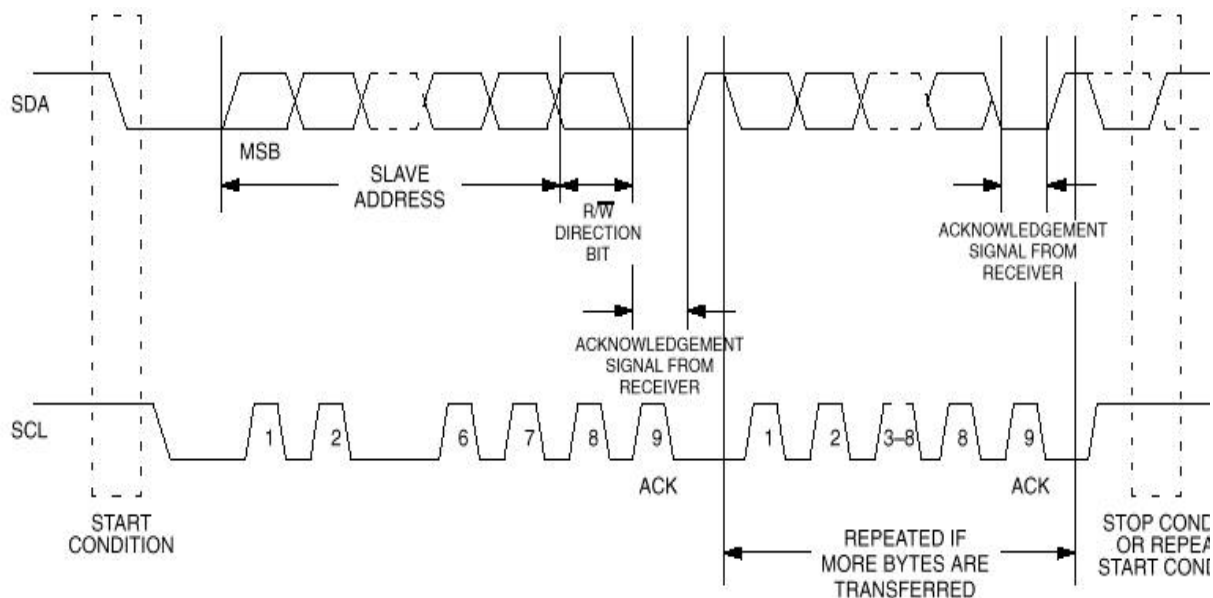
#GIVENOACK
data = 1 : pulse clock : data = 0
return

```

Es läuft bei mir auch ohne dieses GIVENOACK. Aber machen wir es genau:

```
Gosub start
Schreibe=adresse+1
Gosub putbyte
Gosub getack
Gosub getbyte
Merke den gelesenen Wert I2Caus für das erste Byte
Gosub giveack
Gosub getbyte
Merke den gelesenen Wert I2Caus für das zweite Byte
Gosub givenoack
Gosub stop
```

Damit haben wir jetzt ein komplettes I2C-Busprotokoll gebastelt. Damit lassen sich später alle Chips aus dieser Familie ansteuern.



Zeitdiagramm einer Datenübertragung über den I2C-Bus

Wir kommen jetzt zu den Spezialitäten des DS1621.

Die wichtigste Funktion : das Lesen von Temperaturen haben wir ja oben schon besprochen. Bleibt noch zu erklären, wie man mit dem zweiten Byte umgeht. Das erste Byte enthält die Temperatur in Grad Celsius zwischen - 55 Grad Celsius und +125 Grad Celsius. Dabei muß man wissen, dass zwischen 0 und 70 Grad die größte Genauigkeit liegt.

Bei den Werten außerhalb muß man mit größeren Ungenauigkeiten rechnen (siehe Datenblatt).

Für viele Anwendungen reicht diese Auflösung aus. Doch es geht auch genauer; zumindest in 0,5 Grad – Schritten.

(später werden wir sehen, dass wir auch feiner auflösen können)

Vorweg: eine gelesene Temperatur von 23.0 Grad kann bedeuten:

22.76 Grad bis 23.25 Grad. Eine Temperatur 23.5 Grad bedeutet entsprechend 23.26 bis 23.75 Grad.

Dieser Dezimalwert wird mit dem zweiten gelesenen Byte geliefert. Daher spricht man hier auch von einem 9 Bit –Wert.

Wir wissen jetzt, dass dies eben nur mit zwei Bytes zu schaffen ist. Beim Lesevorgang kommt also zuerst das Temperaturbyte, das zweite Byte liefert im MSB entweder eine 1 oder eine 0.

Der Rest ist immer 0. Ist das Bit gesetzt (Wert 128) können wir daraus einen 0.5 Grad-Wert lesen. Der Wert wird auf das erste Byte addiert. Entsprechend wird aus erstem Byte =23, zweites Byte =128 ein Temperaturwert von 23.5 Grad Celsius.

In Basic können wir dies auf verschiedene Art lösen.

```
Erstes gelesenes Byte I2Caus =23
Temp=I2Caus
```

```
Zweites gelesenes Byte I2Caus =128
If I2Caus = 128 then dezi=5 else dezi=0
Print „Aktuelle Temperatur:“; temp; “.“; dezi; “ °C“
```

Oder, wenn man Variablen sparen will

```
Erstes gelesenes Byte I2Caus =23
Print „Aktuelle Temperatur“; I2Caus
Zweites gelesenes Byte I2Caus =128
If I2Caus = 128 then print „.5“ else print „.0“
```

Temperatur	digitaler Ausgang
+125°C	01111101 00000000
+25°C	0001100100000000
+1/2°C	0000000C10000000
+0°C	0000000C00000000
-1/2°C	11111111 10000000
-25°C	11100111 00000000
-55°C	1100100100000000

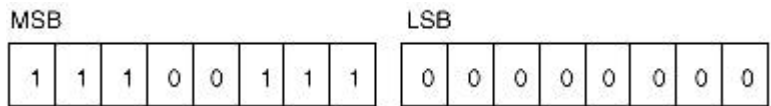
Spannender wird es dann bei den negativen Temperaturen. Hier muß man etwas rechnen. Zunächst einmal ein paar Beispiele:

```
Erstes Byte    Zweites Byte
0 0 0 1 0 1 1 1  1 0 0 0 0 0 0 0    = 23 + 0.5 = + 23.5 Grad Celsius
0 0 0 1 0 1 1 1  0 0 0 0 0 0 0 0    = 23 + 0.0 = + 23.0 Grad Celsius

0 0 0 0 0 0 0 0  1 0 0 0 0 0 0 0    = 0 + 0.5 = + 0.5 Grad Celsius
0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0    = 0 + 0.0 = + 0.0 Grad Celsius
```

1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 = 255 + 0.0 = - 1.0 Grad Celsius
 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 = 255 + 0.5 = - 0.5 Grad Celsius

 1 1 1 0 1 0 0 0 1 0 0 0 0 0 0 0 = 232 + 0.5 = - 23.5 Grad Celsius
 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 = 232 + 0.0 = -24.0 Grad Celsius



T = -25°C

Bei Dallas wird das 9.Bit als LSB bezeichnet, da alle folgenden Bits immer 0 sind.

Zunächst einmal können wir erkennen, dass bei negativen Temperaturen das MSB des ersten gelesenen Bytes 1 ist und nur dann. Erinnern wir uns an die Höchsttemperatur von +125 Grad. Sie ist kleiner als 128, so dass im positiven Bereich kein positives MSB geliefert wird.

Erkennen wir eine 1 an erster Stelle, so können wir reagieren. Wir wissen: die Temperatur ist negativ.

Wie wir an den Beispielen sehen können, müssen wir jetzt die Temperatur anders ermitteln, das Ergebnis erhalten wir aus einer Subtraktion von 256 – neuer Temperaturwert, da der erste Minuswert eine -1 ergibt. Der Minuswert muß dann addiert werden, um auf -1/2 Grad zu kommen.

Damit ergibt sich bei einer gelesenen Temperatur von 255 : $256 - 255 = - 1$; bei gelesenen 254 entsprechend $256 - 254 = - 2$. Oder gelesen 232 ergibt $256 - 232 = 24$.

In Basic können wir dies wieder auf verschiedene Weise realisieren.

```
wert = 232
temp = 256 -wert
```

Oder wieder eleganter, wie oben gelernt:

```
temp = (255 XOR temp) +1
```

Zeigen wir noch, wie wir das Minus lesen.

In meinem Programm merke ich mir die negative Temperatur mit einem Bit, das am Anfang deklariert wurde. Diese Methode ist natürlich sehr platzsparend.

Für alle Werte, die nur 1 oder 0 sein können, sollte man sich dies bei der Programmierung angewöhnen. Es sind nämlich nur 24 Bytes als Variable verfügbar. Arbeitet man bitweise, so kann man ein ganz spezielles Bit im Speicher ansprechen.

24 Bytes machen bekanntlich $24 * 8 = 192$ Bits. Das deklarierte Bit kann dann entsprechend on oder off sein.

Um nicht mit den anderen Variablen zusammenzustoßen, sollte man vom letzten Bit 192 nach unten deklarieren.

```
Define minus Bit[192]
```

Wert =232

If wert AND 128 then minus = on else minus =off
Bit 7 ist gesetzt bei 232, also minus=on

Wert 23

If wert AND 128 then minus = on else minus = off
Oder: if wert = 128 then minus = on else minus = off
Bit 7 ist nicht gesetzt bei 23, also minus = off

Ganz am Ziel sind wir noch nicht.

Wir können zwar jetzt Temperaturen auslesen. Doch der Chip wird sie noch nicht liefern. Der DS1621 ist ein komplexer Chip mit vielen Funktionen.

Um die gewünschte Reaktion zu erhalten oder anzuregen, muß dies vorher mitgeteilt werden. Dazu gibt es verschiedene Controlparameter. Zusätzlich gibt es ein Statusregister oder auch als Configurationsregister zu betrachten.

Zusätzlich muß man ein bisschen über die Methodik des Chips wissen., wie er aus dem Analogwert ‚Temperatur‘ einen Digitalwert herstellt. Der Temperatursensor, der sich im Chip integriert befindet, liefert einen analogen Spannungswert.

Gemessen wird mit einem A/D-Wandler, der aus dem analogen Signal einen entsprechenden Digitalwert herstellt. Dabei wird die anliegende Spannung mit einem Taktsignal so lange gelesen, bis ein Nulldurchgang festgestellt wird.

Die Anzahl der Impulse ist ein Maß für die Temperatur, die vom Minimalwert –55 Grad Celsius entfernt ist. Zusätzlich wird ein zweiter Zähler angeworfen, der die Kennlinienkompensation bewerkstelligt. Die Werte werden in einem internen EEPROM – Speicher abgelegt, so dass sie später vom I2C-Bus abgeholt werden können.

Die Wandlung besteht nun aus einem Anstoß der Zähler und dem erfolgten Messen der Temperatur. Diese Prozedur dauert einige Millisekunden (typisch 0.4 Sek).

Wenn man normale Temperaturen mit dem 0.5 Dezimalwert messen will, so braucht man sich darum gar nicht zu kümmern. Man stellt den Modus ‚kontinuierliches Wandeln‘ ein. Der AD-Wandler wird angestoßen–das Ende der Wandlung erkannt, der Wert wird in den Speicher geschrieben – der A/D-Wandler wird wieder angestoßen usw. usw. Beim Lesen der beiden Datenbytes erhält man so den gerade abgeschlossenen Zyklus zurück.

Access Config [ACh]

If $\overline{R/\overline{W}}$ is 0, this command writes to the configuration register. After issuing this command, the next data byte is the value to be written into the configuration register. If $\overline{R/\overline{W}}$ is 1, the next data byte read is the value stored in the configuration register.

Also stellen wir jetzt einmal diese Methode ein.

Das Bit 0 vom Statusbyte muß auf 0 gestellt werden. Dies bedeutet, dass die Wandlung kontinuierlich ausgeführt wird. Steht hier eine 1, so bedeutet dies, dass nur eine Wandlung

auf Anforderung durchgeführt wird. Anschließend bleibt der Wandler stehen und man mißt immer die gleiche Temperatur. Die Einleitung des Befehls ‚Statusregister schreiben‘ wird mit dem Byte &HAC oder dezimal 173 eingeleitet.

Also:

Senden des Adressbytes im Schreibmodus

```
Gosub start
Schreibe=adresse
Gosub putbyte
Gosub getack
Schreibe=173
Gosub putbyte
Gosub getack
Schreibe=0
Gosub putbyte
Gosub getack
Gosub stop
```

Wir haben jetzt einfach eine Null eingesetzt, um die kontinuierliche Wandlung einzuleiten. Später sollte man dieses nicht so einfach realisieren, wenn man verschiedene Funktionen des Chips nutzen will. Dann muß man das Statusregister zuerst auslesen und das Bit0 gezielt setzen oder löschen. Dazu später aber noch mehr.

Jetzt ist die kontinuierliche Wandlung eingestellt. Aber noch werden keine Werte geliefert. Wir müssen nämlich mit einer anderen Sequenz die Wandlung erst anstoßen.

Start Convert T [EEh]

This command begins a temperature conversion. No further data is required. In one-shot mode, the temperature conversion will be performed and then the DS1621 will remain idle. In continuous mode, this command will initiate continuous conversions.

Das Byte hierzu heißt Start Convert &HEE ; dezimal 238

```
Gosub start
Schreibe=adresse
Gosub putbyte
Gosub getack
Schreibe=238
Gosub putbyte
Gosub getack
Gosub stop
```

Jetzt ist die Wandlung angestoßen und der Chip mißt die Temperaturen. Doch um die Temperatur auch lesen zu können, müssen wir dem Chip unseren Wunsch mitteilen.

Das Kommando Read Temperature ist &HAA oder dez 170.

Read Temperature [AAh]

This command reads the last temperature conversion result. The DS1621 will send two bytes, in the format described earlier, which are the contents of this register.

```
Gosub start
Schreibe=adresse
Gosub putbyte
Gosub getack
Schreibe=170
Gosub putbyte
Gosub getack
Gosub stop
```

So, jetzt sind wir soweit, dass wir ständig die aktuelle Temperatur auslesen können, wie oben beschrieben. Hier noch einmal kurz die Zusammenfassung.

```
Gosub start
Schreibe=adresse+1
Gosub putbyte
Gosub getack
Gosub getbyte
Temp merken
Gosub giveack
Gosub getbyte
Dezi merken
Gosub givenoack
Gosub stop
```

Was kann man noch mit dem Chip anstellen. Wie schon erwähnt, können wir dem Chip sagen, dass er nur eine Wandlung ausführen soll und danach in den stromsparenden Betrieb gehen soll. Die abgespeicherten Werte bleiben erreichbar.

CONFIGURATION/STATUS REGISTER

DONE	THF	TLF	NVB	1	0	POL	1SHOT
------	-----	-----	-----	---	---	-----	-------

Die sogenannte oneshot – Methode lösen wir mit dem Bit0 =1 im Statusbyte aus. Zunächst aber stoppen wir die Wandlung mit Stop Convert &H22; dez. 34.

Dann schreiben wir das Statusbyte neu mit einer 1 als Bit0.

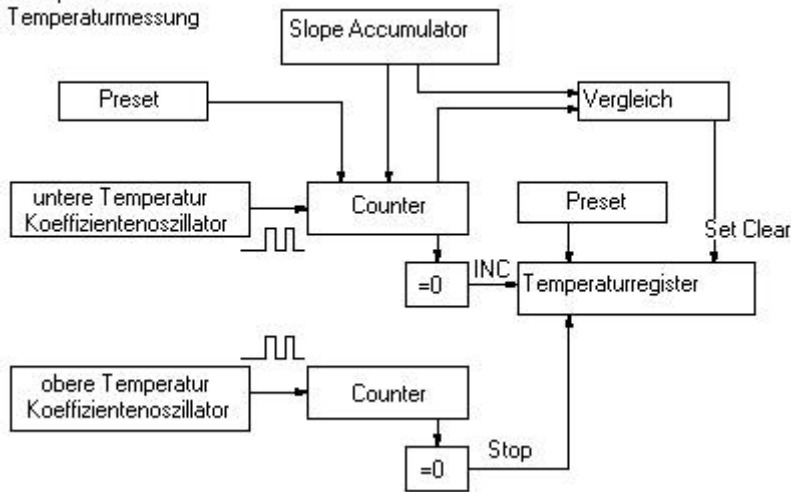
Danach starten wir die Wandlung mit &HEE.

Wie oben schon beschrieben, braucht die Unit einige Millisekunden, bis der Wert digital vorliegt. Der Chip meldet dies im Statusbyte durch das Setzen des MSB auf logisch 1.

Also muß man so lange warten, bis hier eine 1 gesetzt wurde. Immer wieder Statusbyte lesen, nachsehen, ob Bit 7 auf 1 ist, wenn nicht: wieder lesen, wenn ja: geht es weiter. Man kann jetzt nach obiger Prozedur den Inhalt lesen.

Im Programm finden Sie dies im Unterprogramm #oneshot_Temperatur_messen.

Prinzip der
Temperaturmessung



Es gibt eine Methode, bei der man dem Chip genauere Temperaturwerte entlocken kann. Wie weit man diesen Werten dann in den Stellen nach dem Komma trauen kann, vermag ich nicht zu beurteilen.

Jedenfalls werden wir diese Methode jetzt einmal durchgehen. Ich sprach oben schon von den beiden Zählern, die im Chip bei jeder Wandlung angeworfen werden. Diese Zähler können ausgelesen werden, wenn man die oneshot-Methode einleitet, damit sie anschließend bei der nächsten Wandlung nicht direkt wieder überschrieben werden.

Der Temperaturzähler und der Kompensationszähler können einzeln ausgelesen werden, um dann in einer Formel zur genaueren Temperaturbestimmung genutzt zu werden.

Zunächst einmal leiten wir die oneshot-Methode ein.

Wir lesen die Temperatur aus und verwerfen den evtl. gelieferten 0,5 Wert.

Danach lesen wir die Zähler aus. Sie heißen Slopecounter und Counter.

Read Counter [A8h]

This command reads the value of the counter byte. This command is valid only if $\overline{R/W}$ is 1.

Read Slope [A9h]

This command reads the value of the slope counter byte from the DS1621. This command is valid only if $\overline{R/W}$ is 1.

Die Kommandos lauten für Read Counter &HA8, dez. 168 und für den Read Slope &HA9 oder 169 dezimal.

Die Werte werden dann in die Formel eingesetzt:

$$\text{Temperatur} = \text{temp} - 0.25 + (\text{slopecounter} - \text{counter}) / \text{slopecounter}$$

Hier kommen nun wirklich Zwischenwerte heraus, die beim Ablesen über eine längere Zeit auch einen guten und sinnvollen Eindruck machen. Zumindest mathematisch stimmt alles. Ob es mit der Auflösungsgenauigkeit so korreliert, kann ich nicht entscheiden.

Nun kommt aber noch die große Aufgabe der Wandlung des Ergebnisses in Nachkommastellen. Da die CControl keine Floating – Zahlen berechnen kann, muß man sich das selbst basteln.

Man muß dabei die Werte mit 100 multiplizieren, um auf die 0,25 reagieren zu können.

Ebenso wird der entstehende Bruch mit 100 multipliziert und später wieder geteilt. Den Temperaturwert erhält man durch eine folgende Teilung durch 100.

$$\text{ganzwert} = ((\text{temp} * 100 - 25) + ((\text{slope} * 100 - \text{counter} * 100) / \text{slope} * 100) / 100) / 100$$

Der ganze Ausdruck wird dann mit der Funktion MOD ausgelesen, um den Nachstellenwert zu ermitteln. Das Ergebnis ist dann der Rest des Ausdrucks.

$$\text{deziwert} = ((\text{temp} * 100 - 25) + ((\text{slope} * 100 - \text{counter} * 100) / \text{slope} * 100) / 100) \bmod 100$$

Im Programm kann man sich dies unter `#genauere_Temperaturen_messen` ansehen.

Thermostat:

Der Chip besitzt eine raffinierte Thermostاتفunktion. Der Ausgang tout am DS1621 (Pin 3) ist ein digitaler Ausgang, der als Thermostat benutzt werden kann. Er liefert entweder eine 1 oder eine 0.

Dieser Ausgang kann in seiner Polarität bestimmt werden, je nach Auswerteelektronik, die man dahinter setzen will.

Gehen wir einfach einmal an die Einstellung der Polarität. Sie können den Wechsel der Polarität am Pin 3 messen. Ich habe mir auf die Testschaltung eine Leuchtdiode über einen Transistor geschaltet, um den Wechsel beobachten zu können.

Das Einstellen der Polarität wird wieder beim Schreiben des Statusbytes vorgenommen. Und zwar dient hierzu das Bit1. Eine 1 dort gesetzt bedeutet positive Polarität; eine 0 entsprechend negative Polarität.

Jetzt wollen wir dies aber richtig machen und das Bit setzen, ohne die anderen Statusbits zu beeinflussen.

Also lesen wir zuerst einmal das Statusbit über das Kommando `&HAC, 173 dez.aus`

```
Gosub start
Schreibe=adresse
Gosub putbyte
Gosub getack
Schreibe=173
Gosub putbyte
Gosub getack
Gosub start
Schreibe=adresse+1
Gosub putbyte
Gosub getack
Gosub getbyte
```

```
Gosub stop
```

Der Inhalt des Statusbytes liegt jetzt in der Variablen I2Caus vor.

Wir müssen den Inhalt an der Stelle von Bit1 (Polaritätsbit) wissen, um die positive Polarität einzuschalten. Wir fragen, ob das Bit1 = 0 ist.

```
If (i2caus and 1 shl 1) = 0 then polaritaet = i2caus + 2
```

Machen wir hier gleich das Gegenteil mit. Wir wollen die Polarität auf Null setzen. Wir fragen, ob das Bit1 = 1 ist.

```
If (i2caus and 1 shl 1) <>0 then polaritaet = i2caus - 2
```

Mit dem so gewonnenen Wert können wir das Statusbyte neu schreiben.

```
Gosub start  
Schreibe=adresse  
Gosub putbyte  
Gosub getack  
Schreibe=173  
Gosub putbyte  
Gosub getack  
Schreibe=polaritaet  
Gosub putbyte  
Gosub getack  
Gosub stop
```

Damit haben wir die Polarität auf positiv geschaltet. Gibt man den zweiten Polaritätswert ein, mit Bit1=0, so stellt man auf negative Polarität.

Ein Thermostat zeichnet sich durch eine Hysterese aus. Dieses ist der Bereich, in dem er arbeitet.

Der obere Hysteresezeitpunkt ist der Einschaltzeitpunkt, der untere ist der Ausschaltzeitpunkt. Diese Hysterese ist einstellbar.

Ein Beispiel: Einschalten bei +25.5 Grad Celsius, z.B. einen Ventilator, Ausschalten wieder bei 23.0 Grad Celsius.

Laut Datenblatt sollte man diese Funktionen nur im Bereich zwischen 0 und 70 Grad Celsius nutzen. Das macht es uns auch leichter mit der Umrechnung der Minustemperaturen.

Access TH [A1h]

If R/\overline{W} is 0, this command writes to the TH (HIGH TEMPERATURE) register. After issuing this command, the next two bytes written to the DS1621, in the same format as described for reading temperature, will set the high temperature threshold for operation of the T_{OUT} output. If R/\overline{W} is 1, the value stored in this register is read back.

Das Kommando für die Einstellung des oberen Hysteresezeitwertes lautet hexA1, 161 dez.

```
Gosub start
```

```

Schreibe=adresse
Gosub putbyte
Gosub getack
Schreibe=161
Gosub putbyte
Gosub getack
Schreibe=25
Gosub putbyte
Gosub getack
Schreibe=128
Gosub putbyte
Gosub getack
Gosub stop

```

Die 25,5 Grad sind eingeschrieben. Der Ventilator würde anlaufen, wenn die nachgeschaltete Elektronik mit der eingestellten Polarität harmoniert. Aber die kann man ja einstellen, wie oben gesehen.

Access TL [A2h]

If R/\overline{W} is 0, this command writes to the TL (LOW TEMPERATURE) register. After issuing this command, the next two bytes written to the DS1621, in the same format as described for reading temperature, will set the high

Auf die gleiche Weise wird jetzt die untere Temperatur geschrieben. Das Kommando dazu lautet entsprechend hexA2, 162 dez.

Bei Erreichen der unteren Temperatur schaltet sich der Ventilator wieder aus.

Nun kann man zur Kontrolle diese Temperaturwerte natürlich auch wieder lesen. Die obere Temperatur z.B.

```

Gosub start
Schreibe=adresse
Gosub putbyte
Gosub getack
Schreibe=161
Gosub putbyte
Gosub getack
Gosub start
Schreibe=adresse+1
Gosub putbyte
Gosub getack
Gosub getbyte
Temp=i2caus
Gosub giveack
Gosub getbyte
Dezi=i2caus
Gosub givenoack
Gosub stop
Print temp,dezi

```

Die obere eingestellte Thermostattemperatur wurde gelesen. Natürlich muß man sich die Anzeige wieder so hinbasteln, wie man dieses wünscht. Dafür haben wir aber schon Beispiele gesehen.

Ebenso kann die untere Temperatur mit hexA2 gelesen werden.

Noch eine Spezialität in Verbindung mit der Thermostاتفunktion soll beschrieben werden. Es gibt noch einen Minimum- und Maximummerker. Man kann über das schon oft beschriebene Statusbyte auslesen, ob die obere Temperatur oder die untere Temperatur jemals erreicht worden ist.

Wurde z.B. in unserem Beispiel die Temperatur 25.5 erreicht, oder ist die Temperatur darüber gegangen, so wird im Statusbyte das Bit6 auf 1 gesetzt. Entsprechend bei Unterschreitung des unteren Punktes das Bit5 auf 1 gesetzt.

CONFIGURATION/STATUS REGISTER

DONE	THF	TLF	NVB	1	0	POL	1SHOT
------	-----	-----	-----	---	---	-----	-------

Diese beiden Bits THF und TLF bleiben gesetzt, auch wenn die Temperatur wieder im Normalbereich ist. Sie können nur mit einem Befehl über den Schreibvorgang des Statusbytes gelöscht werden.

Es geht natürlich einzeln über Bit6 oder Bit5, man kann sie auch gemeinsam zurücksetzen.

Achtung : ist die Temperatur noch außerhalb der Hysterese, so wird das Bit sofort wieder gesetzt.

Mit all diesen Möglichkeiten lassen sich raffinierte Schaltungen aufbauen. Zum Beispiel beim Einsatz in einer Heizungssteuerung lassen sich exakte Hystereseverläufe realisieren. Mißt man mit diesem Chip die Kesseltemperatur, so kann man exakt die Wassertemperatur steuern. Kessel auf 70 Grad – Brenner aus. Kessel auf 60 Grad: Brenner ein.

Oder aber die Konstruktion eines Minimum/Maximum-Thermometers realisieren. Will man die höchste und tiefste Temperatur eines Jahres ermitteln (und man eine Ccontrol dafür opfert), so könnte man bei Inbetriebnahme die aktuelle Temperatur messen und den Maximumwert temp + 0.5 einstellen.

Den Minimumwert entsprechend temp -0.5 Grad Celsius.

Man schaut nach, ob das obere oder untere Flagbit gesetzt ist.

Wenn ja, so schreibt man die neue Temperatur 0.5 höher oder tiefer in den oberen oder unteren Bereich. Wird eine erneute Überschreitung gemessen, so macht man die gleiche Prozedur.

Am Jahresende –oder auch zwischendurch- liest man die hohe und die tiefe Thermostattemperatur aus und kennt die Extremwerte.

Wenn man noch einen Port der Ccontrol opfern kann, so läßt sich der Thermostat-Ausgang tout (Pin3) natürlich auch sehr gut überprüfen und über den I2C-Bus Relais ansteuern.

Beim Lallus wird der I2Cirq auf alle Platinen mitgeführt. Auch ihn (port 6) kann man dazu nutzen.

Das untenstehende Programm für die CControl zeigt alle Möglichkeiten, die man dem Temperaturchip DS1621 entlocken kann. Sie sollten den Programmbereich markieren und ihn in ein neues Editorfenster der CControl kopieren. Das Programm hat etwa 3 Kilobyte Sourcecode.

Programmname bei mir DS1621.BAS

* Programm für die CControl

```
define i2cirq    port [6] ' entdeckt Interrupt von I2C
define clock    port [7] ' clockt die I2C - Leitung (SCL)
define data     port [8] ' setzt die Dataleitung (SDA) für I2C

define temp     word
define slope    word
define counter  word
define merker   word
define adresse  byte
define i        byte
define i2caus   byte
define schreibe byte
define zeichen  byte
define tendenz  bit[190]
define flag     bit[191]
define minus    bit[192]

adresse=&H92

data = on
gosub anzeige
#main
if rxd then get zeichen
if zeichen=27 then gosub anzeige
if zeichen>91 then zeichen=zeichen-32
if zeichen=65 then gosub Temperatur_ein_Byte_lesen
if zeichen=66 then gosub Temperatur_zwei_Byte_lesen
if zeichen=67 then gosub genauere_Temperatur_messen
if zeichen=68 then gosub oneshot_Temperatur_messen
if zeichen=69 then gosub Thermostat_high_lesen
if zeichen=70 then gosub Thermostat_low_lesen
if zeichen=71 then gosub Thermostat_high_schreiben
if zeichen=72 then gosub Thermostat_low_schreiben
if zeichen=73 then gosub High_und_Lowflag_loeschen
if zeichen=74 then gosub Ausgangs_polaritaet_positiv
if zeichen=75 then gosub Ausgangs_polaritaet_negativ
if zeichen=76 then gosub Configurationsbyte_lesen
if zeichen=77 then gosub Adresse_einstellen
if zeichen=78 then goto Staendig_genau_messen
if zeichen=79 then goto Staendig_Zweibyte_messen
if zeichen=80 then gosub Tendenz_anzeigen
zeichen = 0
goto main

#Tendenz_anzeigen
tendenz=not tendenz
return

#Staedig_Zweibyte_messen
print "Mit <ESC> die Messung abbrechen"
#Staedig_zwei
if rxd then get zeichen
if zeichen=27 then goto main
```

```

gosub Temperatur_zwei_Byte_lesen
goto staendig_zwei

#Staendig_genau_messen
print "Mit <ESC> die Messung abbrechen"
#Staendig_genau
if rxd then get zeichen
if zeichen=27 then goto main
gosub genauere_Temperatur_messen
goto staendig_genau

#High_und_Lowflag_loeschen
gosub Configbyte_lesen
gosub Byte_binaer_darstellen
if i2caus and 64 then print "Highflag war gesetzt" else print
"Highflag war nicht gesetzt"
if i2caus and 64 then i2caus=i2caus-64
if i2caus and 32 then print "Lowflag war gesetzt" else print
"Lowflag war nicht gesetzt"
if i2caus and 32 then i2caus=i2caus-32
gosub schreiben
schreibe=&HAC:gosub putbyte:gosub getack
schreibe=i2caus:gosub putbyte : gosub getack : gosub stop
return

#Configurationsbyte_lesen
gosub Configbyte_lesen
gosub Byte_binaer_darstellen
return

#Byte_binaer_darstellen
for i=7 to 0 step -1
if i2caus and 1 shl i then print "1 "; else print "0 ";
next i
print " ";i2caus
return

#Ausgangs_polaritaet_positiv
gosub Configbyte_lesen
gosub schreiben
schreibe=&HAC:gosub putbyte:gosub getack
if (i2caus and 2)=0 then schreibe = i2caus+2
gosub putbyte:gosub getack
gosub stop
print "Polarität positiv"
return

#Ausgangs_polaritaet_negativ
gosub Configbyte_lesen
gosub schreiben
schreibe=&HAC:gosub putbyte:gosub getack
if i2caus and 2 then schreibe = i2caus - 2
gosub putbyte:gosub getack
gosub stop
print "Polarität negativ"
return

```

```

#Configbyte_lesen
gosub schreiben
schreibe=&HAC : gosub putbyte:gosub getack
gosub lesen
gosub getbyte:gosub givenoack:gosub stop
return

#Thermostat_low_schreiben
print "Eingabe der Thermostat-low Temperatur"
print "Format : 275 für 27.5 oder 270 für 27.0"
input slope
print slope; " wird geschrieben"
temp=slope/10
counter=slope mod 10
gosub schreiben
schreibe=&HA2: gosub putbyte:gosub getack
schreibe=temp : gosub putbyte :gosub getack
if counter =0 then schreibe=0 else schreibe=128
gosub putbyte:gosub getack
gosub stop
gosub anzeige
return

#Thermostat_high_schreiben
print "Eingabe der Thermostat-high Temperatur"
print "Format : 275 für 27.5 oder 270 für 27.0"
input slope
print slope; " wird geschrieben"
temp=slope/10
counter=slope mod 10
gosub schreiben
schreibe=&HA1:gosub putbyte:gosub getack
schreibe=temp : gosub putbyte
gosub getack
if counter<>0 then schreibe=128 else schreibe=0
gosub putbyte : gosub getack
gosub stop
gosub anzeige
return

#Thermostat_low_lesen
gosub schreiben
schreibe=&HA2:gosub putbyte:gosub getack
gosub lesen
gosub getbyte
print "Thermostat low ";i2caus;". ";
gosub giveack
gosub getbyte
print (i2caus = 128)*-5 ; " °C"
gosub stop
return

#Thermostat_high_lesen
gosub schreiben
schreibe=&HA1:gosub putbyte:gosub getack
gosub lesen
gosub getbyte

```

```

print "Thermostat high ";i2caus;".";
gosub giveack
gosub getbyte
print (i2caus = 128)*-5 ; " °C"
gosub stop
return

#Adresse_einstellen
print "Sytemadresse einstellen"
print "Grundadresse = "; &H90
input adresse
print "neue Adresse = " ;adresse
gosub anzeige
return

#genauere_Temperatur_messen
gosub oneshot_Temperatur_messen
gosub schreiben
schreibe=&HA8:gosub putbyte:gosub getack
gosub lesen
gosub getbyte
counter=i2caus
print "Remaincounter ";counter
gosub stop
gosub schreiben
schreibe=&HA9:gosub putbyte:gosub getack
gosub lesen
gosub getbyte
slope=i2caus
print "Slopecounter ";slope
gosub stop
print "genaue Temperatur ";
if minus=on then print "-";
if minus then temp = (255 xor temp)+1
print ((temp*100-25)+ ((slope-counter)*100/slope*100)/100)/100;".";
i2caus=((temp*100-25)+ ((slope-counter)*100/slope*100)/100) mod 100
if i2caus<10 then print "0";
print i2caus; " °C"
return

#oneshot_Temperatur_messen
gosub schreiben
schreibe=&HAC : gosub putbyte:gosub getack
if (i2caus and 1) = 0 then schreibe=i2caus + 1 else schreibe=i2caus
gosub putbyte:gosub getack
gosub schreiben
schreibe=&H22: gosub putbyte : gosub getack
gosub schreiben
schreibe=&HEE : gosub putbyte : gosub getack
print "Wandlung gestartet"
gosub schreiben
schreibe=&HAC : gosub putbyte:gosub getack
#nochmals
gosub lesen
gosub getbyte:gosub stop
print i2caus
if (i2caus and 128) = 0 then goto nochmals

```

```

print "Wandlung beendet"
gosub schreiben
schreibe=&HAA : gosub putbyte:gosub getack
gosub lesen
gosub getbyte : gosub giveack
temp=i2caus
gosub getbyte : gosub stop
slope=i2caus
gosub Temperatur_untersuchen
return

#Temperatur_ein_Byte_lesen
flag=on
goto Temperatur_zwei_Byte_lesen
return

#Temperatur_zwei_Byte_lesen
gosub Temperatur_lesen_initialisieren
gosub lesen:gosub getbyte
temp=i2caus:gosub giveack
gosub getbyte
slope=i2caus
gosub stop
gosub Temperatur_untersuchen
return

#Temperatur_untersuchen
minus=off
if temp and 128 then minus=on
if slope and 128 then counter=5 else counter=0
if minus then temp=255 xor temp
if minus then print "Temperatur: "; "-" ; else print "Temperatur ";
if flag=on then print temp; " °C"
if flag=off then print temp; "." ; counter; " °C"
if tendenz=on then gosub tendenz
flag=off
return
#tendenz
if merker > temp*10+counter then print "Tendenz: fallend"
if merker = temp*10+counter then print "Tendenz: unverändert"
if merker < temp*10+counter then print "Tendenz: steigend"
merker=temp*10+counter
return
#Temperatur_lesen_initialisieren
gosub schreiben
schreibe=&HAC : gosub putbyte:gosub getack
gosub lesen
gosub getbyte:gosub stop

gosub schreiben
schreibe=&HAC:gosub putbyte:gosub getack
if i2caus and 1 then schreibe=i2caus - 1 else schreibe=i2caus
gosub putbyte:gosub getack
gosub schreiben
schreibe=&HEE:gosub putbyte:gosub getack
gosub schreiben
schreibe=&HAA:gosub putbyte:gosub getack

```

```

return

#schreiben
gosub start:schreibe=adresse:gosub putbyte:gosub getack
return
#lesen
gosub start:schreibe=adresse+1:gosub putbyte:gosub getack
return

' ***** Anzeige *****
#anzeige
print "Temperatur lesen (1 Byte) <A>"
print "Temperatur lesen (2 Byte) <B>"
print "genauere Temperatur lesen <C>"
print "einzelne Wandlung (oneshot) <D>"
print "Thermostat high lesen <E>"
print "Thermostat low lesen <F>"
print "Thermostat high schreiben <G>"
print "Thermostat low schreiben <H>"
print "High- und Lowflag löschen <I>"
print "Ausgangspolarität positiv <J>"
print "Ausgangspolarität negativ <K>"
print "Configurationsbyte lesen <L>"
print "Systemadresse wählen <M>"
print "ständig genaue Temperatur <N>"
print "ständig Zweibytetemperatur <O>"
print "Tendenz ein-/ausschalten <P>"
print "diese Übersicht <ESC>"
print "Systemadresse: ";adresse
return

' ***** I2C *****
#START
clock = 1 : pulse data : clock = 0
return

#STOP
data = 0 : clock = 1 : data = 1
return

#GETBYTE
I2Caus = 0
for i = 7 to 0 step - 1
if data < > 0 then I2Caus = I2Caus + 1 shl i
pulse clock
next
return

#PUTBYTE
for i = 7 to 0 step -1
if (schreibe and 1 shl i) = 0 then data = 0 else data = 1
pulse clock
next
return

#GETACK
data = 1 : deact data : clock = 1 : wait not data : clock = 0

```

```
return

#GIVEACK
data = 0 : pulse clock : data = 1 : deact data
return

#GIVENOACK
data = 1 : pulse clock : data = 0
return
```

Analoge Messungen von Temperaturen

Wir wollen das analoge Messen von Temperaturen nicht ganz außer Acht lassen. Bei der Entwicklung von Lallus wußte ich nämlich auch noch nichts von den digitalen Möglichkeiten, die oben besprochen wurden. Wer evtl. bereits das Temperaturmodul aus der Telemetrieserie besitzt, der soll hier erfahren, wie er eine einfache Temperatursteuerung mit der CControl realisieren kann.

Auf der Lallusplatine und im Applicationboard sind die nötigen Stromversorgungen für das Temperaturmodul bereits realisiert. Es wird einmal eine Versorgungsspannung von +6.5 Volt und einmal eine Referenzspannung von +2,5 Volt benötigt. Das beigefügte Anschlußkabel hat drei Strippen. Schwarz = Masse, rot = +6.5 Volt, weiß= Meßausgang.

Ist alles angeschlossen, so können wir mit dem Programmieren beginnen.

Zunächst einmal wird der analoge Eingang deklariert. Sagen wir, dass für die Temperaturmessung der Analogport 1 benutzt werden soll.

Define temp ad[1]

Weiterhin müssen wir eine Variable deklarieren, die den Temperaturwert aufnehmen soll. Hier müssen wir uns entscheiden, ob wir mit einer Bytedeklaration auskommen, oder ob ein Word angebracht ist.

Was ist der Unterschied?

Ein Byte kann 256 Zustände einnehmen. Von 0 bis 255. Wollen wir also Temperaturen messen, die in diesem Bereich darstellbar sind, so genügt eine Bytevariable. Sie werden sagen, dass dieses ausreicht, da wir ja wohl keine Temperatur über 255 Grad Celsius messen werden.

Dieses stimmt auch, wenn wir bei ganzzahligen Temperaturen bleiben und auf die Nachkommastelle von 0,5 Grad Celsius verzichten wollen. Die Tabelle, die wir später benutzen werden, um die Anpassung der Temperaturen in verschiedenen Bereichen zu realisieren (Eichkurve), ist in 0.5 er Schritten aufgelöst. Sie können sich diese Tabelle in einem Editor ansehen.

Der Name ist kty.tab und liegt im Tabellenverzeichnis der CControl-Software. Vorab: Sie muß später im selben Verzeichnis liegen, in der die Programmsoftware ausgeführt wird, also wahrscheinlich im CCEW32D – Verzeichnis.

Wir haben oben schon gesehen, dass wir mit der CControl nur ganzzahlig arbeiten können und daher keine Nachkommastellen so ohne weiteres verarbeiten können. Hier muß wieder

multipliziert werden und später wird die Temperatur durch eine Teilung ermittelt, der Nachkommawert 0.5 wird durch eine MOD – Operation errechnet.

Wollen wir auch die Minuswerte aus der Tabelle mitnehmen, so muß sowieso ein Word geopfert werden.

Wir wollen natürlich die Möglichkeiten der genaueren Messung ausnutzen und müssen daher ein Word deklarieren. Es ist klar, dass wir bei einer Multiplikation von 27 Grad = 270 schon im Bereich über der Möglichkeit von einem Byte liegen.

An dieser Stelle können wir uns schon einmal das spätere Vorgehen ansehen. Die Gradzahlen aus der Tabelle kty.tab werden so verändert, dass wir den Dezimalpunkt der einzelnen Werte entfernen. Somit wird aus 27.5 eine 275. Später bei der Rückverwandlung des gelesenen Wertes 275 teilen wir wieder durch 10. $275 / 10$ ist für die CControl 27. Ein zweiter Rechenvorgang ermittelt den Teilungsrest $275 \text{ MOD } 10$ ergibt 5. Somit liegen die beiden gewünschten Ergebnisse in Variablen vor.

```
Define temp ad[1]
Define grad word
```

Zunächst basteln wir ein ganz einfaches Programm, um den Analogport auszulesen, ohne auf die Temperaturmessung mit realistischen Zahlen zu achten.

```
#main
pause 10
print temp
goto main
```

Im Terminalprogramm sehen wir jetzt nacheinander Zahlen auftauchen. Wenn wir nun den Finger an den kleinen Temperatursensor bringen (oder ihn vorsichtig mit einem Feuerzeug erwärmen), so werden die Zahlen größer werden.

Der Chip reagiert also positiv. Je höher die Temperatur, desto größer der Zahlenwert. Bei Zimmertemperatur dürfte der Wert so bei 80 sein. Wenn wir die Temperatur kennen, sagen wir 20 Grad Celsius, so könnten wir von der 80 nun 60 abziehen und kämen auf 20 Grad Celsius.

In einem bestimmten Bereich ist dies vielleicht sogar noch tragbar, doch schon bald wäre unser Thermometer fürchterlich ungenau. Wir können uns dies verdeutlichen, wenn wir in die Tabelle schauen. Der letzte Wert ist eine 100, sie ist der 255. Eintrag. Mit der obigen Methode müßten wir dann $255 - 60 = 195$ Grad Celsius rechnen. Kleiner Unterschied!

Wir arbeiten jetzt mit der Tabelle, die zuvor jedoch in Ganzzahlwerte umgewandelt werden muß.

Nehmen Sie WORD oder WORDPAD und öffnen im Verzeichnis TABLES die Datei KTY.TAB.

Öffnen Sie unter BEARBEITEN den Befehl ERSETZEN.

Geben Sie den Dezimalpunkt ein.

In der nächsten Zeile geben Sie nichts ein – Und ALLES ERSETZEN eingeben.

Sie speichern dann die neue Datei z.B. als KTY1.TAB in das Verzeichnis, in dem sich Ihr CControlprogramm befindet ab.

Ich weiß, dass es am Anfang schwer ist, sich in neue Techniken einzuarbeiten; auch wenn sie kärglich in der Anleitung beschrieben sind. Deshalb hier eine kurze Hinführung zur Arbeit mit Tabellen. Dazu zunächst wieder ein kleines Programm.

```

Define wert byte

looktab zahlen,2,wert
print wert
end

Table zahlen 6 5 4 3 2 1 0
Tabend

```

Unsere Tabelle befindet sich außerhalb des Programmes, nach dem END. Sie wird eingeleitet mit dem Schlüsselwort table oder TABLE. Danach kommt der Name der Tabelle, auf den man sich während des Programms bezieht.

Hier ist der Name ‚zahlen‘. Danach stehen die Tabelleneinträge, die jeweils mit einem Leerzeichen getrennt sind. Ganz zum Schluß wird das Schlüsselwort tabend oder TABEND eingegeben, was natürlich das Ende der Tabelle signalisiert.

Die Tabelle wird nun mit dem Schlüsselwort LOOKTAB gelesen. Der nächste Parameter ist die Tabelle, die gelesen werden soll. In unserem Falle die Tabele zahlen. Danach steht der Indexeintrag, beginnend mit 0. Also looktab zahlen,0 liest den ersten Eintrag, eine 6. So käme allerdings eine Fehlermeldung zurück.

Es fehlt noch ein Parameter: die Variable, in die das Ergebnis geschrieben wird. Richtig ist also looktab zahlen,0,wert. Da wir in dem obigen Beispiel den Index 2 auslesen, müßte im Terminalprogramm eine 4 erscheinen. Ändern Sie einfach einmal die Indexwerte und schauen sich die Anzeige an.

Es werden wahrscheinlich sogar mehrere 4 erscheinen, da die Starttaste prellt. Geben Sie vorher noch eine pause 10 ein, dann dürfte dies behoben sein. Um es zu komplettieren: Sie können natürlich auch mit mehreren Tabellen gleichzeitig im Programm arbeiten.

```

Define wert byte
Pause 10
looktab zahlen,2,wert
print wert
looktab negativ,1,wert
print wert
end

Table zahlen 6 5 4 3 2 1 0
Tabend
Table negativ -276 -200 -450
Tabend

```

Wir lesen im Terminalprogramm eine 4 und eine 56. Die 4 ist in Ordnung; doch wo kommt die 56 her? Wir haben an der ausgelesenen Tabellenstelle eine -200 stehen. Da wir Wert nur als Byte definiert haben, können wir mit den negativen Werten nicht arbeiten. Intern wird eine Subtraktion $256-200$ vorgenommen, die zu 56 führt. Wir definieren die Variable deshalb neu.

Define wert word.

Jetzt werden für die Variable zwei Bytes reserviert. Damit lassen sich $256*256 = 65536$ Werte darstellen.

Probieren wir jetzt das geänderte Programm:

```

Define wert word
Pause 10
looktab zahlen,2,wert
print wert
looktab negativ,1,wert
print wert
end

Table zahlen 6 5 4 3 2 1 0
Tabend
Table negativ -276 -200 -450
Tabend

```

Die Temperaturtabell KTY1.TAB können wir automatisch mit in den Speicher der CControl einlesen lassen. Bedingung ist jedoch, dass die Tabelle im selben Verzeichnis liegt wie das Programm.

Damit wird unser Temperaturprogramm ganz einfach:

```

Define temp ad[1]
Define wert word

#main
looktab temperatur , temp , wert
print „Temperatur: „ wert / 10 ; „.“ wert mod 10
goto main

table temperatur „kty1.tab“

```

Wir erkennen aber hier einen gravierenden Nachteil. Unser kleines Basicprogramm ist ganz schön angewachsen. Der Compiler sagt: 560 Basic Bytes. Dieses liegt natürlich an der Tabelle. Wenn wir später noch mehr dieser analogen Bausteine hinzufügen wollen, so kommen immer wieder mächtige Tabelleneinträge hinzu. Da ist der 1621 schon bedeutend sparsamer, wenn man den I2C-Bus sowieso im Programm hat.

Noch ein Tip: Wenn Sie Temperaturen in einem vorhersehbaren Bereich messen wollen, so kann man die Tabelle kräftig abspecken. Wenn Sie nur die Raumtemperatur messen wollen, so reicht es wahrscheinlich, wenn der Sensor zwischen +10 Grad Celsius und +35 Grad ausgelesen werden muß. Die Tabelleneinträge davor und dahinter können gelöscht werden. Man muß sich lediglich merken, wieviele Einträge vor der 100 für 10 Grad standen. Ich habe dies einmal schnell gemacht – es sind 60 Einträge. Das Compilat ist sehr geschmolzen. Es sind noch 148 Basic Bytes, nur noch 1/3.

Im Programm wird die eine Zeile vervollständigt:

```

Alt:
looktab temperatur , temp , wert

Neu:
looktab temperatur , temp-60 , wert

```