**E. Eube - G. Heinrichs - U. Ihlefeldt**

# Das $I^2C$-Bus-System

## Aufbau, Funktionsweise und Programmierung

**Fortbildungsveranstaltung**
**in Mönchengladbach**
**am 23.08.2008**

# 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 Phillips entwickelte Steuermöglichkeit aus der Consumelektronik, 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.

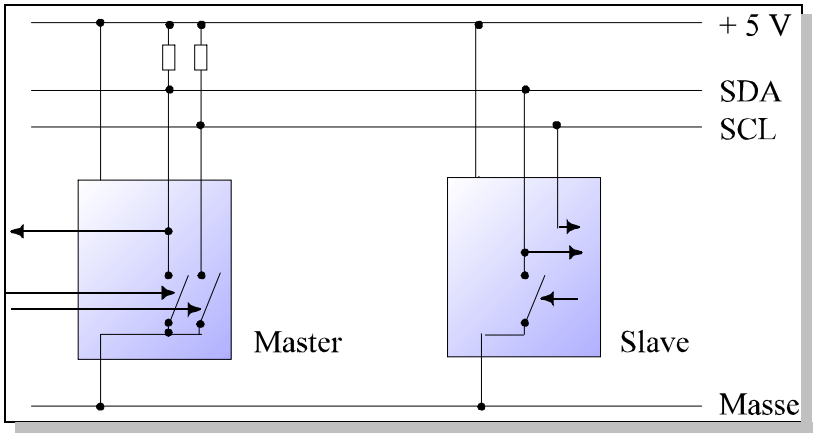


**Abb. 1**: Der I²C-Bus

## 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

Zur Steuerung genügen die Befehle:

| Befehl | in der COMX-Komponente | | |
|---|---|---|---|
| | Typ | Bezeichnung | Rückgabewert |
| Init | Procedure | initI2C | |
| Start | Procedure | startI2C | |
| Stop | Procedure | stopI2C | |
| Senden | Function | ausgebenI2C(wert) | Boolean |
| Empfangen | Function | lesenI2C | Byte |
| Acknowledge | Procedure | acknowledgeI2C | |
| Kein Acknowledge | Procedure | keinAcknowledgeI2C | |

Der I²C-Bus befindet sich normalerweise im **Ruhezustand.** Dabei sind SDA und SCL im 1-Zustand (5V) und alle Bausteine inaktiv. Er wird hergestellt durch die Befehle *Init* (nach einer Störung) oder *Stop* (am Ende einer Datenübertragung). 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.

Das Vorhandensein des adressierten Bausteins kann so beispielsweise in VB auf folgende Weise abgefragt werden:

```
Dim istda As Boolean
.
```

```
.
.
istda = Comx1.ausgebenI2C(Adresse)
If Not istda Then MsgBox ("Baustein antwortet nicht")
 .
 .
```

Weitere Informationen zur Adresse finden Sie bei den einzelnen Bausteinen.

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

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.

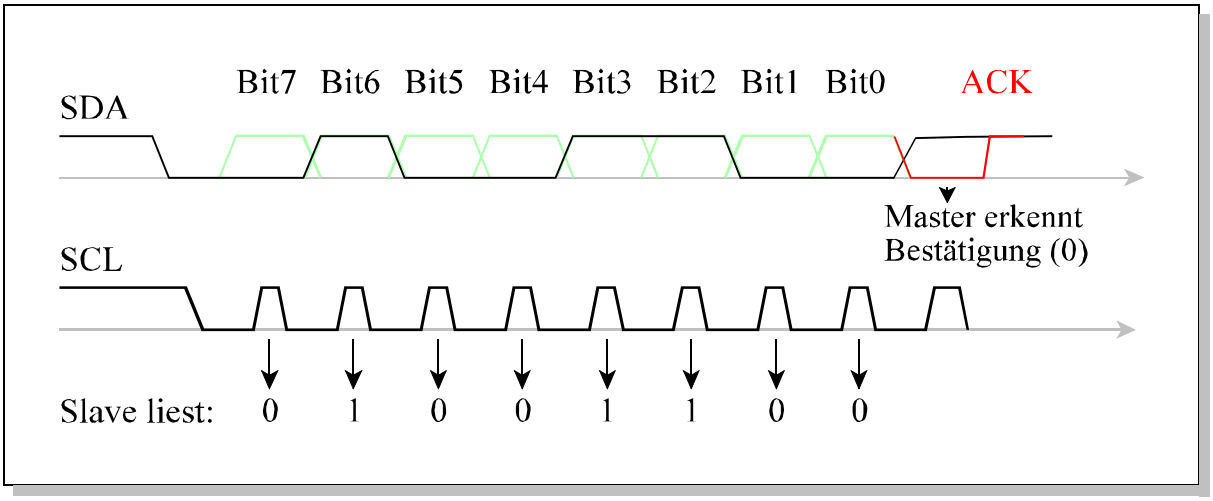


**Abb. 2**: Die Signale des Masters sind schwarz, die des Slaves rot dargestellt. Grün sind alternative Bitwerte des Masters angedeutet.

## 1.4 Ein einfaches Visual-Basic-Programm

Visual Basic (z. B. Version 5) wird gestartet; ein neues Projekt wird angelegt. Zunächst wird über das Menü Projekt - Komponenten die ComX-Komponente (ComX3) in die Komponentenleiste aufgenommen (Vgl. Anleitung zu ComX). Anschließend werden auf das leere Formular gelegt:

- 1 ComX-Komponente
- 2 Label-Komponenten
- 2 Textfeld-Komponenten
- 1 Schaltflächen-Komponente

Das Formular sieht dann so aus wie in Abb. 3.

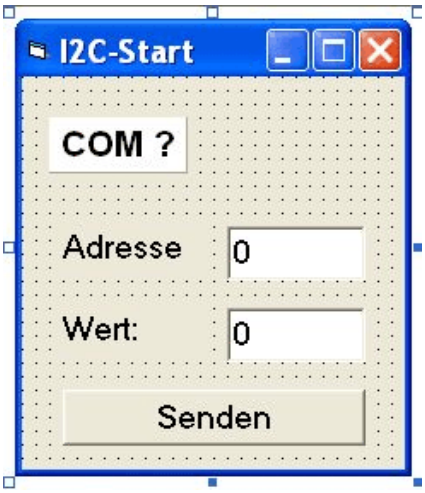


**Abb. 3**: Das Formular

Bei dem folgenden Programm ist zu beachten, dass VB automatisch die in den Textfeldern eingegebenen Zeichenketten in Zahlen konvertiert; bei vielen anderen Entwicklungsumgebungen (z. B. Delphi) muss diese Umwandlung explizit programmiert werden.

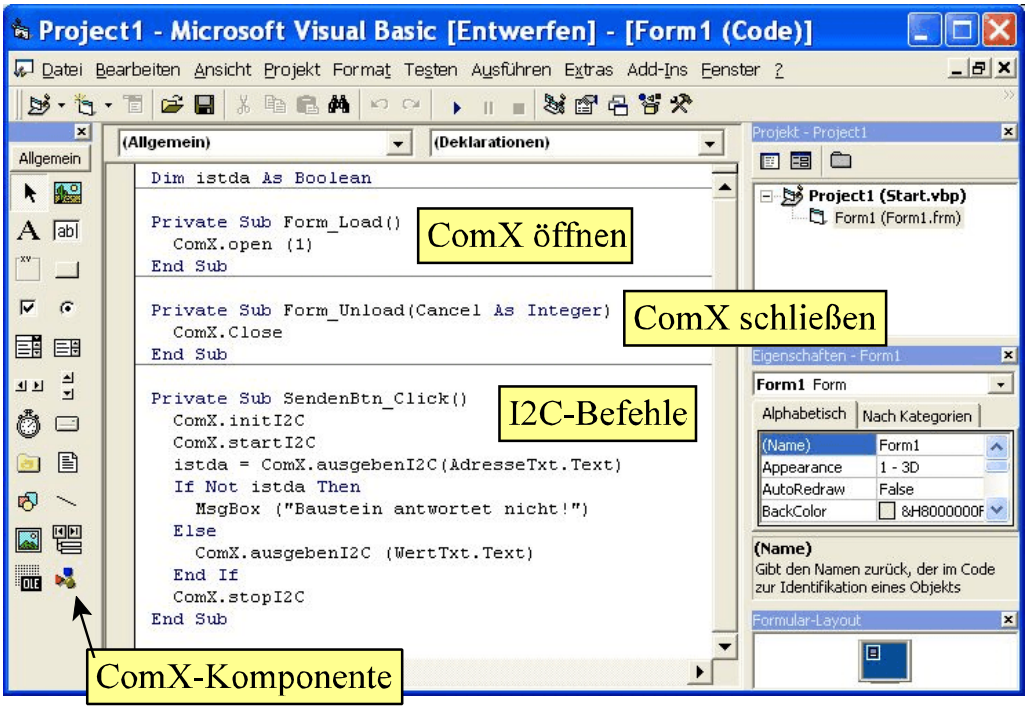


**Abb. 4**: Der Quellcode

# 2. Hardware

## 2.1 Grundset

Zum Grundset gehören eine Master-und eine Slave-Platine. Die Masterplatine stellt auch eine geregelte Spannungsquelle für die Slaves zur Verfügung. 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.

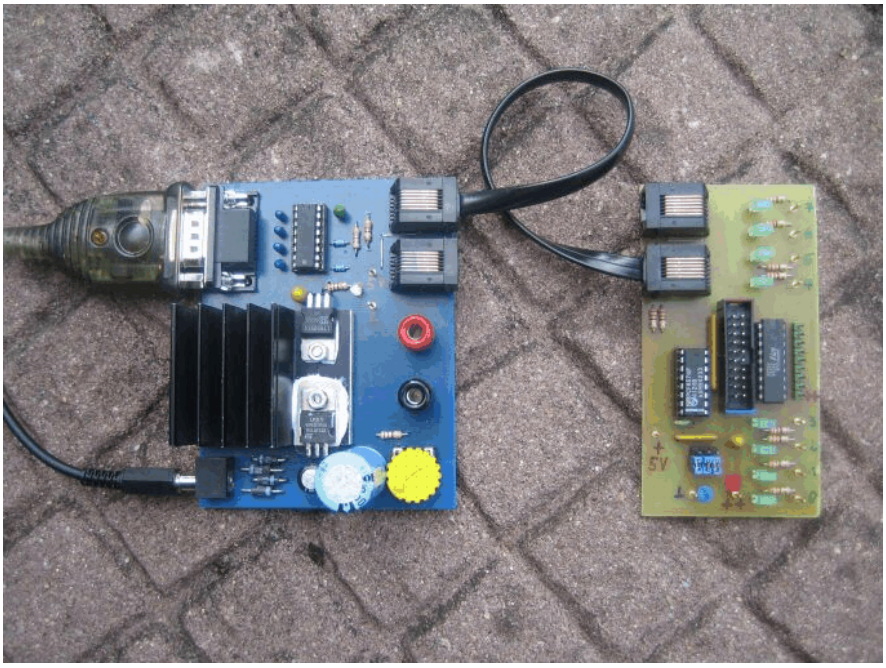

**Abb. 5**: Master (links) und Slave (rechts)

## 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.

## 2.4 Erweiterungen

Verschiedene Platinen sind in Vorbereitung: z. B. Leuchtziffernanzeige, Temperatursensor und Schrittmotorenansteuerung.

# 3. Aufgaben zum Steuern mit I²C

## Aufgabe 1

In dieser Aufgabe sollen einzelne Werte auf der Universal-Platine als Dualzahlen angezeigt werden. Ziel der Übung ist es, mit der COMX-Komponente und den neuen I2C-Befehlen vertraut zu werden.

Vorausgesetzt wird, dass auf dem Rechner die COMX-Komponente und VisualBasic installiert sind.

1.1 Schließen Sie die Interfaceplatine an das Netzgerät an, verbinden Sie diese Platine einerseits über das serielle Kabel mit der COM-Schnittstelle des Rechners und über das I2C-Kabel mit der Universalplatine (s. Abb. 5)
Achten Sie darauf, dass die Brücken für die Adresseinstellung auf der Universalplatine "entfernt" sind; dann ist die Adresse des PCF8574A-Bausteins gleich 126. (Beim PCF8574-Baustein ist die Adresse dann 78.)

1.2 Kopieren Sie das Verzeichnis *Aufgabe1* aus dem Verzeichnis *Source* der I2C-CD auf ihre Festplatte und öffnen Sie dieses Verzeichnis. Öffnen Sie die Datei *Aufg1.vbp*. VisualBasic wird dann automatisch gestartet und das Projekt aus Abb. 6 wird geladen; dieses enthält schon ein fertiges Formular sowie einige wenige Zeilen Quelltext.

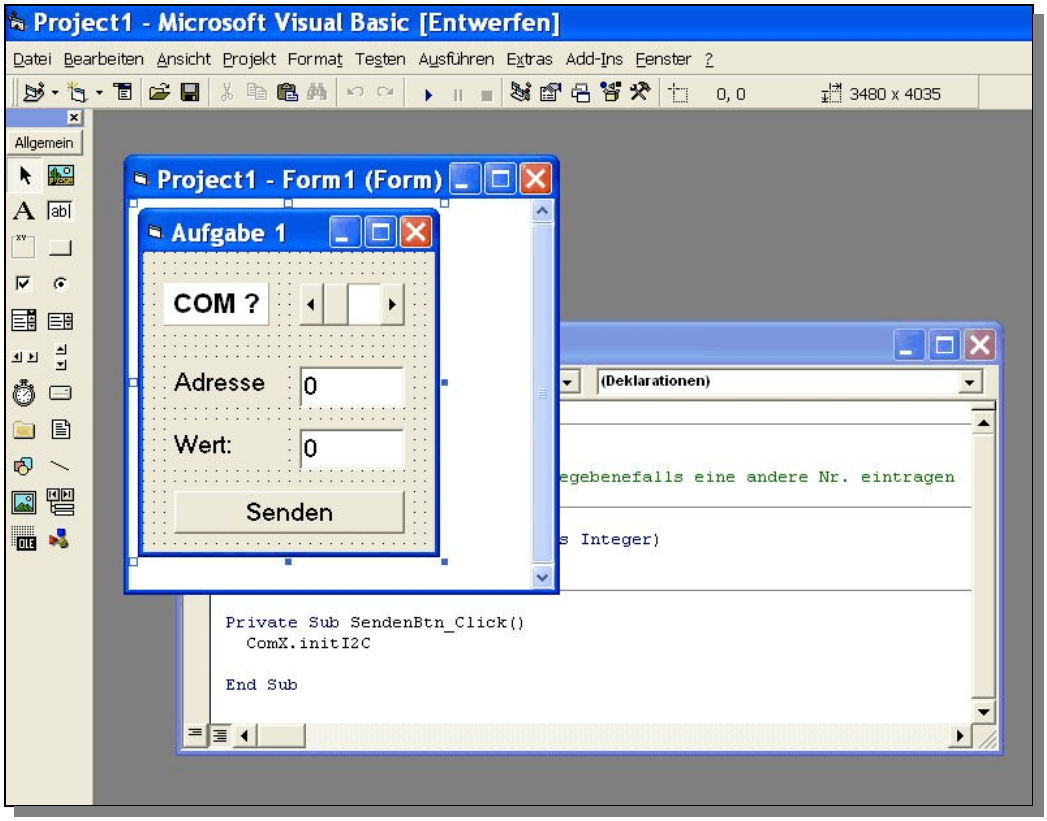


**Abb. 6**

1.3 Die COMX-Komponente ist hier schon in das Formular eingefügt und wird beim Starten des Programms automatisch geöffnet und beim Beenden geschlossen.
Die Textfelder und der Scrollbalken werden erst später benutzt.

Starten Sie das Projekt und kontrollieren Sie, ob die COMX-Komponente dabei grün wird. Wird die Komponente rot eingefärbt, gibt es die Schnittstellennummer nicht oder sie ist blockiert. Gegebenenfalls müssen Sie VB neu starten oder mit dem Windows-Gerätemanager die Schnittstellennummer Ihres Rechners in Erfahrung bringen.

> ***Wichtig - Wichtig - Wichtig!***
> Beenden Sie ein laufendes VB-Projekt immer mit dem  des Projektformulars und niemals mit der  -Schaltfläche der VB-Entwicklungsumgebung. Nur so wird garantiert, dass die Schnittstelle (über das onClose-Ereignis und die zugehörige Prozedur) geschlossen wird. Wenn ein Programm wegen eines Fehlers automatisch unterbrochen wird, sollten Sie stets versuchen es fortzusetzen und anschließend regulär zu beenden.

Die COMX-Komponente ist grün? Dann sollten Sie die folgende Aufgabe bearbeiten:

1.4 Fügen Sie unter ComX.initI2C in der *richtigen* Reihenfolge die COMX-Befehle an für:

- *Wert 5 übertragen*
- *Starten der Übertragung*
- *Übertragung stoppen*
- *Adresse 126 übertragen*

Hinweis: Die Funktion “ausgeben” kann hier wie eine Prozedur (d. h. ohne Rückgabewert) benutzt werden.

Speichern Sie Ihr Projekt ab und testen Sie es aus. Wenn Sie die Schaltfläche “Senden” betätigen, leuchten die 2 LEDs auf. Welche?

1.5 Ändern Sie jetzt Ihr Projekt folgendermaßen ab: Der zu übertragende Wert soll dem Textfeld “WertTxt” entnommen werden. Der Zugriff auf dieses Feld erfolgt mit:

```
... = WertTxt.Text
```

(Eine Umwandlung von Text nach Zahl braucht bei VB nicht vorgenommen zu werden.)

Speichern Sie Ihr Projekt wieder ab und testen Sie die Übertragung unterschiedlicher Werte aus.

1.6 Nun soll auch die Adresseingabe flexibel gestaltet werden. (Den Namen des Adressfeldes erkennen Sie, wenn Sie das Adressfeld im Formular einmal anklicken und im Objektinspektor nachschauen!)

Testen Sie Ihr Projekt mit unterschiedlichen Adressen aus.

1.7 Das Programm soll nun eine Rückmeldung geben, wenn der I²C-Baustein keine Rückmeldung gegeben hat. Dazu nutzt man aus, dass die Funktion "ausgeben" den Wert true zurückgibt, wenn ein Rückmeldung erfolgt (Acknowledge), ansonsten aber den Wert "false" zurückgibt.

Durch die Anweisung

```
IstDa = Comx.ausgeben(126)
```

wird also

1. die Adresse 126 ausgesendet
2. In der Variablen `IstDa` der Wert `true` bzw. `false` gespeichert, je nachdem ob der Baustein sich gemeldet hat oder nicht.

Ergänzen Sie Ihr Programm nun so, dass eine entsprechende Meldung (MsgBox) erscheint, wenn die Baustein nicht gefunden wurde. Ansonsten soll der gewünschte Wert wie üblich ausgegeben werden.
*Hinweis*: Benutzen Sie eine if - then - else - endif - Struktur.

Gibt es Probleme? Dann sollten Sie Ihr Programm mit Abb. 4 des Skripts vergleichen!

1.8 Tragen Sie nun im Objektinspektor beim AdressTxt-Feld unter der Eigenschaft Text den Wert 126 ein. Dieser Wert ist dann künftig der Vorgabewert.

Ändern Sie die Jumper (Brücken) so ab, dass die Adresse Ihres PCF 8574A - Bausteins nun 124 ist; dazu muss einer der Jumper gesetzt werden!) Versuchen Sie nun diesen I²C-Baustein unter der alten und unter der neuen Adresse zu erreichen.

Entfernen Sie anschließend wieder den Jumper.

**Aufgabe 2**

2.1 Kopieren Sie die Dateien Ihres Projektes von Aufgabe 1 in ein neues Verzeichnis *Aufgabe2*. Laden Sie Ihr Projekt.

2.2 Die Universalplatine soll nun im Dualsystem von 0 bis *n* zählen; dabei soll *n* die im Feld "WertTxt" angebene Zahl sein. Das Zählen soll im Sekundenrhytmus erfolgen.

Hinweise: Benutzen Sie eine Zählschleife und die delay-Methode von ComX.

**Achtung**: Wählen Sie beim Testen den Wert von *n* zunächst sehr klein!

2.3 Die Zeitverzögerung beim Zählen soll durch den Scrollbalken einzustellen sein.

Ein mögliches Ergebnis finden Sie im Verzeichnis *Lösungen*.

## Aufgabe 3

Diese Aufgabe müssen Sie im Team mit Ihrem Nachbarn/Ihrer Nachbarin bearbeiten. Es geht nämlich darum, zwei verschiedene Universalplatinen an ein einziges Interface (und damit auch an einen einzigen Rechner) anzuschließen.

*Hinweis*: Wenn Sie ein Projekt mit der ComX-Komponente neu erstellen, muss diese zunächst aus der Komponentenleiste auf das Formular gezogen werden. Das zugehörige Icon sieht so aus:

Ikon für COMX3

Sollte sich dieses Ikon nicht in der Komponentenleiste befinden, müssen sie COMX erst wie im COMX-Handbuch beschrieben in VB einbinden.

3.1 Geben Sie einer der beiden Platinen eine neue Adresse (vgl. 1.8). Schließen Sie beide Platinen an eine einzige Interfaceplatine an. Testen Sie mit dem Programm aus Aufgabe 1 nach, ob sich beide Universalplatinen getrennt ansteuern lassen.

3.2 Legen Sie ein neues Verzeichnis für die Aufgabe 3 an. Erstellen Sie ein Programm, bei welchem die LEDs auf den beiden Platinen im stetigen Wechsel ein- und ausgeschaltet werden (Wechsellicht).

## Aufgabe 4

Diese Aufgabe können Sie wieder solo bearbeiten.

4.1 Programmieren Sie ein Lauflicht: Zunächst soll die LED 0, dann die LED 1, anschließend die LED 2 ... leuchten. Nach der LED 7 soll wieder die LED 0 leuchten. Der zeitliche Abstand soll wieder durch einen Scrollbalken eingestellt werden können.

4.2 Ein beliebiges Bitmuster soll durch das Ringschieberegister aus 4.1 rotieren.

4.3 Aufgabe 4.1 oder Aufgabe 4.2 können auch für 2 Universalplatinen programmiert werden.

## Aufgabe 5

Programmieren Sie eine Discolicht-Anlage. Als Discolicht sollen die LEDs der Universalplatine dienen. Die LEDs sollen Programm-gesteuert an- und ausgehen. Das Programm kann z. B. durch Benutzung einer Listbox flexibel gehalten werden (s. Abb. 7). Nützlich wäre es auch, wenn diese Daten abgespeichert und wieder geladen werden könnten. Viel Spaß dabei!


**Abb. 7**

## Aufgabe 6

An einen I²C-Bus können mehrere Slaves angeschlossen werden. Überlegen Sie sich, wie ein Programm aussehen müsste, dass die Adressen aller angeschlossenen Slaves ermittelt und anzeigt. Es sollen nur Slaves mit geradzahliger Adresse (also Slaves im Write-Modus) berücksichtigt werden. (Die Kontrolle von Slaves im Read-Modus erfordert zusätzliche Kenntnisse; wir werden darauf später noch eingehen!)

# 4. 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.

## 4.1 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 die Funktion lesenI2C der ComX-Komponente:

```
messwert = ComX.lesenI2C
```

Will man aus demselben Baustein weitere Messwerte auslesen, so muss der Master ein Acknowledge-Signale senden; der zugehörige Befehl lautet acknowledgeI2C. 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; der zugehörige Befehl lautet keinAcknowledgeI2C.

***Achtung***: Ohne das Kein-Acknowledge-Signal sendet der Slave bei jedem Clock ein Bit seines Datenregisters; ja sogar Stop- und Start-Signal werde als Clock-Signale fehlinterpretiert und führen nur zum Senden eines weiteren Bits!

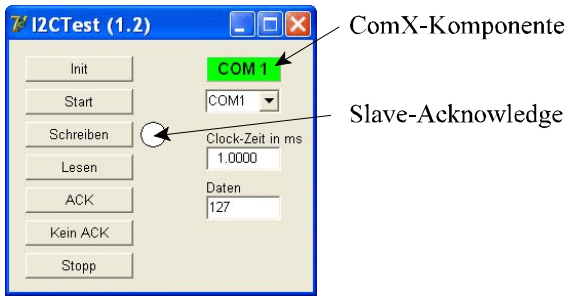


**Abb. 8**

## 4.2 Messen mit der Universalplatine

Mithilfe des Testprogramms I2CTest2 wollen wir nun einen Messvorgang schrittweise durchführen. Wir schließen zunächst unsere Universalplatine über den Master an die serielle Schnittstelle des Rechners an und starten dann das Programm I2CTest2. Nun stellen wir als erstes die Nummer der seriellen Schnittstelle ein; die ComX-Komponente sollte dann eine grüne Farbe besitzen (Abb. 8).

In das Datenfeld schreiben wir jetzt die Adresse der Universalplatine, z. B. 127, und betätigen die Schaltflächen

<Stop/Init>
<Start>
<Schreiben>

Dadurch wird der PCF8574-Baustein im READ-Modus adressiert. Wenn er korrekt adressiert wurde, wird dies mit einem grünen Signal neben der Schreiben-Schaltfläche angezeigt (Slave-Acknowledge).

Nun legen wir das Bit 3 (Stufenwert 8 im Zweiersystem) der Universalplatine auf 0; dazu verbinden wir die entsprechende Signalleitung des Pfostensteckers mit dem Masseanschluss der Platine (s. Abb. 9).

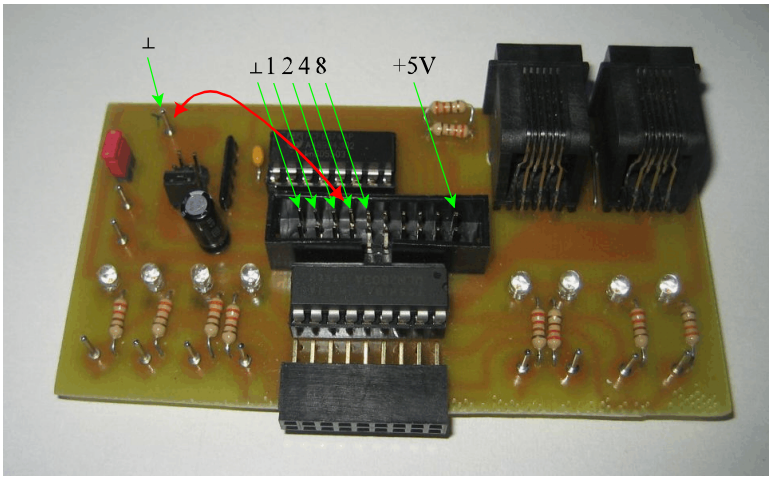


**Abb. 9**

***Vorsicht***: Die beiden linken Pins des Pfostensteckers dürfen nicht mit der Masse verbunden werden, weil sie mit der Versorgungsspannung +5 V belegt sind; ein Kurzschluss wäre die Folge.

Die LED Nr. 3 mit dem Stufenwert 8 erlischt; die von den LEDs angezeigte Zahl ist damit 11110111(2) = 255 - 8 = 247. Jetzt betätigen wir die Schaltfläche

<Lesen>

Im Datenfeld erscheint die vom Slave übertragene Zahl 247! Wir haben unseren ersten Lesevorgang erfolgreich durchgeführt.

Wir bestätigen mit

<KeinAcknowledge>

und beenden damit dem Messvorgang.

Hinweis: Soll der PCF 8574 zum Messen benutzt werden, so müssen seine Ausgangsleitungen auf 1 liegen. Dies ist z. B. beim Einschalten automatisch der Fall. Andernfalls muss zuvor die Zahl 255 ausgegeben werden.

Wir probieren:

- drei verschiedene Messwerte hintereinander erfassen. Muss das Bitmuster vor oder nach dem Acknowledge-Signal eingestellt werden?

  Notiz:

- einen einzigen Messwert (0) lesen, ein Acknowledge-Signal geben und versuchen, anderen nicht existierenden Baustein (120) zu adressieren.

  Notiz:

***Zum Nachdenken...***

An den I2C-Bus wurde ein Baustein mit der Adresse 156 angeschlossen. Das folgende Programm liefert drei (und nicht wie vielleicht erwartet zwei) Adressen. Wie lässt sich das erklären?

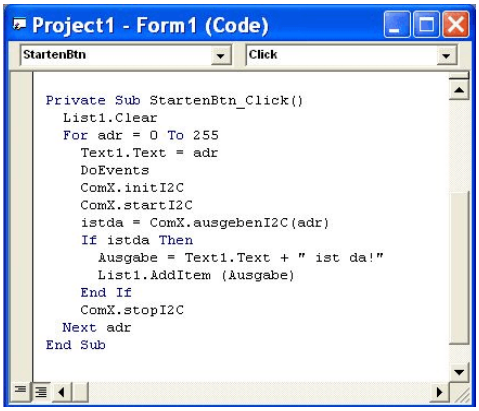


**Abb. 10**

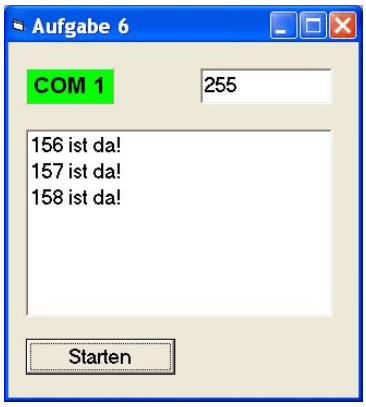


**Abb. 11**

Notiz:

## 4.3 Der Temperatursensor LM 75

Mit der Temperatursensor-Platine (Abb. 12) 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.

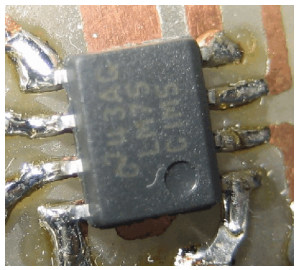

**Abb. 12**

Der benutzte Baustein LM 75 besitzt zwei Datenregister, die vom Master ausgelesen werden können. Mit dem Testprogramm I2CTest2 sieht das z. B. so aus:

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

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.

# 5. Aufgaben zum Messen

Von den folgenden Aufgaben sollten Sie auf jeden Fall 5.1, 5.2, 5.3 und eine weitere bearbeiten. Von den restlichen Aufgaben können Sie je nach Zeit eine oder mehrere bearbeiten; Sie können aber auch Ihren eigenen Ideen freien Lauf lassen!

## 5.1 Aufgabe mit Testprogramm und Universalplatine

Verbinden Sie die Universalplatine über die Interfaceplatine mit dem COM-Anschluss ihres Rechners. Starten Sie das Testprogramm I2CTest2.exe aus dem Verzeichnis Testprogramme der I²C-CD. Stellen Sie ggf. die COM-Schnittstelle ein. Die COM-Anzeige (s. Abb. 8) sollte grün hinterlegt sein und es sollten alle LEDs der Platine leuchten.

Sollten nicht alle LEDs leuchten, muss vor dem Lesevorgang zunächst der Wert 255 in das Register des PCF-Bausteins geschrieben werden. Achten Sie dabei auf die korrekte Adressierung!

Verbinden Sie wie in Abb. 9 angedeutet einen der mittleren Pins des Pfostensteckers mit dem Masse-Anschluss. Die entsprechende LED erlischt dann. Führen Sie einen oder mehrere Read-Vorgänge durch wie in 4.2 beschrieben.

Benutzen Sie auch die Eingabe-Platine, die Sie leihweise zur Verfügung gestellt bekommen.

## 5.2 Aufgabe mit Testprogramm und LM 75

Tauschen Sie nun die Universalplatine gegen die Temperatursensorplatine aus. Führen Sie wieder Read-Vorgänge durch. Achten Sie dabei auf das (Kein-)Acknowledge. Was geschieht, wenn man es vergisst? Erwärmen Sie auch den Sensor und führen Sie dabei fortlaufend Messungen durch.

*Jetzt wollen wir selbst programmieren! Benutzen Sie VB6 (auf der CD) oder eine andere Entwicklungsumgebung, welche mit ActiveX-Komponenten zusammenarbeiten können. Die konkreten Programmierhilfen beziehen sich immer auf VB6.*

## 5.3 Einzelmessungen mit VB (Universalplatine oder LM 75)

(Lösungs-Dateien in: source/I2C-Lesen)

Schreiben Sie ein Programm, welches nach Drücken einer Schaltfläche einen Messwert anzeigt. (Beim Temperatursensor soll nur das erste Byte berücksichtigt werden.) Benutzen Sie das Vorlage-Projekt aus dem Verzeichnis source/vorlage. Es ähnelt dem Projekt aus Abb. 6.

*Hinweis*: Schließen Sie ggf. zuerst VisualBasic. Kopieren Sie dann alle Dateien des Vorlage-Projekts in ein neues Verzeichnis; arbeiten Sie dann mit diesem neuen Projekt, indem Sie das Vorlage-Projekt in dem neuen Verzeichnis (doppelt) anklicken. Beim Abspeichern können

Sie dann neue Namen vergeben. (Auf diese Weise vermeiden Sie das Vermischen von unterschiedlichen Projekten!)

***Weitere Aufgaben...***

### 5.4 Dauermessungen oder Messprotokoll (Timer)

(Lösungs-Dateien in: source/Lösungen/Aufgabe 8)

Nun soll das Programm aus 5.3 fortwährend Temperatur-Messungen durchführen. Benutzen Sie dazu entweder

- eine Schleife und die COMX.delay-Methode

oder

- eine Timerkomponente.

Achten Sie darauf, dass die Messung jederzeit durch eine weitere Schaltfläche abgebrochen werden kann.

Beobachten Sie die Anzeige, wenn der Baustein erwärmt wird. Was geschieht, wenn die Sensorplatine entfernt wird, während das Programm läuft?

### 5.5 Temperaturmessung mit Nachkommastelle

(Lösungs-Dateien in: source/Lösungen/Aufgabe9)

Ergänzen Sie Ihr Programm aus 5.3 oder 5.4 so, dass zusätzlich eine Nachkommastelle angezeigt wird.

### 5.6 Liste aller Slaves

(Lösungs-Dateien in: source/Lösungen/Aufgabe7)

Ähnlich wie bei Aufgabe 6 von Kapitel 3 sollen sämtliche Slaves angezeigt werden. Diesmal sollen aber alle Typen (sowohl Write- als auch Read-Typen) angezeigt werden.

### 5.7 Messung mit LM 75 und Anzeige über Universalplatine

Die Messergebnisse aus Aufgabe 5.4 sollen nun über die Universalplatine ausgegeben werden.

# 6. Die 7-Segment-Anzeige

Wir benutzen zur Wiedergabe von Klartext eine Doppel-7-Segment-Anzeige, mit der Ziffern und diverse andere Zeichen dargestellt werden können. Sie wird fertig aufgebaut geliefert und kann mit dem rückseitig angebrachten Stecker direkt an die 2x9-polige Buchsenleiste der Universalplatine angesteckt werden. Eine weitere Beschaltung ist nicht notwendig.

Die Doppel-7-Segment-Anzeige besitzt insgesamt 14 Segmente; diese lassen sich natürlich nicht einzeln über die 8 Bit des PCF 8574 ansteuern. Deswegen benutzen wir hier folgendes Verfahren: ***Nacheinander*** werden das linke Zeichen, dann das rechte, dann wieder das linke, dann das rechte und so weiter zur Anzeige gebracht. Geschieht dies schnell genug, sieht das Auge beide angezeigten Zeichen gleichzeitig. Für jedes Zeichen werden 7 Bit benötigt; zum Umschalten zwischen den beiden Zeichen wird ein weiteres Bit eingesetzt. Dieses Verfahren wird auch als ***Multiplexen*** bezeichnet.

## 6.1 Hardware

Dem Schaltbild der Universalplatine kann man entnehmen, dass die 2x9-polige Buchsenleiste – dort mit „St2“ bezeichnet – von den Ausgängen des Treiberbausteins ULN 2803 bedient wird. Bei einer logischen 1 am Ausgang des PCF 8574 ist der Ausgang des ULN 2803 mit Masse verbunden, die zugehörige LED leuchtet.

Die hier verwendete Doppelanzeige besteht aus zwei getrennten 7-Segment-Anzeigen; bei jeder 7-Segment-Anzeige sind die Anoden der einzelnen LEDs zusammengeschaltet, man spricht deshalb von einem Baustein mit gemeinsamer Anode. Verbindet man nun diese gemeinsame Anoden mit dem Pluspol und die einzelnen Kathoden mit den Ausgängen des ULN 2803, so kann durch eine entsprechende Kombination von Zweierpotenzen jedes beliebige Muster dargestellt werden.

Eins der acht Bits ist bisher unbenutzt, es kann deshalb für die Umschaltung der beiden Anzeigen verwendet werden.

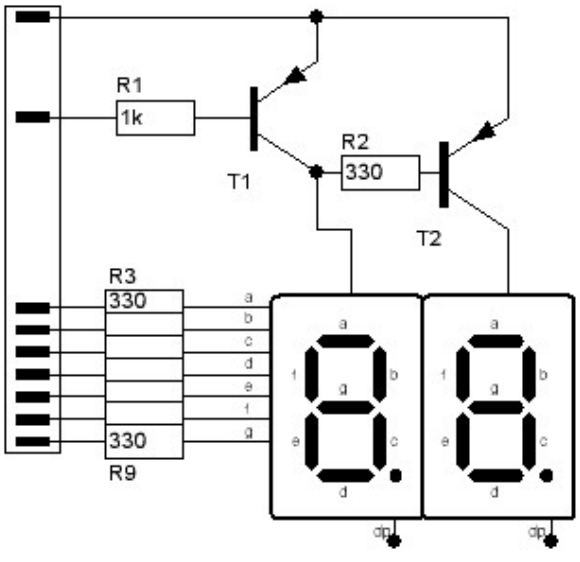


**Abb. 13**: Schaltbild der Doppel-7-Segmentanzeige

**Abb. 14**

Unten links in Abb. 14 sind die beiden Transistoren zu erkennen, die für die Umschaltung der Anzeigen sorgen. Sie legen den Pluspol an die Anoden und werden selbst vom niederwertigsten Bit ($2^0$) geschaltet.

Damit die Zuordnung einigermaßen stringent ist, wird Bit 2 (also $2^1$) dem Segment a, $2^2$ dem Segment b, $2^3$ dem Segment c usw. zugeordnet, Segment g entspricht also $2^7$. Damit kann man den Wert für jedes Zeichen leicht ausrechnen.

Setzt man das erste Bit (also $2^0$), dann ist der Dezimalwert um eins höher und das Zeichen wird an der Zehnerstelle (also vorn) angezeigt. Auch das kann man sich leicht merken: der Wert für die Anzeige hinten ist immer geradzahlig, für die vordere immer ungeradzahlig.

*Beispiel 1*: Die Ziffer 7 soll an der Einerstelle angezeigt werden. Dazu müssen die Segmente a, b und c leuchten; deren Bits müssen also high sein. Dementsprechend ergibt sich als Ausgabewert die Zahl 14.

| Stelle | **a** | **b** | **c** | (d) | (e) | (f) | (g) | Erg. |
|---|---|---|---|---|---|---|---|---|
| $2^0$ | $2^1$ | $2^2$ | $2^3$ | $2^4$ | $2^5$ | $2^6$ | $2^7$ | |
| 0 | 2 | 4 | 8 | 0 | 0 | 0 | 0 | 14 |

*Beispiel 2*: Die Ziffer 3 soll an der Zehnerstelle angezeigt werden. Dazu müssen die Segmente a, b, c, d und g leuchten; deren Bits müssen also high sein. Dementsprechend ergibt sich als Ausgabewert die Zahl 159.

| Stelle | **a** | **b** | **c** | **d** | (e) | (f) | **g** | Erg. |
|---|---|---|---|---|---|---|---|---|
| $2^0$ | $2^1$ | $2^2$ | $2^3$ | $2^4$ | $2^5$ | $2^6$ | $2^7$ | |
| 1 | 2 | 4 | 8 | 16 | 0 | 0 | 128 | 159 |

Will man die Zahl 37 anzeigen lassen, so muss man in rascher Folge abwechselnd immer wieder die Zahlen 14 und 159 an den PCF-Baustein ausgeben.

# 7. Aufgaben

## 7.1 Eine Zahl mit Doppel-7-Segment-Anzeige ausgeben

(Lösungs-Dateien in: source/Lösungen/Aufgabe10)

Benutzen Sie wieder das Vorlage-Projekt; kopieren Sie es dazu in ein neues Verzeichnis.

7.1.1 Wenn Sie die Schaltfläche “Senden” betätigen, soll in der Einerstelle die Ziffer “4" angezeigt werden.

7.1.2 Wenn Sie die Schaltfläche “Senden” betätigen, soll in der Zweierstelle die Ziffer “5" angezeigt werden.

7.1.3 Erstellen Sie eine Zuordnungstabelle Ziffer → Code:

| Ziffer | Code (Zehner) | Code (Einer) |
|---|---|---|
| 0 | | |
| 1 | | |
| 2 | | |
| 3 | 159 | 158 |
| 4 | 205 | 204 |
| 5 | | |
| 6 | | |
| 7 | | |
| 8 | | |
| 9 | | |

7.1.4 Zahl 34 zur Anzeige bringen. Benutzen Sie der Einfachheit halber eine Zählschleife für das Multiplexen (z. B. eine Schleife von 1 bis 50 durchlaufen lassen).

*Für Programmierprofis*:

7.1.5 Wenn die Schaltfläche “Senden” betätigt wird, dann soll nun die Zahl im Wert-Feld angezeigt werden.

Tipps:

1. Die Code-Tabelle für die Einer lässt sich mit einem Array erfassen:

Deklaration:

Dim anzeige(9) legt ein Array anzeige mit den Indizes 0 bis 9 an.

Initialisierung:

```
anzeige(0) = ...
anzeige(1) = ...
...
```

2. Zur Isolierung der Einer und Zehner aus der darzustellenden Zahl kann man die Ganzzahldivision (\) und die Restoperation (mod) benutzen.

## 7.2 Zählwerk mit Doppel-7-Segment-Anzeige

(Lösungs-Dateien in: source/Lösungen/Aufgabe11)

Schreiben Sie ein Programm, welches die Zahlen von 0 bis 99 hintereinander auf der Doppel-7-Segment-Anzeige anzeigt.

## 7.3 Temperaturanzeige mit Doppel-7-Segment-Anzeige

Schließen Sie nun zusätzlich den Temperatursensor an die Interface-Platine an. Schreiben Sie ein Programm, welches nach Betätigen der Schaltfläche “Temperaturanzeige” die aktuelle Temperatur auf der Doppel-7-Segment-Anzeige anzeigt.

## 7.4 Sekundenzähler mit Doppel-7-Segment-Anzeige

(Lösungs-Dateien in: source/Lösungen/Aufgabe13)

Die Doppel-7-Segment-Anzeige soll - nach Betätigen des Start-Knopfes - die Sekunden anzeigen. Nach der Zahl 59 soll jeweils wieder die Zahl 0 angezeigt werden. Abbruch mit Stopp-Schaltfläche. Versuchen Sie eine möglichst hohe Ganggenauigkeit zu erzielen.