Messen und Steuern mit dem PDA

Von Wolfgang Back

Ich muss es am Anfang sagen: nicht jeder Besitzer eines PDA's wird an diesem Bastelvorschlag Freude haben. Bei den meisten älteren PDA's hat man in punkto serieller Schnittstelle ziemlich geschludert und nur das Nötigste eingebaut: Daten rein, Daten raus.
Dass man eventuell ein Modem anschließen möchte, das Steuersignale benötigt – oder dass man damit gar steuern möchte, war nicht in der Entwicklung vorgesehen.

Es ist mir bisher nur der Compaq 3850 bekannt, der die RS232 – Spezifikation komplett erfüllt. Der Vorgänger 3630 hat die Spezifikationen nur zur Hälfte erfüllt. Mein Cassiopeia 125 gehört zu den PDA's, die keine besonderen Steuerleitungen aufweisen. Der HP - Jornada hat ebenfalls ein eingeschränktes RS232 – Verhalten.

Nachtrag: Der LOOX hat offensichtlich auch eine voll ausgerüstete Schnittstelle. Die Dokumentation der Schnittstelle ist eine einem anderen Artikel auf diesen Seiten niedergelegt.
Der LOOX und die serielle Schnittstelle.

Natürlich gelten die Einschränkungen nicht für die seriellen Schnittstellen im Laptop oder im Desktop. Hier sind normalerweise die Steuerleitungen ansprechbar.

Aber warum dies alles?

Die serielle Schnittstelle, die einmal sehr wichtig war, wird immer mehr zum Auslaufmodell. Die Modems, die hierüber bedient wurden, sterben aus: Karten oder Onboardlösungen ersetzen diese Geräte. Früher waren sogar zwei serielle Ports üblich, da einer von der Maus belegt wurde.

Heutzutage aber hat (fast) jeder Computer noch eine serielle Schnittstelle, die oftmals ungenutzt ihr Dasein fristet. Also könnte sie neue Aufgaben bekommen, wenn man über diese Schnittstelle neue Aufgaben definieren würde.

Beim PDA ist es wiederum anders. Er hat zur Außenwelt nicht allzu viele Möglichkeiten, z.B. finden wir hier keine Standard – Parallel -Schnittstelle. Natürlich gibt es den Slot für die PC-Karten; doch wer kann und will hier eingreifen, um Steuerbefehle abzusetzen. Neben den Kopfhörerausgängen und der seriellen Schnittstelle findet man vielleicht noch den USB - Ausgang vor.

Mit den PDA's halten wir aber komplette Computer in der Hand. Sie haben alle Funktionen eines Computers – doch es fehlt die Möglichkeit der direkten Steuerung einer Umgebungselektronik.

Es soll hier nun ein Projekt beschrieben werden, das es ermöglicht, über die PDA – Serielle - Schnittstelle (Compaq 3850 ausprobiert) oder aber über die Serielle eines anderen Computers eine Steuerung über den sogenannten I2C – Bus zu realisieren.

Der I2C - Bus

Der I2C – Bus ist ein in die Jahre gekommener Bus, der aber heutzutage noch immer zum Einsatz kommt. Gutes muss nicht immer ausgetauscht werden! Dieser Bus wurde damals von Philips entwickelt, um in den HiFi - Geräten möglichst einfach Funktionen realisieren zu können, z.B. die Lautstärkeänderung oder die Wahl eines Senders.

Alle Busse müssen in ihrem Verhalten genau definiert sein. Der I2C – Bus ist ein sogenannter Zweidrahtbus – natürlich sind es drei Drähte, da die Masse ja noch hinzukommt.
Um die Information abzurunden: es gibt auch einen Eindrahtbus, einen Dreidrahtbus usw.

Beim Eindrahtbus muss die gesamte Information über eine Leitung (natürlich zwei Leitungen, denn die Masse kommt hinzu) abgearbeitet werden. Dieses bedeutet aber, dass man eine große Disziplin in punkto Zeitverhalten voraussetzen muss. Um Daten hin und herzuschieben muss man genaue zeitliche Protokolle vereinbaren, damit dieses funktioniert; die Zeitschlitze sind zu gering, um es mit dem PDA per Visual Basic zu versuchen.

Beim I2C – Bus haben wir es einfacher. Da hier zwei Leitungen zur Steuerung des Busses vorhanden sind, kann das Protokoll in punkto Zeitverhalten entspannt erfüllt werden. Um es auf die Spitze zu treiben: der I2C – Bus funktioniert noch, wenn heute ein Befehl, morgen der nächste, in einem Monat der nächste .... geschickt wird.

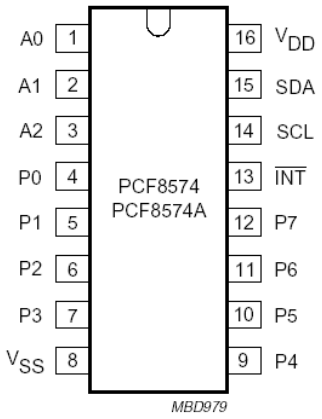
Dieses ist ein Riesenvorteil des I2C – Busses. Nachteile hat er auch: er ist nicht der ‚superschnellste Bus'. Für unsere Steuerungsaufgaben aber reicht es vollkommen aus.

Man benötigt spezielle I2C – Chips.

Um den I2C – Bus einsetzen zu können, muss man natürlich spezielle Chips einsetzen, die das Protokoll des Busses verarbeiten können. Es gibt hier verschiedene Chips vom Analog/Digitalwandler bis zum Uhrenchip. Wir wollen hier einen Portexpanderchip einsetzen, der 8 Ausgänge oder Eingänge zur Verfügung stellt.

Wir wollen ja mit der eingeschränkten Möglichkeit der wenigen Steuerleitungen im PDA viele Ausgänge bedienen.

Also: wir benötigen einen Chip, der dieses ermöglicht. Der PCF8574 ist ein solcher Chip, der eine Porterweiterung: aus 1 mach 8 zulässt.



Das Anschlussbild für diesen Chip sieht so aus, wie oben gezeigt. Zunächst sagt einem die Pinbezeichnung nicht viel, wenn man noch nie mit diesem Chip zu tun hatte.

Wir sprachen eben davon, dass wir aus ‚**eins wird acht**' machen wollen. Der oben abgebildete Chip hat die Ausgänge P0 bis P7. Das macht nach Adam Riese 8 Ausgänge.

Dieses passt zu unserer Vorhersage.

Doch wie kommen wir dazu, die einzelnen Ausgänge (oder Eingänge) anzusprechen? Der I2C – Bus kann dies eben , indem er nur zwei Steuerleitungen benutzt.

Zunächst wird durch das Aussenden eines Startsignals die Übertragung begonnen:

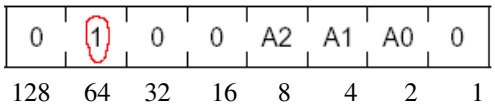
Start heißt: die Data-Leitung (SDA) und die Clock (SCL) -Leitung wechseln im richtigen Rhythmus (s. späteres Programm).

Für den I2C - Chip oder für die I2C - Chips auf der Platine bedeutet diese Startsequenz: Achtung alle I2C – Chips: : „Ohren auf, hier kommt etwas Neues“.

Danach kommt das Übermitteln einer Adresse.

Die Sache mit der Adresse muss ein wenig erklärt werden:
Jeder I2C-Chip erhält werksseitig eine Grund- oder Systemadresse. Alle Chips mit dieser Funktion haben die gleiche Systemadresse. Bei dem Chip, der oben abgebildet ist, dem Input-Output – Porterweiterer PCF 8574, beträgt die Systemadresse 64 dezimal. Kauft man also einen solchen Chip, so hat er diese Ansprechadresse.

Das Bild zeigt es. Das zweite Bit von links (rot umrandet) ist 1 und hat die Wertigkeit 64.



So wie es bisher beschrieben wurde, würde dies bedeuten, dass man einen Chip nur einmal ansprechen kann. Dieses ist falsch. Man kann 8 dieser Chips ansprechen, da die Möglichkeit der Adressierung besteht. Am obigen Schaltbild sieht man: es gibt die Eingänge A0, A1,A2.

Je nach Belegung der Adressen können die Chips adressiert werden.
Beispiel 1: alle Eingänge A0, A1, A2 sind auf Masse gelötet: die Adresse ist Systemadresse + 0+ 0 + 0 = 64.

Es können insgesamt 8 verschiedene Adressen erzeugt werden. Aus drei Bit (2^3) können eben 8 Adressen angesprochen werden.

Alle Möglichkeiten hier als Beispiel:

Nr.	Systemadresse = 64	A2=8	A1=4	A0=2	Chipadresse
1	64	0	0	0	64
2	64	0	0	2	66
3	64	0	4	0	68
4	64	0	4	2	70
5	64	8	0	0	72
6	64	8	0	2	74
7	64	8	4	0	76

Nach dem Aussenden des Startsignals kommt also jetzt das Versenden der Adresse des gewünschten Chips. Bleiben wir im ersten Beispiel bei der Wahl der Systemadresse 64 (A0,A1,A2 sind alle mit Masse verbunden und daher 0). Unser zu sendendes Byte sieht also jetzt so aus: 0 1 0 0 A2=0 A1=0 A0=0. Wenn wir genau nachzählen, so stellen wir fest, dass wir erst 7 Bits in Verwendung haben, ein Byte sich aber auch 8 Bits zusammensetzt.

Das letzte fehlende (8.) Bit hat beim I2CBus eine wichtige Funktion. Hier wird entschieden, ob man im nächsten Arbeitsgang schreiben oder lesen möchte. Will man einen Schreibvorgang auslösen, so wird das 8. Bit auf 0 (Masse) gesetzt, will man dagegen lesen, so wird hier eine 1 gesetzt.

Zunächst wollen wir einen Ausgang des PCF8574 – Chip so schalten, dass z.B, ein Relais oder eine Leuchtdiode ein- oder ausgeschaltet wird. Dieses entspricht also einem Schreibvorgang und das

letzte Bit wird auf 0 gesetzt.
Jetzt ist das Adressbyte mit 8 Bits aufgefüllt. Es sieht so aus: 0 1=Systemadresse 0 0 A2=0 A1=0 A0=0 0 für Schreiben.

Dieses Adressbyte wird jetzt dem Chip mitgeteilt. Man beginnt mit dem höchstwertigen Bit mit der Wertigkeit 128, dann Wertigkeit 64 usw.
In Basic sieht das dann z.B. so aus:

```
For i = 7 to 0 step –1
        If (adressbyte and 2 ^ I) = 0 then call data(0) else call data(1)
Next i
```

Wenn wir dieses auflösen, so wird das gewünschte Datentelegramm ausgesendet: 01000000. Später sehen wir genauer, was es mit data(0) und data(1) auf sich hat.

Funktionieren würde das jedoch noch nicht, wenn wir dieses Telegramm einfach so auf den Bus geben würden. Es fehlt noch das Clocksignal. Nach jedem ausgesendetem Bit erwartet der Chip ein Clocksignal, damit er die Information in den richtigen Zeitrahmen setzen kann.

Nach der Startbedingung Call clock(1): Call data(1): Call data(0): Call clock(0) ist das Clocksignal 0 vom Chip als Datenbit erkannt. Das zweite Bit (in unserem Falle die 1 von der 64) wird erkannt, wenn wieder ein Clockwechsel stattgefunden hat. Das nächste Bit ..... usw.

Der Programmteil ändert sich daher folgendermaßen, wir nennen ihn jetzt auch schon einmal PUTBYTE, da wir etwas senden wollen:

```
Adressbyte = 64
For i = 7 to 0 step –1
        If (adressbyte and 2 ^ I) = 0 then call data(0) else call data(1)
        Call clock(1) : call clock (0)
Next i
```

Mit dem Startsignal und dieser bitweisen Übertragung des Adressbytes haben wir folgendes bewirkt: alle Chips am Bus sind aufgewacht. Der Chip mit der Systemadresse 64 hat erkannt, dass er angesprochen wurde und zwar soll als nächste Aktion eine Information geschrieben (nicht gelesen) werden.

Der Chip zeigt nun, dass er die Aktion verstanden hat und zieht seine Dataleitung auf Lowsignal. Dieses kann man vom Programm abfragen: man wartet so lange, bis der Lowzustand eingetreten ist, bevor man weitermacht. In der Fachsprache heisst dies: man fragt das Acknowledge ab, die Zustimmung des Chips. Wir nennen diese Prozedur deshalb GETACK, da wir das Acknowledge abfragen.

```
Call data(1)
      Do While Comm1.CTSHolding = True
      Loop
   Call clock(1)
   Call clock(0)
```

Data wird zunächst auf Highpegel gesetzt. Mit der Do---while – Schleife wird die Veränderung des Pegels abgefragt. Danach erfolgt wieder ein Clockimpuls und die Funktion GETACK ist beendet. Warum dort Comm1.CTSHolding steht, wird später noch erklärt.

Noch einmal zusammengefasst, wo wir jetzt sind.
Startsignal START wurde gesendet.

Chipadresse 64 mit (A2,A1,A0=) und schreiben =0 wurde gesendet.
Das Acknowledge wurde mit GETACK eingeholt.

Jetzt sind wir soweit, dass wir unsere gewünschten Daten übertragen können, denn jetzt ist nur noch der Chip mit der Systemadresse 64 aktiv. Beispielhaft wollen wir den Ausgang P0 und P1 schalten. Dieses entspricht dem Datenbyte: 00000011. Die Funktion PUTBYTE haben wir oben schon gesehen.

Ist das Datenbyte versandt, so kommt jetzt wieder die Frage, ob es vom Chip verstanden wurde. Also GETACK.

Danach ist die Operation zuerst einmal beendet. Wir senden das Stoppsignal:

Call data(0): Call clock(1): Call data(1)

An diesen Ausführungen kann man erkennen, dass der I2C-Bus ein relativ langsamer aber sehr toleranter Bus ist, da er keine Anforderungen an zeitliche Raster stellt. Gehen wir noch einmal schnell einen Datenaustausch (Schreiben) durch:

START
PUTBYTE (Adresse)
GETACK
PUTBYTE (Daten)
GETACK
STOP

Dieses sollte man sich einprägen, denn die nächste Aktion: das Lesen wird damit auch verständlich.

Wir möchten jetzt wissen, welche Werte die Aus- Eingänge des Chips mit der Systemadresse 64 haben. Es kann ja sein, dass hier z.B. Rollladensteuerungen angeschlossen sind, die den Stand offen/geschlossen melden. Diesen Zustand möchten wir abfragen.

Es beginnt fast wie gehabt. Nur jetzt wird dem Chip schon mit der PUTBYTE-Aktion mitgeteilt, dass wir Lesen möchten: das letzte Bit wechselt von 0 auf 1. Unser Adressbyte sieht jetzt so aus: 01000001.
START
PUTBYTE
GETACK

Da wir jetzt als zweite Aktion Lesen möchten, müssen wir eine neue Funktion einführen, die hier GETBYTE heißen soll. Unter GETBYTE versteckt sich etwas ähnliches wie unter PUTBYTE – nur anders herum.

```
i2caus = 0
   For i = 7 To 0 Step -1
   If Comm1.CTSHolding = False Then i2caus = i2caus + 2 ^ i
   Call clock(1)
   Call clock(0)
   Next
```

Hier erscheint eine neue Hilfsvariable i2caus. Sie erhält den gelesenen Bytewert. Auch hier erkennen wir, dass mit der For ... Next – Schleife der Inhalt bitweise ausgetaktet wird. Es wird nachgesehen, welches Bit ist low und welches Bit ist high – getrennt jeweils von einem Clockwechsel.

Ist i2caus = 255 dann sind alle Ausgänge high – quasi alles aus. Ist i2caus=3 dann ist der letzte und

vorletzte (P7 und P8) Ausgang low –also aktiv.

Wenn wir dieses gelesen haben, so können wir dem Chip mitteilen, dass wir es verstanden haben und schicken ihm ein GIVEACK, das ähnlich ist wie das GETACK – nur anders herum und nicht so kompliziert mit der Wartefunktion.

Call data(0): Call clock(1): Call clock(0): Call data(1)

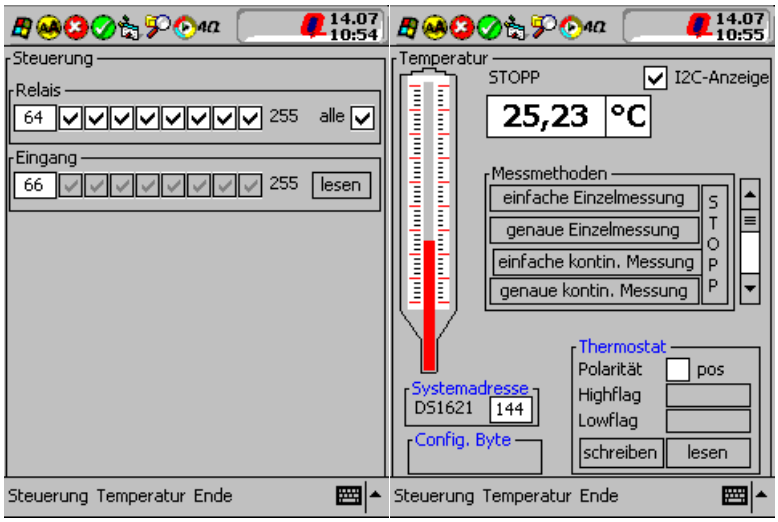
Danach ist die Prozedur abgeschlossen und es erfolgt das Stoppsignal. Noch einmal das einfache Schema:

START
PUTBYTE (Lesen)
GETACK
GETBYTE (Daten)
GIVEACK
STOP

Der I2C – Bus ist damit fast erklärt. Es fehlt noch eine Funktion GIVENOACK. Es ist das Gegenteil von GIVEACK. GIVENOACK benötigt man, wenn man andere I2C – Chips einsetzt und dort Sequenzen von Abfragen einsetzt. Wenn man zum Beispiel Temperaturen misst mit dem Chip DS1621 oder dem LM75 und dort mehrere Register nacheinander in einem Befehlszyklus abfragt, so folgt am Schluß ein GIVENOACK:STOP.

Der I2C – Bus ist damit vom Prinzip her erkärt. Das Programm aus der Sendung veröffentliche ich daher hier als Sourcecode :

Zunächst zwei Screenshots.



Option Explicit
Declare Sub Sleep Lib "Coredll" (ByVal dwMilliseconds As Long)

Dim tempi As Variant
Dim i2cein, i2caus, adresse, schreibe, i As Integer
Dim log_start, kon As Boolean
Dim checkarray(), textarray()

Private Sub Check1_Click()
Call relais_schalten1(CInt(txt_systemadresse1))
End Sub

```
Private Sub Check10_Click()
Call relais_schalten2(CInt(Txt_systemadresse2))
End Sub
Private Sub Check11_Click()
Call relais_schalten2(CInt(Txt_systemadresse2))
End Sub

Private Sub Check12_Click()
Call relais_schalten2(CInt(Txt_systemadresse2))
End Sub

Private Sub Check13_Click()
Call relais_schalten2(CInt(Txt_systemadresse2))
End Sub

Private Sub Check14_Click()
Call relais_schalten2(CInt(Txt_systemadresse2))
End Sub

Private Sub Check15_Click()
Call relais_schalten2(CInt(Txt_systemadresse2))
End Sub

Private Sub Check16_Click()
Call relais_schalten2(CInt(Txt_systemadresse2))
End Sub

Private Sub Check17_Click()
Call relais_schalten3(CInt(txt_systemadresse3))
End Sub

Private Sub Check18_Click()
Call relais_schalten3(CInt(txt_systemadresse3))
End Sub

Private Sub Check19_Click()
Call relais_schalten3(CInt(txt_systemadresse3))
End Sub

Private Sub Check2_Click()
Call relais_schalten1(CInt(txt_systemadresse1))
End Sub

Private Sub Check20_Click()
Call relais_schalten3(CInt(txt_systemadresse3))
End Sub

Private Sub Check21_Click()
Call relais_schalten3(CInt(txt_systemadresse3))
End Sub

Private Sub Check22_Click()
Call relais_schalten3(CInt(txt_systemadresse3))
End Sub
```

```
Private Sub Check23_Click()
Call relais_schalten3(CInt(txt_systemadresse3))
End Sub

Private Sub Check24_Click()
Call relais_schalten3(CInt(txt_systemadresse3))
End Sub

Private Sub Check25_Click()
Call relais_schalten4(CInt(txt_systemadresse4))
End Sub

Private Sub Check26_Click()
Call relais_schalten4(CInt(txt_systemadresse4))
End Sub

Private Sub Check27_Click()
Call relais_schalten4(CInt(txt_systemadresse4))
End Sub

Private Sub Check28_Click()
Call relais_schalten4(CInt(txt_systemadresse4))
End Sub

Private Sub Check29_Click()
Call relais_schalten4(CInt(txt_systemadresse4))
End Sub

Private Sub Check3_Click()
Call relais_schalten1(CInt(txt_systemadresse1))
End Sub

Private Sub Check30_Click()
Call relais_schalten4(CInt(txt_systemadresse4))
End Sub

Private Sub Check31_Click()
Call relais_schalten4(CInt(txt_systemadresse4))
End Sub

Private Sub Check32_Click()
Call relais_schalten4(CInt(txt_systemadresse4))
End Sub

Private Sub Check4_Click()
Call relais_schalten1(CInt(txt_systemadresse1))
End Sub

Private Sub Check5_Click()
Call relais_schalten1(CInt(txt_systemadresse1))
End Sub

Private Sub Check6_Click()
Call relais_schalten1(CInt(txt_systemadresse1))
```

```
End Sub

Private Sub Check7_Click()
Call relais_schalten1(CInt(txt_systemadresse1))
End Sub

Private Sub Check8_Click()
Call relais_schalten1(CInt(txt_systemadresse1))
End Sub

Private Sub Check9_Click()
Call relais_schalten1(CInt(Txt_systemadresse2))
End Sub

Private Sub Comm1_OnComm()

End Sub

Private Sub Command1_Click()
If Command1.Caption = "alle aus" Then

   Check1.Value = 0
   Check2.Value = 0
   Check3.Value = 0
   Check4.Value = 0
   Check5.Value = 0
   Check6.Value = 0
   Check7.Value = 0
   Check8.Value = 0

   Command1.Caption = "alle ein"
Else
   Check1.Value = 1
   Check2.Value = 1
   Check3.Value = 1
   Check4.Value = 1
   Check5.Value = 1
   Check6.Value = 1
   Check7.Value = 1
   Check8.Value = 1
   Command1.Caption = "alle aus"
End If
End Sub

Private Sub Command2_Click()
kon = True
Call einfache_temperaturmessung
End Sub
Private Sub Command3_Click()
kon = False
Call genaue_temperaturmessung
End Sub

Private Sub Command4_Click()
Timer1.Enabled = True
```

```
End Sub

Private Sub Command5_Click()
If Command5.Caption = "alle aus" Then

   Check9.Value = 0
   Check10.Value = 0
   Check11.Value = 0
   Check12.Value = 0
   Check13.Value = 0
   Check14.Value = 0
   Check15.Value = 0
   Check16.Value = 0

   Command5.Caption = "alle ein"
Else
   Check9.Value = 1
   Check10.Value = 1
   Check11.Value = 1
   Check12.Value = 1
   Check13.Value = 1
   Check14.Value = 1
   Check15.Value = 1
   Check16.Value = 1
   Command5.Caption = "alle aus"
End If
End Sub

Private Sub Command6_Click()
If Command6.Caption = "alle aus" Then

   Check17.Value = 0
   Check18.Value = 0
   Check19.Value = 0
   Check20.Value = 0
   Check21.Value = 0
   Check22.Value = 0
   Check23.Value = 0
   Check24.Value = 0

   Command6.Caption = "alle ein"
Else
   Check17.Value = 1
   Check18.Value = 1
   Check19.Value = 1
   Check20.Value = 1
   Check21.Value = 1
   Check22.Value = 1
   Check23.Value = 1
   Check24.Value = 1
   Command6.Caption = "alle aus"
End If
End Sub

Private Sub Command7_Click()
```

```
If Command7.Caption = "alle aus" Then

   Check25.Value = 0
   Check26.Value = 0
   Check27.Value = 0
   Check28.Value = 0
   Check29.Value = 0
   Check30.Value = 0
   Check31.Value = 0
   Check32.Value = 0

   Command7.Caption = "alle ein"
Else
   Check25.Value = 1
   Check26.Value = 1
   Check27.Value = 1
   Check28.Value = 1
   Check29.Value = 1
   Check30.Value = 1
   Check31.Value = 1
   Check32.Value = 1
   Command7.Caption = "alle aus"
End If
End Sub

Private Sub Command8_Click()
adresse = CInt(txt_systemadresse4)
Call START: schreibe = adresse + 1: Call PUTBYTE: Call GETACK
Call GETBYTE: Call GIVEACK: Call STOPP
txt_bytewert4 = 255 - i2caus
End Sub

Private Sub Command9_Click()
Timer1.Enabled = False
End Sub

Private Sub Form_Activate()
' zeichnet das Thermometer
'HScroll1.Value = 10
pb1.BorderStyle = 0
pb1.ScaleMode = picPixels
pb1.Width = 37 * Screen.TwipsPerPixelX
pb1.Height = 193 * Screen.TwipsPerPixelY
pb1.Left = 90
Frame_temperatur.BackColor = RGB(192, 192, 192)
Frame_logger.BackColor = RGB(192, 192, 192)


pb1.BackColor = RGB(192, 192, 192)

pb1.DrawLine 5, 6, 31, 152, vbWhite, True, True ' zeichnet weissen Kasten
pb1.DrawLine 15, 10, 21, 152, RGB(192, 192, 192), True, True ' zeichnet inneren grauen Kasten
pb1.DrawLine 15, 148, 22, 190, vbRed, True, True ' zeichnet roten Balken

pb1.DrawLine 0, 5, 7, 5, vbBlack ' oben links
```

```
pb1.DrawLine 7, 5, 7, 3
pb1.DrawLine 7, 3, 14, 3
pb1.DrawLine 14, 3, 14, 1

pb1.DrawLine 14, 1, 22, 1

pb1.DrawLine 22, 1, 22, 3
pb1.DrawLine 22, 3, 28, 3
pb1.DrawLine 28, 3, 28, 5
pb1.DrawLine 28, 5, 35, 5

pb1.DrawLine 0, 6, 0, 152, vbBlack ' linke Seite runter
pb1.DrawLine 35, 6, 35, 152, vbBlack ' rechte Seite runter

pb1.DrawLine 0, 152, 14, 172 ' linke Seite schräg
pb1.DrawLine 35, 152, 22, 172 ' rechte Seite schräg

pb1.DrawLine 13, 172, 13, 191 ' Spitze links

pb1.DrawLine 23, 172, 23, 191 ', vbBlack, True, True ' Spitze rechts

pb1.DrawLine 13, 191, 23, 192, vbBlack, True, True ' Spitze unten

For i = 0 To 136 Step 3
If i Mod 5 = 0 Then pb1.DrawLine 6, i + 12, 13, i + 12, vbRed
If i Mod 5 <> 0 Then pb1.DrawLine 9, i + 12, 13, i + 12, vbBlack
If i Mod 5 = 0 Then pb1.DrawLine 24, i + 12, 31, i + 12, vbRed
If i Mod 5 <> 0 Then pb1.DrawLine 24, i + 12, 27, i + 12, vbBlack
Next i

Comm1.PortOpen = True
End Sub

Private Sub Form_KeyUp(KeyCode As Integer, Shift As Integer)

End Sub

Private Sub Form_OKClick()
   App.End
End Sub
Private Sub Form_Load()
Me.Caption = "I2C - Steuerung"

ReDim checkarray(32)
ReDim textarray(9)

   Frame_steuerung.Width = 3570
   Frame_steuerung.Left = 15
   Frame_steuerung.Top = 0
   Frame_steuerung.Height = 4000
   Frame_steuerung.Caption = "Steuerung"
   Frame_steuerung.BackColor = RGB(192, 192, 192)

      Set checkarray(1) = Check1
      Set checkarray(2) = Check2
```

```
    Set checkarray(3) = Check3
    Set checkarray(4) = Check4
    Set checkarray(5) = Check5
    Set checkarray(6) = Check6
    Set checkarray(7) = Check7
    Set checkarray(8) = Check8

    Set checkarray(9) = Check9
    Set checkarray(10) = Check10
    Set checkarray(11) = Check11
    Set checkarray(12) = Check12
    Set checkarray(13) = Check13
    Set checkarray(14) = Check14
    Set checkarray(15) = Check15
    Set checkarray(16) = Check16

    Set checkarray(17) = Check17
    Set checkarray(18) = Check18
    Set checkarray(19) = Check19
    Set checkarray(20) = Check20
    Set checkarray(21) = Check21
    Set checkarray(22) = Check22
    Set checkarray(23) = Check23
    Set checkarray(24) = Check24

    Set checkarray(25) = Check25
    Set checkarray(26) = Check26
    Set checkarray(27) = Check27
    Set checkarray(28) = Check28
    Set checkarray(29) = Check29
    Set checkarray(30) = Check30
    Set checkarray(31) = Check31
    Set checkarray(32) = Check32

    Set textarray(1) = txt_bytewert1
    Set textarray(2) = txt_bytewert2

Frame_steuerung.Left = 0
Frame_steuerung.Top = 0


Frame_temperatur.Left = 0
Frame_temperatur.Top = 0

Frame_logger.Left = 0
Frame_logger.Top = 0


Dim steuerung As MenuBarLib.MenuBarMenu
Dim Temperatur As MenuBarLib.MenuBarMenu
Dim ende As MenuBarLib.MenuBarMenu

Set steuerung = MenuBar1.Controls.AddMenu("Steuerung", "steuerung")
steuerung.Items.Add 1, "relais", "Relais"
steuerung.Items.Add 2, "eingang", "Eingang"
```

```
    Set Temperatur = MenuBar1.Controls.AddMenu("Temperatur", "Temperatur")
    Temperatur.Items.Add 1, "mnuEditSort", "Temperaturmessung"
    Temperatur.Items.Add 2, "mnuEditFind", "Datenlogger"

    Set ende = MenuBar1.Controls.AddMenu("Ende", "Ende")
    ende.Items.Add 1, "ende", "wirklich?"
End Sub
Private Sub relais_schalten1(adresswert)
    adresse = adresswert
    i2cein = 0
    For i = 1 To 8
        If checkarray(i).Value = 0 Then i2cein = i2cein + 2 ^ (i - 1)
    Next i
    txt_bytewert1 = i2cein
    Call SCHREIBEN
End Sub
Private Sub relais_schalten2(adresswert)
    adresse = adresswert
    i2cein = 0
    For i = 9 To 16
        If checkarray(i).Value = 0 Then i2cein = i2cein + 2 ^ (i - 9)
    Next i
    txt_bytewert2 = i2cein
    Call SCHREIBEN
End Sub
Private Sub relais_schalten3(adresswert)
    adresse = adresswert
    i2cein = 0
    For i = 17 To 24
        If checkarray(i).Value = 0 Then i2cein = i2cein + 2 ^ (i - 17)
    Next i
    Txt_bytewert3 = i2cein
    Call SCHREIBEN
End Sub
Private Sub relais_schalten4(adresswert)
    adresse = adresswert
    i2cein = 0
    For i = 25 To 32
        If checkarray(i).Value = 0 Then i2cein = i2cein + 2 ^ (i - 25)
    Next i
    txt_bytewert4 = i2cein
    Call SCHREIBEN
End Sub
' ***************** Anfang I2C - Bus - Routinen ****************************
Private Sub LESEN()
    Call START: schreibe = adresse + 1: Call PUTBYTE: Call GETACK
    Call GETBYTE: Call STOPP
End Sub
Private Sub SCHREIBEN()
    Call START: schreibe = adresse: Call PUTBYTE: Call GETACK
    schreibe = i2cein: Call PUTBYTE: Call GETACK: Call STOPP
End Sub
Private Sub START()
    Call clock(1): Call data(1): Call data(0): Call clock(0)
```

```
End Sub
Private Sub STOPP()
   Call data(0): Call clock(1): Call data(1)
End Sub
Private Sub GETBYTE()
   i2caus = 0
   For i = 7 To 0 Step -1
   If Comm1.CTSHolding = False Then i2caus = i2caus + 2 ^ i
   Call clock(1)
   Call clock(0)
   Next
End Sub

Private Sub PUTBYTE()
   For i = 7 To 0 Step -1
      If (schreibe And 2 ^ i) = 0 Then Call data(0) Else Call data(1)
      Call clock(1)
      Call clock(0)
   Next
End Sub
Private Sub GETACK()
   Call data(1)
      Do While Comm1.CTSHolding = True
      Loop
   Call clock(1)
   Call clock(0)
End Sub
Private Sub GIVEACK()
   Call data(0): Call clock(1): Call clock(0): Call data(1)
End Sub
Private Sub GIVENOACK()
Call data(1): Call clock(1): Call clock(0): Call data(0)
End Sub
Private Sub data(wert)
If wert = 1 Then Comm1.RTSEnable = False
If wert = 0 Then Comm1.RTSEnable = True
End Sub
Private Sub clock(wert)
If wert = 1 Then Comm1.DTREnable = False
If wert = 0 Then Comm1.DTREnable = True
End Sub

Private Sub Frame_steuerung_MouseDown(ByVal Button As Integer, ByVal Shift As Integer, ByVal
X As Single, ByVal Y As Single)

End Sub

Private Sub HScroll1_Change()
Label2 = HScroll1.Value
End Sub

Private Sub Label3_Click()

End Sub
```

```
Private Sub MenuBar1_ButtonClick(ByVal Button As MenuBarLib.MenuBarButton)

End Sub

Private Sub MenuBar1_MenuClick(ByVal Item As MenuBarLib.Item)
Select Case Item.Caption

Case "Relais": _
   Frame_temperatur.Visible = False: _
   Frame_logger.Visible = False: _
   Frame_steuerung.Visible = True

Case "Temperaturmessung": _
   Frame_steuerung.Visible = False: _
   Frame_logger.Visible = False: _
   Frame_temperatur.Visible = True

Case "Datenlogger": _
   Frame_steuerung.Visible = False: _
   Frame_temperatur.Visible = False: _
   Frame_logger.Visible = True
   txt.dateiname = Day(Now) & "_" & Month(Now) & "_" & Year(Now) & ".txt"
Case "wirklich?": App.End
End Select
End Sub

Private Sub genaue_temperaturmessung()
Dim temp, slope_counter, remain_counter

adresse = CInt(txt_ds1621.Text)

schreibe = adresse: Call START: Call PUTBYTE: Call GETACK
schreibe = &HAC: Call PUTBYTE: Call GETACK
' Der Wert im Register wird gelesen
Call GETBYTE: Call GIVEACK

' Der Oneshotwert wird geschrieben
If (i2caus And 2 ^ 0) = 0 Then schreibe = i2caus + 1
Call PUTBYTE: Call GETACK

' zunächst wird eine Umwandlung mit &H22 angehalten
schreibe = adresse: Call START: Call PUTBYTE: Call GETACK
schreibe = &H22: Call PUTBYTE: Call GETACK

' zunächst wird eine Umwandlung mit &HEE eingeleitet
schreibe = adresse: Call START: Call PUTBYTE: Call GETACK
schreibe = &HEE: Call PUTBYTE: Call GETACK

' Label, um den Lesevorgang einzuleiten
' Nachsehen, bis MSB - Bit 'DONE' gesetzt wurde

Do While (i2caus And 2 ^ 7) = 0
   schreibe = adresse + 1: Call START: Call PUTBYTE: Call GETACK
   Call GETBYTE: Call GIVEACK
Loop 'if (i2caus and 1 shl 7)=0 then goto nochmals
```

```
Call GETBYTE: Call GIVENOACK

schreibe = adresse: Call START: Call PUTBYTE: Call GETACK
' Den Slopecounter auslesen
schreibe = &HA9: Call PUTBYTE: Call GETACK
schreibe = adresse + 1: Call START: Call PUTBYTE: Call GETACK
Call GETBYTE: Call GIVEACK

slope_counter = i2caus
Call GETBYTE: Call GIVENOACK: Call STOPP

schreibe = adresse: Call START: Call PUTBYTE: Call GETACK
' den Remaincounter auslesen
schreibe = &HA8: Call PUTBYTE: Call GETACK

schreibe = adresse + 1: Call START: Call PUTBYTE: Call GETACK
Call GETBYTE: Call GIVEACK

remain_counter = i2caus
Call GETBYTE: Call GIVENOACK

schreibe = adresse: Call START: Call PUTBYTE: Call GETACK
' das Temperaturregister anwählen
schreibe = &HAA: Call PUTBYTE: Call GETACK

schreibe = adresse + 1: Call START: Call PUTBYTE: Call GETACK
Call GETBYTE: Call GIVEACK

i2cein = i2caus
Call GETBYTE
' da wir genau messen benötigen wir den .5 er Dezimalwert nicht
Call GIVENOACK: Call STOPP

temp = FormatNumber(i2cein - 0.25 + ((slope_counter - remain_counter) / slope_counter), 2)

If temp <> tempi Then Call thermo_balken(temp)
tempi = temp

If log_start Then Call log_schreiben

End Sub

Private Sub thermo_balken(temp)
   If log_start Then lbl_akttemperatur = temp Else txt_tempausgabe = temp
   pb1.DrawLine 15, 149, 22, 10, RGB(192, 192, 192), True, True
   pb1.DrawLine 15, 149, 22, 148 - (135 / 45 * temp), vbRed, True, True
End Sub
Private Sub einfache_temperaturmessung()
Dim temp
adresse = CInt(txt_ds1621.Text)

schreibe = adresse: Call START: Call PUTBYTE: Call GETACK
schreibe = &HEE: Call PUTBYTE: Call GETACK: Call STOPP
```

```
schreibe = adresse: Call START: Call PUTBYTE: Call GETACK
schreibe = &HAA: Call PUTBYTE: Call GETACK
Call START: schreibe = adresse + 1: Call PUTBYTE: Call GETACK
Call GETBYTE: Call GIVEACK
temp = i2caus
Call GETBYTE: Call GIVENOACK: Call STOPP
If i2caus = 128 Then temp = temp & ",5" Else temp = temp & ",0"

If temp <> tempi Then Call thermo_balken(temp)
tempi = temp
End Sub



Private Sub Timer1_Timer()
Dim chip, wert
If Check33.Value Then wert = wert + 1
If Check34.Value Then wert = wert + 2
If Check35.Value Then wert = wert + 4
If Check36.Value Then wert = wert + 8

i2cein = CInt(Rnd * 256)
chip = CInt(Rnd * wert) + 1
If (wert And 2 ^ 0) Then adresse = CInt(txt_systemadresse1): Call SCHREIBEN
If (wert And 2 ^ 1) Then adresse = CInt(Txt_systemadresse2): Call SCHREIBEN
If (wert And 2 ^ 2) Then adresse = CInt(txt_systemadresse3): Call SCHREIBEN
If (wert And 2 ^ 3) Then adresse = CInt(txt_systemadresse4): Call SCHREIBEN


End Sub
```

Das Programm ist umfangreicher als in der Sendung dargestellt. Es werden hier auch die Routinen gezeigt, wie man einen I2C – Temperaturchip (DS1621, LM75) ausliest. Ich wollte dies eigentlich auch in der Sendung zeigen, doch am Vorabend hat mein DS1621 den Geist aufgegeben.

Die serielle Schnittstelle

Hauptakteur bei dem vorgestellten Beitrag war die serielle Schnittstelle, Sie soll benutzt werden, um Signale zu erzeugen, die sie normalerweise für diesen Zweck nicht erzeugen muss. Sie soll die beiden Datensignale DATA und CLOCK liefern und außerdem soll noch ein Eingang bereitgestellt werden, damit Inhalte der I2C – Chips gelesen werden können.

Bei dem iPaq 3850 liegen normgemäß die brauchbaren Signale vor: man kann RTS (Request to send) und DTR (Data Terminal ready) als Ausgänge nutzen. DTREnable und RTSEnable können über die COMM-Control, die hier zum Einsatz kommt, eingestellt werden. Über CTSHolding kann dann auch ein Eingang realisiert werden – wir sehen das noch.

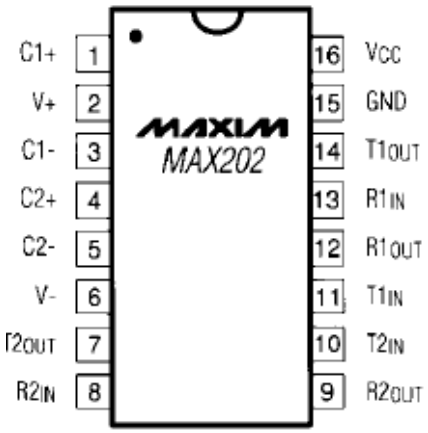
Die serielle Schnittstelle ist eine der ältesten Kommunikationsmöglichkeiten, die man nutzte, um mit dem Computer nach draußen zu kommen. Modem und Co wurden von der seriellen Schnittstelle bedient, die Maus hing lange Zeit an diesem Stecker. Früher war er noch 25 polig – heute ist 9 polig der Standard – wenn es überhaupt noch eine serielle Schnittstelle am Computer gibt.

Die serielle Schnittstelle liefert komische elektrische Signale: aus -18 Volt und +18 Volt werden die

normalerweise 0V und 5V (3V) erzeugt. Beim Laptop sind diese 18 Volt schon reduziert auf ca. –10 Volt und +10 Volt.

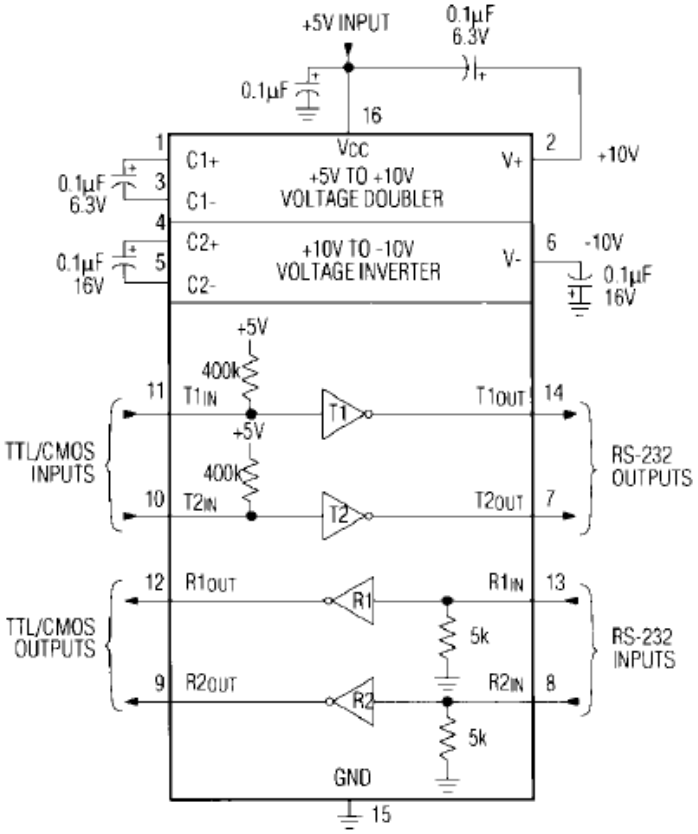
Die I2C- Chips, wie der oben eingesetzte Porterweiterer PCF8574 mögen die normale TTL – Logik, die zwischen 0 Volt und + 5 Volt abläuft. Deshalb muss man der seriellen Schnittstelle des PDA einen Pegelwandler verpassen, der die RS232 – Signale in TTL-Signale und umgekehrt umwandelt.

Hierzu eignet sich der Chip MAX 202 von der Fa. Maxim.
Hier ist die Pinbelegung des Chips zu sehen:



Mit 5 Kondensatoren werden die benötigten Spannungen erzeugt. Die sogenannten Ladepumpen bringen aus 5 Volt Eingangsspannung ca. +/- 10 Volt. Auf der anderen Seite: aus den RS232 – Signalen +/- 10 Volt werden echte 0/5 Volt TTL-Pegel.

Im nächsten Bild ist die interne und externe Verschaltung zu sehen.



Wird der Chip mit den 0.1uF – Kondensatoren bestückt, dann liefert er die richtigen Potentiale.

Jetzt wird Pin 13 des max202 mit dem Ausgang CTS der seriellen Schnittstelle bestückt.
Pin 12 wird als Data- Signal benutzt.
Pin 8 wird mit DTR verbunden. Pin 9 ergibt das Clocksignal.

Das Eingangssignal CTSHolding habe ich direkt geschaltet – ohne Umwandlungspotentiale-. Auf RTS wird die Dataleitung gelegt.

Achtung: Wichtig ist, dass die Dataleitung entkoppelt wird. Sie haben es oben gelesen: das Lesen und Schreiben sind zwei verschiedene Schuhe. Will man lesen, so muss man die Datenwechsel am Chipausgang untersuchen. Da man die Dataleitung nur setzen kann, wirkt sie nicht bidirektional. Das heißt: sie steuert einen Ausgang und sie entdeckt nicht, ob ein Ausgang seine Polarität gewechselt hat

Dafür muss man hier eine Diode einsetzen. Alles, was man steuern will, liefert die Diode ab; alles was man lesen will: die Diode verhindert die Einmischung in das Geschäft. Über die Diode lassen sich die Potentialwechsel einlesen.