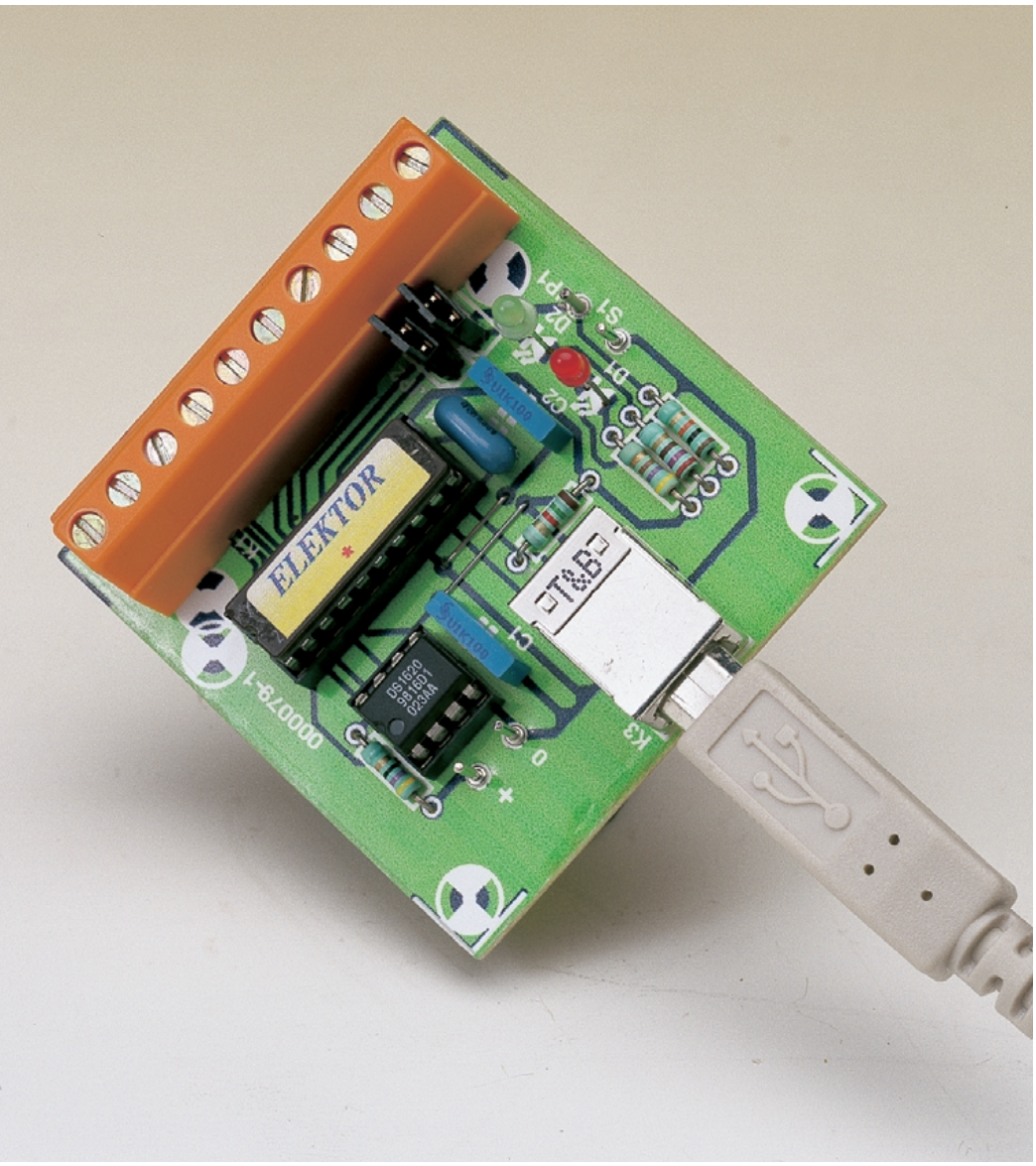
USB-Interface

Experimente mit dem Universal Serial Bus

Von Burkhard Kainka

Das Thema USB ist in aller Munde. Diese neue serielle Schnittstelle hat die Chance, viele der bisherigen PC-Schnittstellen langfristig abzulösen. Deshalb wird sich auch Elektor mit dem Thema beschäftigen.



Vor der Praxis aber erst einige wenige Grundlagen zu Universal Serial Bus. In der USB-Version 1.1 gibt es Lowspeed-Geräte mit 1,5 Mb/s und Fullspeed-Geräte mit 12 Mb/s. Bereits die geringere Übertragungsgeschwindigkeit überragt die möglichen Baudraten an der seriellen Schnittstelle um ein Vielfaches.

Bisher kam es oft zu der unangenehmen Situation, dass alle Schnittstellen des PC bereits belegt waren. Der USB bringt hier den Vorteil, dass mit dem Anschluss eines zusätzlichen Busverteilers (Hub) wieder vier neue Ports freistehen. Insgesamt können bis zu 127 Geräte angeschlossen werden.

Der USB-Anschluss liefert gleich auch die Betriebsspannung für kleinere Geräte. Bis zu 100 mA können ohne weiteres entnommen werden. Mit einer speziellen Anmeldung des Bedarfs dürfen es auch 500 mA sein. Typische Laboranwendungen kommen oft mit wesentlich weniger aus. Der Wegfall eines zusätzlichen Versorgungskabels pro Gerät hilft im Kampf gegen den allgemeinen Kabelsalat.

USB-Geräte können grundsätzlich im laufenden Betrieb angeschlossen und getrennt werden. Das Betriebssystem lädt automatisch den erforderlichen Treiber. Diese erweiterte Plug-And-Play-Fähigkeit erleichtert den Umgang mit mehreren Geräten erheblich. Ein neu angeschlossenes

Gerät erhält automatisch eine Bus-Nummer (Enumeration). Das Betriebssystem liest vorher einige Informationen aus dem Gerät und lädt automatisch den passenden Treiber.

Jedes USB-Gerät ist daher von vornherein wesentlich intelligenter als ein Gerät an der RS232-Schnittestelle. Es muss sich dem Betriebssystem durch eine Reihe von Beschreibungstabellen (Deskriptoren) zu erkennen geben, so dass der richtige Treiber ausgewählt werden kann. Beim ersten Anschluss verlangt das System nach einer Diskette mit dem Treiber. Bei allen folgenden Neuanschlüssen wird der Treiber automatisch geladen, so dass der Anwender nicht mehr eingreifen muss. Trennt man das Gerät wieder vom PC, dann wird der Treiber automatisch aus dem Speicher entfernt.

Die Anwendung des USB ist derzeit leider auf PCs mit dem Betriebssystem Windows 98 eingeschränkt. Unter Windows 95 ist USB zwar ab der Version SR2 prinzipiell möglich, jedoch wird nicht die volle Zuverlässigkeit erreicht. Für die Arbeit mit dem USB-Interface wird daher Windows 98 empfohlen. Windows NT4 unterstützt den USB überhaupt nicht, erst NT5 und Windows 2000. Allerdings ist die zu diesem Projekt gehörende Software nicht unter Windows 2000 getestet.

Die Schaltung

Das USB-Interface basiert auf dem Mikrocontroller CY7C63000 von Cypress. Dieser Controller wurde in erster Linie für kleine USB-Geräte wie Mäuse und Joystickports entwickelt und enthält eine vollständige USB-Engine für den Lowspeed-USB. Die beiden Datenleitungen D+ und D- werden direkt am Controller angeschlossen. **Bild 1** zeigt das Blockschaltbild des ICs. Der Controller enthält neben einem RISC-Prozessor ein OTP-ROM für das Betriebsprogramm (Firmware), zwei Ports, von denen insgesamt zwölf Leitungen herausgeführt sind, sowie RAM und Timer.

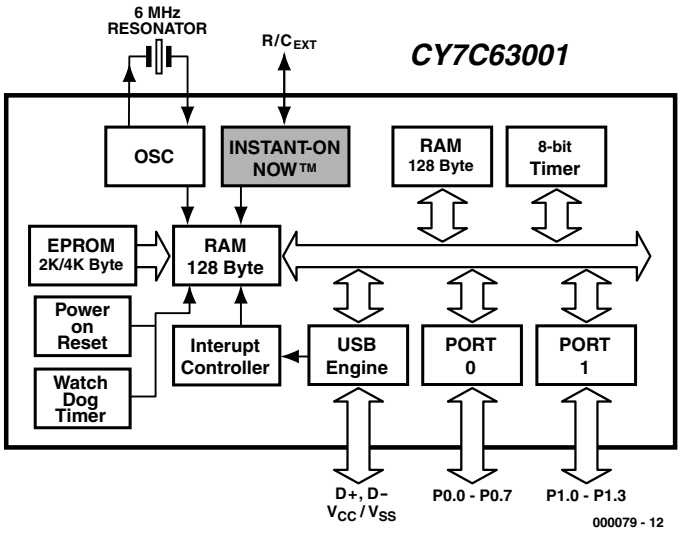


Bild 1. Blockschaltbild des CY7C63001

Das hier vorgestellte USB-Interface basiert

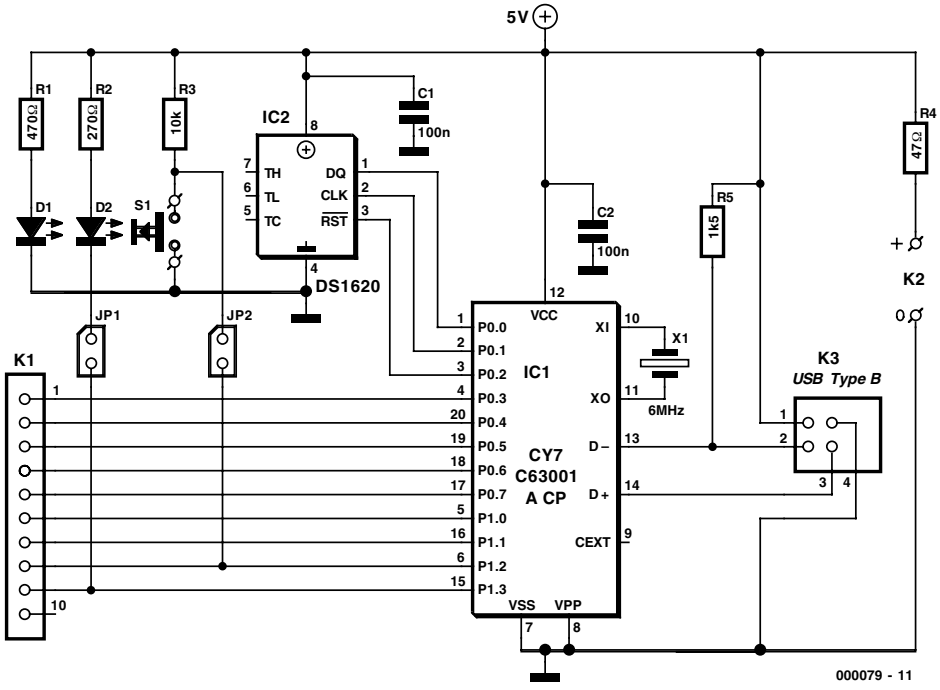


Bild 2. Vollständiges Schaltbild des USB-Interface mit Thermosensor.

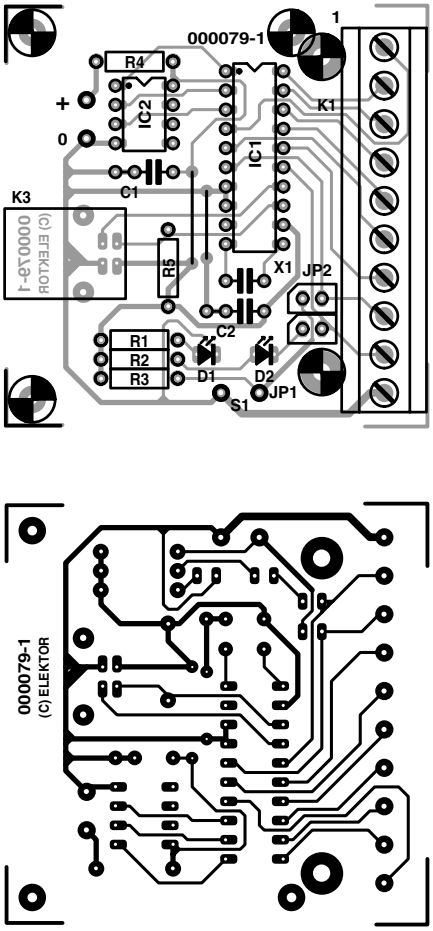


Bild 3. Auf dieser Platine wird das USB-Interface aufgebaut.

Stückliste

Widerstände:
R1 = 470 Ω
R2 = 270 Ω
R3 = 10 k
R4 = 47 Ω
R5 = 1k5

Kondensatoren:
C1,C2 = 100 n

Halbleiter:
D1 = LED rot
D2 = LED grün (low effectivity)
IC1 = CY7C63001ACP (EPS 000079-41)
IC2 = DS1620 (Dallas)

Außerdem:
JP1,JP2 = 2-polige Jumper
K1 = 10-polige Platinenanschlussklemme (oder 2·5)
K2 = Lötnägel
K3 = USB-Platinenverbinder Typ B (Farnell 153-503)
S1 = Drucktaster 1·an
X1 = Keramischer Resonator 6 MHz (Murata CSA6.00MG bei Farnell 295-292 oder Newport ZTA6.00MT)
Gehäuse: 61·22·80 mm³ (Conrad 522848)
Diskette EPS 000079-11
Platine EPS 000079-1

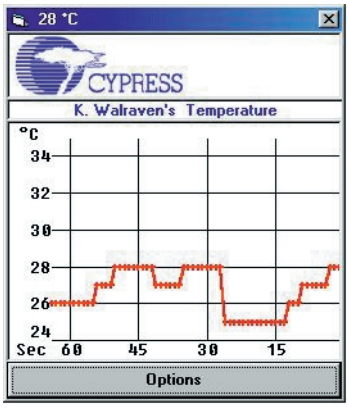


Bild 4. Bildschirmdarstellung der originalen Cypress-Demo.

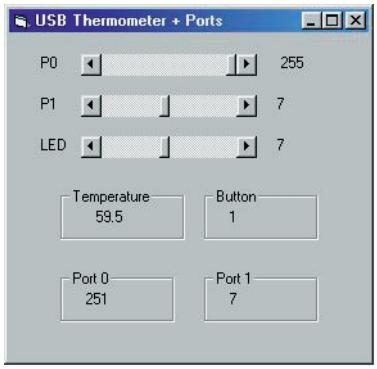


Bild 5. Das Programm erlaubt den Zugriff auf alle Portfunktionen des Controllers und zeigt auch die Temperatur an.

auf einer Applikation von Cypress. Mit dem Starterkit CY3640 hat Cypress sein USB-Thermometer herausgegeben. Das Kit diente in erster Linie zum Kennenlernen des Prozessors und der Schaltungstechnik. Inzwischen ist das Starterkit in dieser Ausführung nicht mehr lieferbar, lebt aber in diesem Elektor-Projekt weiter. Man kann die ersten Erfahrungen mit dem USB sammeln und das USB-Interface für viele unterschiedliche Zwecke ernsthaft einsetzen. Das USB-Interface besitzt folgende Funktionen:

– Messung der Temperatur
– Abfrage eines Tasters
– Einstellung der Helligkeit einer LED
– Insgesamt neun frei zugängliche I/O-Portpins

Das USB-Interface ist voll kompatibel zur ursprünglichen Thermometer-Applikation von Cypress. Es können daher alle Beispielprogramme und der Thermometer-Treiber eingesetzt werden. In der ursprünglichen Anwendung funktionierten allerdings die Portzugriffe nicht vollständig. Die Firmware wurde daher modifiziert und erhielt zusätzliche Funktionalitäten.

Bild 2 zeigt die Schaltung. Die Temperatur misst ein Dallas-Chip DS1620, der insgesamt drei Portleitungen (P0.0...P0.2) belegt. Die restlichen neun Leitungen (P0.3...P0.7, P1.0...P1.3) sind für allgemeine Anwendungen auf die Klemmverbinder-Leiste K1 geführt. Zusätzlich kann eine grüne LED (kein Low-current-Typ!) über den Jumper JP1 an den Portanschluss P13 gelegt werden. Diese LED zeigt die erfolgreiche Enumeration und kann in 16 Stufen in der Helligkeit eingestellt werden. Der Taster S1 ist über Jumper JP2 mit dem Port P1.2 verbunden. Der Taster toggelt ein internes Register des Programms, das über den USB abgefragt werden kann.

Die beiden Busleitungen D+ und D- sind auf einen USB-Verbinder Type B geführt. Wer einen solchen Verbinder nicht auftreiben kann, darf auch das USB-Kabel direkt auf der Platine verlöten. Das USB-Interface erhält seine Betriebsspannung (+5 V, Masse) über den Bus. Die zweite (rote) LED D1 - hier sollte ein energiesparender Typ eingesetzt werden - signalisiert das Vorhandensein der Betriebsspannung. Sie kann auch an den Lötnägeln (0, +) für weitere Applikationen verwendet werden, allerdings sorgt der 47-Ω-Widerstand R4 dafür, dass auch im Kurzschlussfall nicht mehr als die erlaubten 100 mA aus dem USB-Anschluss fließen.
Der Controller wird von einem keramischen Resonator mit 6 MHz und zwei (nicht drei!) Anschlüssen getaktet. Die nötigen Kondensatoren sind im Controller schon integriert.

Infos im Netz

Microsoft VB5
www.microsoft.com/msdownload/vbcce.htm

Cypress Controller CY7C63001ACP
http://www.cypress.com/cypress/prodgate/usb/cy7c63001.html

Dallas Temperatursensor
http://www.dalsemi.com/datasheets/pdfs/1620.pdf

Aufbau und Test

Der Aufbau der kleinen Platine in **Bild 3** sollte keinerlei Probleme hervorrufen. Für beide ICs dürfen Fassungen verwendet werden. Klemme 10 an K1 ist bei der im EPS-Service lieferbaren Platine nicht verbunden, man kann (wenn gewünscht) den Anschluss auf der Platinenunterseite mit einem kurzen Draht an Masse oder +5 V legen. Im hier abgedruckten Layout führt Klemme 10 Mas-

Listing I. Das Modul USBI.BAS mit Deklarationen

```
Attribute VB_Name = "Module1"
Type SECURITY_ATTRIBUTES
nLength As Long
lpSecurityDescriptor As Long
bInheritHandle As Long
End Type

Type OVERLAPPED
 Internal As Long
 InternalHigh As Long
 offset As Long
 OffsetHigh As Long
 hEvent As Long
End Type

Declare Function CreateFile Lib "kernel32" Alias "CreateFileA"
  (ByVal lpFileName As String, ByVal dwDesiredAccess As Long,
   ByVal dwShareMode As Long, lpSecurityAttributes As SECURITY_ATTRIBUTES,
   ByVal dwCreationDisposition As Long, ByVal dwFlagsAndAttributes As Long,
   ByVal hTemplateFile As Long) As Long
Declare Function DeviceIoControl Lib "kernel32" (ByVal hDevice As Long,
   ByVal dwIoControlCode As Long, lpInBuffer As Any, ByVal nInBufferSize
   As Long, lpOutBuffer As Any, ByVal nOutBufferSize As Long,
   lpBytesReturned As Long, lpOverlapped As OVERLAPPED) As Long
Declare Function CloseHandle Lib "kernel32" (ByVal hObject As Long) As Long

Public Security As SECURITY_ATTRIBUTES
Public gOverlapped As OVERLAPPED
Public hgDrvrHnd As Long
Public Const GENERIC_READ = &H80000000
Public Const GENERIC_WRITE = &H40000000
Public Const FILE_SHARE_WRITE = &H2
Public Const FILE_SHARE_READ = &H1
Public Const OPEN_EXISTING = &H3
```

Listing 2. Zugriff auf alle Ports und Funktionen

```
Dim sFileName As String
Dim htemp As Long
Dim lIn As Long, lInSize As Long, lOut As Long, lOut-
Size As Long, lSize As Long
Dim lTemp As Long
Sub USB_IO()
  sFileName = "\\.\Thermometer_0"
  hgDrvrHnd = CreateFile(sFileName, GENERIC_WRITE Or
    GENERIC_READ, FILE_SHARE_WRITE Or FILE_SHARE_READ,
    Security, OPEN_EXISTING, 0, 0)
  lTemp = DeviceIoControl(hgDrvrHnd, 4&, lIn, lInSize,
    lOut, lOutSize, lSize, gOverlapped)
  htemp = CloseHandle(hgDrvrHnd)
End Sub

Sub Brightness(Level)
  lIn = Level * 256 + 14
  lInSize = 2
  lOutSize = 1
  USB_IO
End Sub

Function RdPort(Port) As Integer
  lIn = Port * 256 + 20
  lInSize = 2
  lOutSize = 2
  USB_IO
  RdPort = (lOut / 256) And 255
End Function

Sub WrRAM(Adresse, Wert)
  lIn = 65536 * Wert + Adresse * 256 + 23
  lInSize = 3
  lOutSize = 1
  USB_IO
End Sub

Private Sub HScroll1_Change()
  Wert = HScroll1.Value * 8 + 7
  WrRAM 46, Wert
  Label4.Caption = Str$(Wert)
End Sub

Private Sub HScroll2_Change()
  WrRAM 47, HScroll2.Value
  Label5.Caption = HScroll2.Value
End Sub

Private Sub HScroll3_Change()
  Brightness HScroll3.Value
  Label6 = HScroll3.Value
End Sub

Private Sub Timer1_Timer()
  lIn = 11
  lInSize = 1
  lOutSize = 3
  USB_IO
  Temp = ((lOut \ 256) And 255) / 2
  Minus = (lOut \ 65536) And 255
  If Minus > 0 Then Temp = Temp * -1
  Button = (lOut \ 16777216) And 255
  Label7.Caption = Str$(Temp)
  Label8.Caption = Str$(Button)
  Label9.Caption = Str$(RdPort(0))
  Label10.Caption = Str$((RdPort(1)) And 15)
End Sub
```

sepotenzial. Die USB-Platinenbuchse nimmt über die Kabelabschirmung Kontakt mit Masse auf, die eigentliche Masse (das 0-V-Potenzial) ist Pin 4 der Buchse.

Wenn alles fertig aufgebaut und kein offensichtlicher Bestückungsfehler zu finden ist, kommt der große Moment. Schließen Sie das USB-Interface mit einem USB-Kabel vom Typ A-B an den PC an. Nach einem kurzen Moment erkennt Windows, dass eine neue Hardware ange-

Funktionen

0Bh: Read Thermometer
  lIn: 0Bh (Länge: 1 Byte)
  iOut: Button, Vorzeichen, Temp, Status (Länge: 4 Bytes)

0Eh: Set LED Brightness
  lIn: Helligkeit (0...15), 0Eh (Länge: 2 Bytes)
  iOut: Status (Länge: 1 Byte)

14h: Read Port
  lIn: Port (0,1), 14h (Länge: 2 Bytes)
  iOut: Wert, Status (Länge: 2 Bytes)

15h: Write Port
  lIn: Wert (0...255), Port (0,1), 15h (Länge: 3 Bytes)
  iOut: Status (Länge: 1 Byte)

16h: Read RAM
  lIn: Adresse (0...255), 16h (Länge: 2 Bytes)
  iOut: Wert, Status (Länge: 2 Bytes)

17h: Write RAM
  lIn: Wert (0...255), Adresse (0...255), 17h (Länge: 3 Bytes)
  iOut: Status (Länge: 1 Byte)

18h: Read ROM
  lIn: Index, Adresse (0...255), 18h (Länge: 3 Bytes)
  iOut: Wert, Status (Länge: 2 Bytes)

Diskette EPS 000079-II

CypressSemiconductorsCYPRESS.INF	Info zur Installation der USB-Treiber
Thermometer.exe	Cypress-Demosoftware
USB_20e.ASM	Assembler für Controller
Usb_20e.hex	HEX-Datei für Controller
Usb_20e.lst	Quellkode für Controller
USBTherm.sys	USB-Treiber
USBelektor.vbp	Visua Basic-Projekt
USBelektor.frm	Visual Basic Format-Datei
usb1.bas	Visual Basic Modul
copyright.txt	Copyright-Hinweise
contetns.txt	Inhalt der Diskette

schlossen wurde. Es erscheint ein Fenster mit der Aufforderung, eine Diskette mit dem passenden Treiber einzulegen. Kommen Sie dem nach und legen Sie die Diskette mit der Bezeichnung EPS000079-11 ins Laufwerk A ein. Das System findet hier die Datei *CypressSemiconductorsCYPRESS.INF* zur Beschreibung des Geräts und des Treibers. Es lädt daraufhin den eigentlichen Treiber *USBTherm.sys* Sobald dieser Vorgang abgeschlossen ist, wird das Gerät enumeriert, also am System angemeldet. Nun leuchtet nicht nur die rote LED, sondern (wenn JP1 gesteckt ist) auch die grüne auf.
Die auf der Diskette enthaltene Demosoftware *Thermometer.exe* von Cypress realisiert ein Thermometer mit grafischer Anzeige des Temperaturverlaufs. Der Anwender kann über den Taster auf dem Board zwischen der Anzeige in Grad Celsius und Fahrenheit umschalten, wenn JP2 gesetzt ist. Außerdem lässt sich die Helligkeit der grünen LED verändern. Der Controller enthält dazu einen 4-Bit-D/A-Wandler zur Einstellung der Stromsenke an den Ausgängen des Ports 1. Die Software erlaubt aber nur die Einstellung an Port P1.3. Die Monitordarstellung des Thermometers zeigt **Bild 4**.

Programmierung in VisualBasic

Ein Interface ist nur dann richtig interessant, wenn man es auch mit eigenen Programmen ansteuern kann. Der eigene Zugang auf das Gerät gelingt über ein kleines VisualBasic-Programm. Verwendet wurde VB5CCE, das von Microsoft kostenlos zum Download zur Verfügung gestellt wird und voll funktionsfähig, aber nicht in der Lage ist, ausführbare Stand-alone-Applikationen zu erzeugen.
Nach der Installation des Programmpakets öffnet man das Projekt in Laufwerk A, selektiert und öffnet USBelektor.VBP. Das Projekt umfasst das Modul USB1.BAS (siehe Kasten) mit einigen Deklarationen und die Format-Datei USBElektor.FRM, die nicht nur für die grafische Darstellung auf dem Monitor, sondern auch für die richtige Applikation (Kommunikation mit dem Cypress-Controller, Verarbeitung der Daten) verantwortlich zeichnet. Das Programm startet durch einen Druck auf das RUN-Menü.
Für Zugriffe auf SYS-Treiber benötigt man die Windows-Funktionen CreateFile, CloseHandle und DeviceIoControl. Sie werden hier im Modul USB1.BAS deklariert.

Die Format Datei USBElektor.FRM (**Listing 2**) erlaubt den Zugriff auf alle Portfunktionen des Controllers. Zusätzlich sollten alle ursprünglichen Funktionen des Thermometers erhalten bleiben. Zur Ansteuerung des Treibers benötigt man einige Informationen, die im folgenden erläutert werden. Dieses Programm erzeugt eine Monitordarstellung wie in **Bild 5**.
Beim Aufruf des Treibers mit DeviceIoControll übergibt man eine Treiber-Funktionsnummer, einen Eingangspuffer lIn und einen Ausgangspuffer lOut. Während die Treiber-Funktionsnummer immer 4 ist, wird die eigentliche Aktion mit einem Steuerbyte in lIn ausgewählt. Der Treiber liefert je nach Funktion zwei bis vier Bytes zurück. Das niederwertigste Byte enthält immer eine Statusinformation zum Erfolg des Zugriffs. Neben dem Auslesen des Thermometers gibt es spezielle Funktionen zum Einstellen der LED-Helligkeit und für Zugriffe auf Speicher und Ports des Controllers. All dies ist im Kasten **Funktionen** aufgelistet.
Die Funktion Write Port funktioniert mit dem vorhandenen Treiber nur zum Teil. Es wurde daher ein Weg gesucht, bei unverändertem Treiber und nur durch Änderungen an der Firmware des Controllers alle Portausgaben zu ermöglichen. Die Lösung fand sich in der Verwendung des WriteRAM-Kommandos. Zwei zusätzliche RAM-Variablen wurden definiert:

```
Port0-Ausgaben: 2Eh
Port1-Ausgaben: 2Fh
```

Nun müssen die Portdaten direkt in bestimmte Speicheradressen im RAM des Controllers geschrieben werden. Die Firmware aktualisiert dann die Portausgänge. Das Betriebsprogramm befindet sich übrigens in der Datei *USB_20e.asm* auf der Diskette.

(000079)rg

Literatur:

B.Kainka
Messen, Steuern, Regeln mit USB
Franzis-Verlag
erscheint Herbst 2000