Der I²C-Bus

1. Allgemeines

Das I²C-Bus-System dient zum Aufbau und Betrieb von Geräten, für die die Anzahl der Steuerleitungen oder deren Belastbarkeit nicht ausreichen. Es handelt sich dabei um eine von der Firma Philips entwickelte Steuermöglichkeit aus der Consumer-Eektronik, die auf die Einsparung von Leitungen abzielt. Der I²C-Bus besteht aus 4 Leitungen, der +5V- Leitung und der Masseleitung sowie der Datenleitung SDA und der Taktleitung SCL. Diese verbinden einen Steuercomputer (PC oder Mikroprozessor) den so genannten Master (Systeme mit mehreren Mastern werden hier nicht betrachtet) mit einem oder mehreren Peripheriebausteinen, den Slaves. Hier sind Datenspeicher, I/O-Portbausteine, AD- oder DA-Wandler, Uhrenbausteine und diverse Anzeigentreiber möglich.

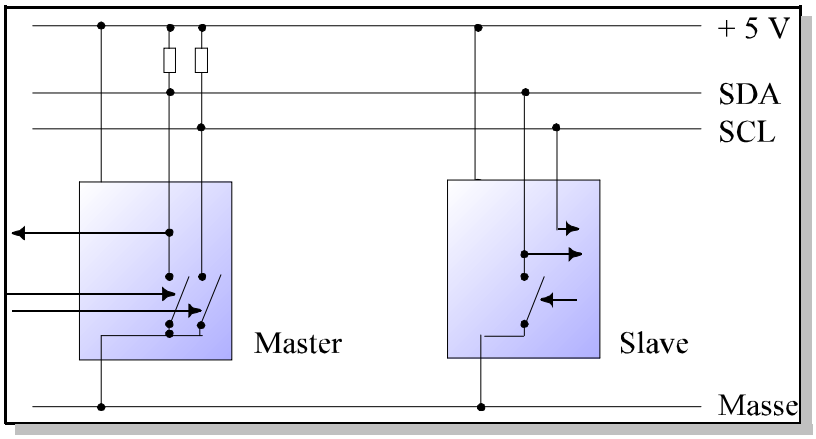

Abb. 1: Der I²C-Bus

1.1 Der Datenbus

Die Daten werden über die beiden Leitungen zwischen dem Master und einem der Slaves ausgetauscht. Da Daten vom Master sowohl gesendet als auch empfangen werden, arbeiten die Leitungen bidirektional. Die entsprechenden Anschlüsse der Bausteine sind also sowohl Eingänge als auch Ausgänge. Von Logikbausteinen ist bekannt, dass niemals 2 Ausgänge verbunden werden dürfen, was hier jedoch erfolgt. Daher sind eine spezielle Schaltungstechnik und ein strenges Datenprotokoll erforderlich, damit keine Fehler auftreten.

a) Schaltungstechnik
   Der 1-Zustand der Leitungen wird durch einen 4,7 kΩ - Widerstand nach +5V ausgeführt. Wirkt ein Anschluss eines Bausteins als Ausgang, so ist er beim Senden einer 1 hochohmig (offen) und beim Senden einer 0 wird über einen elektronischen Schalter

(Transistor) eine Verbindung zur Masse (GND) hergestellt. Die Taktleitung SCL wird nur vom Master angesteuert. Auf die Datenleitung SDA greifen Master und Slaves empfangend und sendend zu.

b) Datenprotokoll
Die gesamte Steuerung der Bausteine erfolgt über eine festgelegte Reihenfolge von Signalen auf den Leitungen SDA und SCL. Sie sind in 7 Grundbefehlen, den sog. I²C-Bus-Primitiven, zusammengefasst und stehen als Methoden in der COMX-Komponente zur Verfügung.

1.2 Die I²C-Bus Befehle (BASCOM)

Befehl	BASCOM		
Konfigurieren	config	SCL = PortB.7 SDA = PortB.5	
Start	i2cstart		
Stop	i2cstop		
Senden	i2cwbyte swert	swert: zu sendender Wert, kann auch eine Adresse sein	
Empfangen	i2crbyte ewert, Parameter	ewert: in dieser Variablen wird der empfangene Wert abgelegt; mögliche Parameter sind Ack und Nack	

Der I²C-Bus befindet sich normalerweise (z. B. nach dem Konfigurieren) im **Ruhezustand.** Dabei sind SDA und SCL im 1-Zustand (5V) und alle Bausteine inaktiv. Er kann auch durch den *Stop*-Befehl (am Ende einer Datenübertragung) hergestellt werden. Sendet der Master nun den Befehl *Start* so wird eine **Datenübertragung** eingeleitet, alle Slaves werden aktiviert. Nun muss der Master eine Adresse senden, um einen Peripheriebaustein auszuwählen. Eine Adresse besteht aus einem 8-Bit-Wort, bei dem die höherwertigen 4 Bit durch den Hersteller festgelegt werden, die nächsten 3 Bit lassen sich an den Anschlüssen des Bausteins schalten und das niederwertigste Bit (LSB) legt fest, ob der Baustein Daten senden oder empfangen soll. Durch die 3 Adressbits lassen sich 8 gleichartige Bausteine mit unterschiedlichen Adressen ansprechen. Mit dem Befehl *Senden* werden die 8 Adressbits seriell zu den Peripheriebausteinen übertragen. Danach erzeugt der Master auf der Leitung SCL einen weiteren Takt in dem der adressierte Baustein den Empfang der Adresse bestätigt. *Senden* besteht also aus insgesamt 9 Takten und ist in der COMX-Komponente als Funktion geschrieben, liefert also einen Rückgabewert.

Je nach LSB des Adressbytes ist der adressierte Peripheriebaustein nun auf Senden oder Empfangen geschaltet, und alle anderen Slaves gehen wieder in den Ruhezustand. Soll ein Slave weitere Daten empfangen, sendet der Master diese Daten mit dem Befehl *Senden* byteweise und beendet die Übertragung mit dem Befehl *Stop*.

Wie der Master Datenbytes von einem Slave empfängt wird später im Zusammenhang mit den Sensor-Bausteinen noch genauer dargelegt.

1.3 Das I²C-Bus–Protokoll genauer betrachtet

Master und Slave tauschen Informationen byteweise aus. Als Beispiel für einen solchen Informationsaustausch soll hier nun ausführlich dargelegt werden, wie der Master einen Slave durch Senden der zugehörigen Adresse aktiviert.

Der Slave habe die Adresse 01001100. Dieses Byte muss nun vom Master über den I²C-Bus geschickt werden.

0. Zunächst befinden sich alle Bausteine im **Ruhezustand**; er ist durch SDA = 1 und SCL = 1 gekennzeichnet.

1. Nun sendet der Master ein **Start**-Signal, indem er durch Schließen der beiden Schalter zuerst SDA und dann SCL auf 0 setzt. Dadurch werden alle Slaves des Bus-Systems in Bereitschaft versetzt: Sie warten jetzt auf das Adressbyte (vgl. Schritt 2).

2. Der Master **sendet** nun bitweise die Adresse 01001100. Dabei beginnt er bei dem höchstwertigen Bit (Bit 7), in unserem Beispiel also mit der 0, die am weitesten links steht.

2.1 Der Master legt Bit 7 an SDA; in unserem Fall wird/bleibt dazu der SDA-Schalter geschlossen. Kurz darauf öffnet der Master den SCL-Schalter, so dass die SCL-Leitung vom Zustand 0 in den Zustand 1 übergeht. Dies ist für die Slaves das Signal, das Bit 7 von der Datenleitung SDA entgegenzunehmen und in einem Puffer zwischenzuspeichern. Kurze Zeit später setzt der Master die Clockleitung SCL wieder auf 0.

2.2 Der Master legt nun Bit 6 an SDA; in unserem Fall wird dazu der SDA-Schalter geöffnet; durch den Pullabwiderstand erhält die SDA-Leitung jetzt den Zustand 1. Kurz darauf öffnet der Master wieder den SCL-Schalter, so dass die SCL-Leitung erneut vom Zustand 0 in den Zustand 1 übergeht. Dies ist für die Slaves das Signal, das nächste Bit (Bit 6) von der Datenleitung SDA entgegenzunehmen und in einem Puffer zwischenzuspeichern. Kurze Zeit später setzt der Master die Clockleitung SCL wieder auf 0.

2.3 Bit 5 bis Bit 0 werden auf gleiche Weise übertragen.
-2.8

2.9 Alle Slaves vergleichen nach dem Empfang des Bit 0, ob das empfangene und zwischengespeicherte Byte mit ihrer eigenen Adresse übereinstimmt. Derjenige Baustein, welcher Übereinstimmung feststellt, legt nun SDA auf 0; dieses Signal wird **Acknowledge**-Signal (ACK) genannt. Der Master belässt/schaltet derweil SDA auf 1 und legt SCL auf 1. Er prüft nun, ob die SDA-Leitung auf 0 oder 1 liegt. Liegt sie auf 1, geht der Master davon aus, dass sich kein Slave mit der gesendeten Adresse im Bus-System befindet. Liegt auf SDA aber ein 0, zeigt dies dem Master, dass der gewünschte Slave gefunden wurde. Der Master setzt nun seinerseits SCL wieder auf 0 und zeigt damit dem Slave, dass er dessen Acknowledge-Signal erhalten hat. Daraufhin öffnet der Slave wieder seinen SDA-Schalter (SDA wird dadurch auf 1 gesetzt, so dass der Master im Folgenden die Datenleitung wieder benutzen kann); der Slave ist jetzt zum Empfang eines weiteren Bytes (Datenbyte) bereit. Alle anderen Slaves gehen wieder in den Ruhezustand über, bis wieder ein Startsignal erfolgt.

3. Nachdem durch entsprechende Schritte wie bei 2.1 - 2.9 ein oder mehrere Datenbytes übertragen worden sind, legt der Master sowohl SDA als auch SCL wieder auf 1. Durch dieses **Stop**- oder **Init**-Signal wird der ursprüngliche Ruhezustand wiederhergestellt.

Diese in den Punkten 0 bis 2.9 dargestellte Signalfolge kann graphisch in einem so genannten **Timing-Diagramm** dargestellt werden. Unser Fall ist in Abb. 2 wiedergegeben.

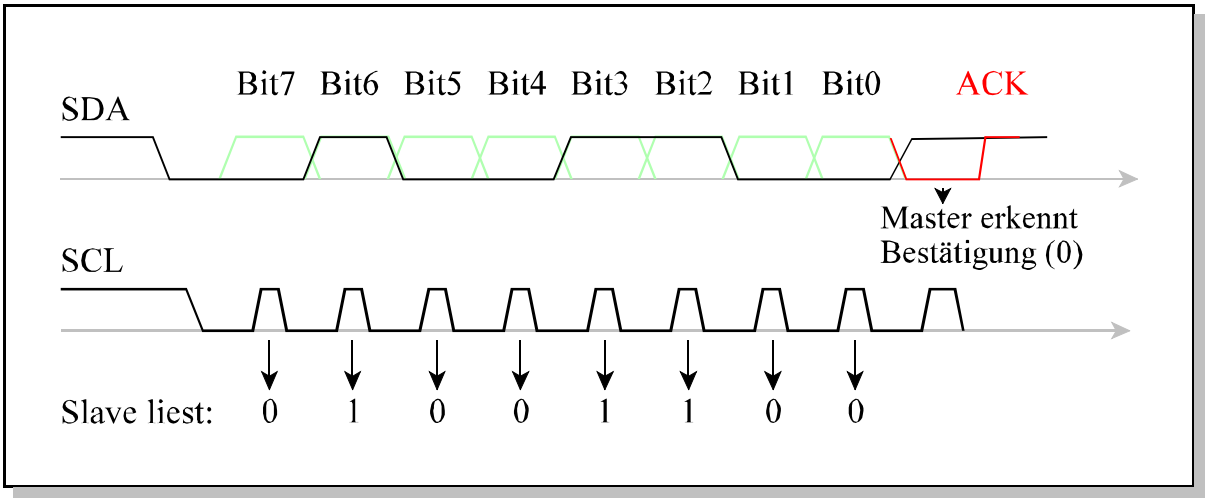

Abb. 2: Die Signale des Masters sind schwarz, die des Slaves rot dargestellt. Grün sind alternative Bitwerte des Masters angedeutet.**Abbildung 2**

1.4 Ein einfaches BASCOM-Programm

Die Werte 0 bis 7 sollen an den PCF8574-Baustein mit der Adresse 78 gesendet werden.

```
' Attiny-Platine von E. Eube, G. Heinrichs und U. Ihlefeldt
' plus I2C-Universalplatine
'-------------------------------------------------------------------------

$regfile = "attiny2313.dat"                                 'Attiny2313
$crystal = 4000000                                          '4 MHz
$baud = 9600

'*************************************************************
'******************* Deklarationen ***********************

Dim pcf_adr As Byte
Dim I As Byte

'****************** Initialisierung **********************

Ddrb = &B11111111               'Port B als Ausgangsport
Ddrd = &B01110000               'D4, D5, D6 als Ausgang; Rest als Eingang
Portd = &B10001111              'Eingänge auf high legen

Config Scl = Portb.7             'Konfigurieren der I2C-Leitungen
Config Sda = Portb.5

Pcf_adr = 78                     'Adresse von PCF8574

'*************************************************************
'******************** Hauptprogramm **********************

for I=0 to 7
  i2cstart
  i2cwbyte pcf_adr
  if err = 0 then i2cwbyte I   'In der Systemvariablen err wird das Ack-Bit
                               hinterlegt
  i2cstop
  wait 1
next I
```

1.5 Anschließen von I2C-Bausteinen an die Attiny-Platine

Der Anschluss erfolgt über die Westernbuchse (in Abb. 3 oben rechts); diese versorgt die Bausteine aich mit elektrischer Spannung. Wichtig: Auf der Attiny-Platine müssen die Jumper bei PortB.5 und PortB.7 gesetzt werden. Sie verbinden diese Ports mit den Pull-Up-Widerständen.

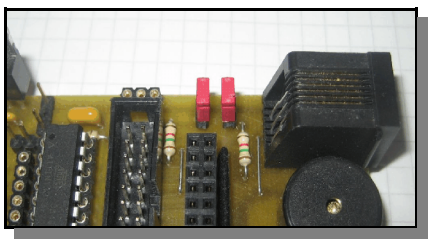

Abb. 3

2. Hardware

2.1 Die Universal-Slave-Platine

Die Slave-Platine dient zunächst zur Erprobung der I²C-Übertragung (Empfangen ***und*** Senden). Mit ihr können - dank eines zusätzlichen Treiberbausteins - aber auch schon kleinere Geräte gesteuert werden.

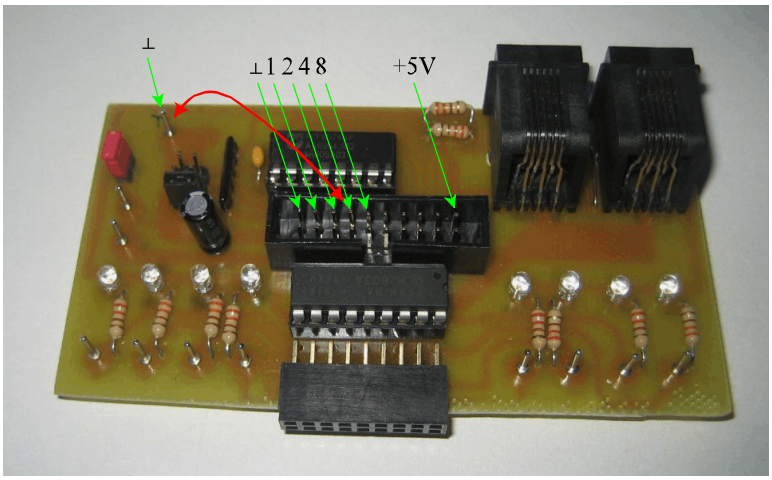

Abb. 4

2.2 Ansteuerung der Portbausteine PCF 8574 und PCF 8574A

Um den Baustein anzusprechen, muss zunächst seine Adresse vom Master gesendet werden. Es gibt den Baustein in 2 verschiedenen Ausführungen als PCF 8574 und als PCF 8674A. Sie unterscheiden sich durch ihre Grundadressen. Die Adressen sind wie folgt aufgebaut:

Baustein	D7	D6	D5	D4	D3	D2	D1	D0
PCF 8574	0	1	0	0	Ad2	Ad1	Ad0	R/W
PCF 8574A	0	1	1	1	Ad2	Ad1	Ad0	R/W

Die Grundadresse des Bausteins PCF 8574 beträgt also dezimal 64 und die des PCF 8574 A beträgt 112. Die Adresseingänge der Bausteine lassen sich auf der Platine mit Jumpern mit GND verbinden, über Widerstände liegen sie an +5V. Bei nicht gesteckten Jumpern ist also der jeweilige Wert (2, 4 oder 8) zur Grundadresse zu addieren. Sollen Daten zum Baustein gesendet

werden, ist zunächst diese Adresse zu übertragen und danach weitere Datenbytes.

Sollen Daten vom Baustein gelesen werden, so ist der Adresswert um l zu erhöhen und diese Adresse zu senden, danach kann man mit Lesen und Acknowledge Datenbytes empfangen. Dabei ist zu beachten, dass die Datenleitungen des PCF 8574 ebenso bidirektional arbeiten wie die Busleitung SDA. Da die Datenleitungen ihren Zustand nach Ende einer Übertragung bis zur nächsten Übertragung beibehalten, müssen die Leitungen, die zum Lesen benutzt werden sollen, vorher als Ausgang in den l-Zustand gesetzt werden (siehe Beispiel unten).

2.3 Bedienung der Universalplatine

Betreibt man die Universalplatine nur als LED-Anzeige (*Standardmodus*), so ist der Anschluss einer weiteren Quelle nicht erforderlich; ihre Benutzung ist dann völlig problemlos.

Die Platine bietet aber deutlich mehr Möglichkeiten; diese werden im Folgenden beschrieben.

Wegen der vielseitigen Verwendbarkeit der Platine ist bei falscher Beschaltung eine Zerstörung der angeschlossenen Platinen möglich. Bitte beachten Sie deswegen die folgenden Ausführungen, wenn Sie die Universalplatine nicht im Standardmodus betreiben wollen.

2.3.1 Externe Spannungsquelle

Die Ausgänge der Platine können sowohl mit der 5 V Betriebsspannung des I2C-Bus-Systems als auch mit einer externen Spannungsquelle betrieben werden. Ist der Jumper zwischen den Leiterbahnen +5V und U+ (Beschriftung auf der Leiterbahnseite der Platine) gesteckt, so darf über den Lötnagel bei U+ keine externe Spannung zugeführt werden.

2.3.2 Benutzung der Ausgänge

Als Ausgänge lassen sich entweder die Lötnägel verwenden, oder man benutzt die 2x9-polige Steckerleiste, an die die Verbraucher angeschlossen werden. Die Ausgangsleitungen werden über den invertierenden Treiberbaustem ULN 2803 gegen GND geschaltet, d.h. bei einer logischen l der entsprechen Datenleitung des PCF ist der Lötnagel bzw. das an der 2x9-poligen Steckerleiste angeschlossene Kabel mit GND verbunden und die zugehörige LED leuchtet. Bei einer logischen 0 ist der Ausgang offen. Ein Verbraucher muss also mit (s)einem (negativen) Anschluss an einen der 8 Ausgänge und mit dem anderen Anschluss an die gemeinsame Leitung U+ angeschlossen werden. Diese Leitung U+ kann nun entweder über die 5 V Leitung des I2C-Bus gespeist

werden: dann ist der Jumper zwischen diese Leitungen zu stecken. Allerdings erfolgt die Stromzuführung über die dünne Telefonleitung, die mit maximal 100 mA für alle Verbraucher belastet werden sollte. Verwendet man Verbraucher mit einer anderen Spannung oder höheren Stromstärken, so ist der Jumper zu entfernen und über die Lötnägel bei GND und U+ eine externe Spannung zuzuführen.

2.3.3 Benutzung der Eingänge

Die 8 Datenleitungen des I²C-Bus-Bausteins arbeiten nach dem „wired AND" Prinzip. Sie werden über einen Widerstand in den logischen l Zustand gesetzt und können entweder vom Baustein selbst oder von außen durch eine Verbindung mit GND auf logisch 0 gesetzt werden. Jede Leitung kann also z.B. durch einen Schalter, Taster oder Fotozelle, die zwischen dieser Leitung und GND angeschlossen ist, entsprechend gesetzt werden. *Diese Leitung des Bausteins muss dann aber als Ausgang auf logisch l programmiert sein*. Die Sensoren werden an der 2xl0-poligen Stiftleiste mit Kragen angeschlossen und zwar zwischen einer der Datenleitungen und GND (auf der Leiterbahnseite beschriftet). Der +5V-Anschluss dieser Stiftleiste dient z.B. zum Anschluss einer Leuchtdiode, um eine Lichtschranke aufzubauen.

3. Messen mit dem I²C-Bus

Zahlreiche Sensoren existieren im Handel, welche über einen I²C-Bus ihre Messwerte an einen Master senden. Auch unsere Universalplatine kann als Eingabe-Platine benutzt werden; wie man dabei vorgeht, wird weiter unten noch ausführlich dargestellt. Zunächst soll aber dargestellt werden, wie der Master Daten vom Slave bezieht.

3.1 Daten lesen mit I2C (READ-Vorgang)

Grundsätzlich wird beim Lesen dasselbe Protokoll wie beim Senden (WRITE-Vorgang) benutzt. Deswegen können wir uns hier kürzer fassen und für Details auf den Abschnitt 3 von Kapitel 1 verweisen.

Zunächst wird der Bus in der üblichen Weise initialisiert, ein Startsignal gegeben und die Adresse des Slaves gesendet. Die Adresse des Slaves ist immer ungeradzahlig, wenn er zum Lesen angesprochen wird.

Der adressierte Slave gibt wieder ein Acknowledge-Signal zurück und schiebt den aktuellen Messwert in das Datenregister. Bei den nächsten Clocksignalen des Masters werden die einzelnen Bits dieses Registers ausgelesen, beginnend mit den MSB, dem höchstwertigsten Bit. Dazu muss natürlich der Master den SDA-Schalter (vgl. Abb. 1 offen lassen) und bei jedem

Clock-Signal den Zustand der SDA-Leitung abfragen. Diese Aufgabe übernimmt der i2crbyte-Befehl bei BASCOM:

```
i2crbyte messwert, ACK
```

Hier wird der Messwert in der Variablen `messwert` abgelegt.Will man aus demselben Baustein weitere Messwerte auslesen, so muss der Master ein Acknowledge-Signale senden; dies geschieht durch den zusätzlichen Parameter ACK. Dadurch wird der nächste aktuelle Messwert in das Datenregister des Slaves geschoben und der Slave steht für einen nächsten Lesevorgang bereit.

Sollen jedoch keine weiteren Messwerte gelesen werden, so muss der Master nach dem Empfang des Datenbytes ein Kein-Acknowlege (No-Acknowledge) - Signal senden; dies geschieht durch den zusätzlichen Parameter NACK.

3.2 Der Temperatursensor LM 75

Mit der Temperatursensor-Platine (Abb. 5) kann man Temperaturen zwischen -55 °C und +125 °C messen. Die voreingestellte Adresse ist auf der Buchse abzulesen. Wenn man die Lötpunkte A und B miteinander leitend verbindet, erhält der Baustein eine andere Adresse.

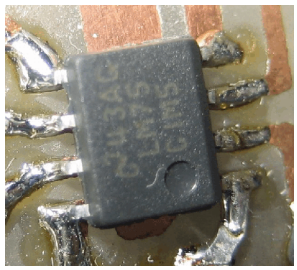

Abb. 5

Der benutzte Baustein LM 75 besitzt zwei Datenregister, die vom Master ausgelesen werden können. Die Vorgehensweise ist folgende:

<Start>
Adresse <schreiben>
<lesen> Byte1
<Acknowledge>
<lesen> Byte2
<KeinAcknowledge>
<Stop>

Die Bedeutung der Bytes ist folgende:

Byte1:

VZ	b6	b5	b4	b3	b2	b1	b0

Byte2:

n	X	X	X	X	X	X	X

VZ: Vorzeichenbit (1 entspricht -, 0 entspricht +)
b6-b0: Temperaturwert in °C
n: Erste Nachkommastelle des Temperaturwerts (1 entspricht 0,5 °C, 0 entspricht 0,0°C)
X: irrelevant

Hinweis: Um das Bit 7 vom zweiten Byte abzufragen, kann man einfach überprüfen, ob der Wert größer oder gleich 128 ist oder nicht.

Beispiel:

Byte1:

0	0	0	1	0	0	1	0

Byte2:

1	X	X	X	X	X	X	X

Das Ergebnis ist +18,5 °C.

Das zweite Byte interessiert häufig nicht; dann bricht man mit einem KeinAcknowledge-Signal nach dem ersten Lese-Vorgang ab.

Das folgende BASCOM-Programm misst fortwährend Temperaturwerte (ohne Nachkommastelle und ohne Berücksichtigung des VZ-Bits) und gibt sie über die serielle Schnittstelle an ein Terminalprogramm (Initialisierungen und Konfigurierungen sind weggelassen):

```
Do
  I2cstart
  I2cwbyte Lm75_adr
  I2crbyte Wert , Nack        'kein ACK, weil LM75 sonst die Nachkommastelle
                               senden möchte
  I2cstop
  Wait 1
  Printbin Wert
Loop
```