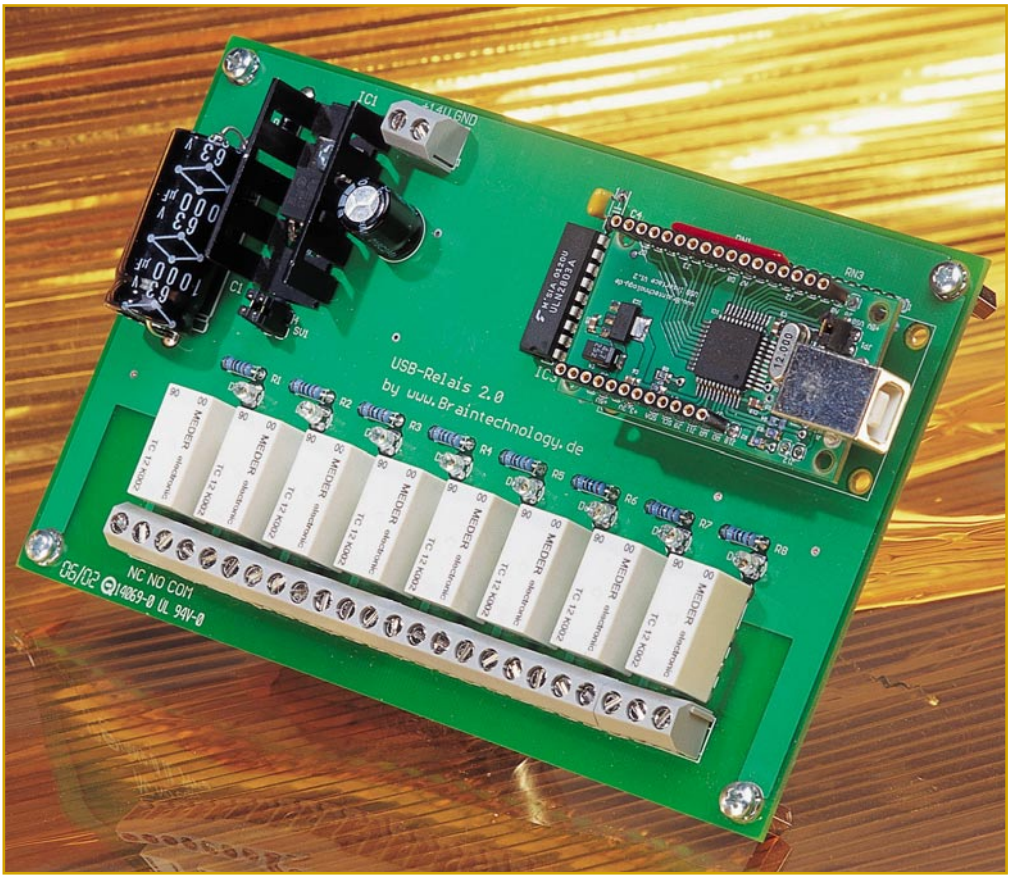

Einfacher andocken - USB-Interface

Der heute an nahezu allen Computern vorhandene Universal Serial Bus (USB) bietet sich für den Anschluss individueller Peripherie geradezu an. Allerdings ist die Programmierung aufgrund des aufwändigen USB-Protokolls nicht ganz einfach. Um dennoch ein einfaches Andocken von Peripherie an den USB zu ermöglichen, stellen wir hier ein kompaktes USB-Interface vor, das einen 8-Bit Adress- und Datenbus, sowie einen I²C-Bus mit I²C-EEPROM bietet. Damit können auf einfachste Weise eigene Schaltungen per USB mit dem PC verbunden werden. Für den schnellen Einstieg und für eigene Experimente dient eine zum USB-Interface passende Relaisplatine.

Allgemeines zum USB

Der USB (Universeller Serieller Bus) ist angetreten, die vielen unterschiedlichen Schnittstellen des PCs sowie die Unmenge „handgestrickter" Protokolle abzulösen. Aus diesem Anspruch heraus resultiert (natürlich) ein durch das USB-Konsortium entwickeltes und kontrolliertes, umfangreiches und entsprechend kompliziertes Protokoll sowie eine genaue Spezifikation der USB-Hardware.

Bevor wir uns näher damit beschäftigen, wollen wir einen kurzen Blick auf die Features und Vorteile des USB werfen.

Am USB sind (theoretisch) bis zu 128 Geräte anschließbar. Über einen meist auf dem Motherboard des PCs integrierten USB-Controller und einen sich anschließenden Rout-Hub können in einer baumartigen Struktur über weitere Hubs die Endgeräte (Functions) angeschlossen werden. Die Datenübertragung erfolgt dabei über eine 2-Draht-Verbindung mit zur Zeit drei spezifizierten Übertragungsgeschwindigkeiten:

- Low Speed Modus 1.2 Mbit/s
- Full Speed Modus 12 Mbit/s
- High Speed Modus 480 Mbit/s

Dabei kann das Endgerät sowohl über eine eigene Stromversorgung als auch über den Bus versorgt werden. Dieser kann maximal 500 mA liefern, was für viele Endgeräte auch ausreichend ist. Zudem verfügen die Hubs meist über ein eigenes Netzteil und versorgen hiermit die an den Hub angeschlossenen USB-Geräte, entlasten also den vom Rechner kommenden Bus-Zweig.

Das Anschließen und Trennen von USB-Geräten kann ohne Ausschalten des Computers geschehen (Hot-Plug).

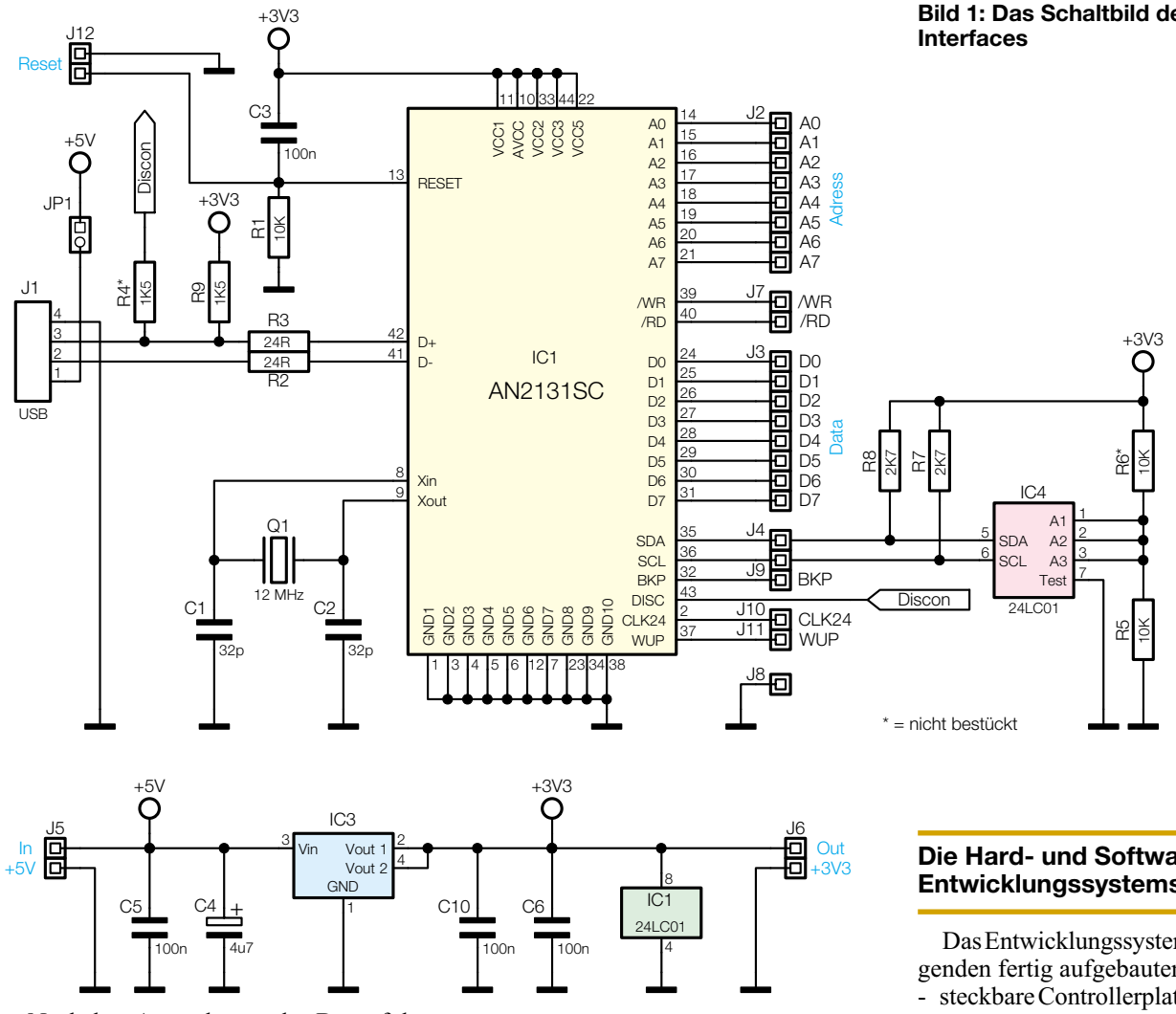

Bild 1: Das Schaltbild des USB-Interfaces

Nach dem Anstecken an den Bus erfolgt eine Enumeration des Gerätes. Dabei wird dem USB-Gerät eine Adresse zugewiesen, unter welcher es später durch den Host (den Controller im PC) angesprochen wird. Weiterhin erfolgt ein Abrufen der Daten des neuen Gerätes sowie das Laden des zugehörigen Treibers. Bereits diese Vorgänge verlangen vom Programmierer recht umfangreiches Wissen auf den Gebieten der PC- und Mikrocontrollerprogrammierung, wie wir noch sehen werden.

Ab Windows 98 ist der USB in das Betriebssystem integriert, sodass er sogar von der DOS-Ebene aus angesprochen werden kann.

Bei der Beschäftigung mit dem USB wird schnell deutlich, dass Entwicklungen auf diesem Gebiet aufgrund der Komplexität der Schnittstelle und der nicht einfachen Einbindung in das Betriebssystem des Rechners alles andere als ein Spaziergang sind. Gerade für Entwickler, die alles im Alleingang bewältigen müssen, bedeutet das einen erheblichen Entwicklungsaufwand, sodass das eigentliche Anliegen, ein elektronisches Gerät für die Nutzung am USB zu bauen, erst nach einer Unmenge Grundlagenstudium zum Thema USB beginnen kann.

USB ohne Grundlagenstudium

Im Folgenden wollen wir eine Möglichkeit vorstellen, den USB für Eigenentwicklungen zu nutzen, ohne sich dazu in die Tiefen der USB-Programmierung begeben zu müssen. Die weiteren Vorteile dieses Konzepts gegenüber anderen dieser Art sind:

- Die Firmware wird vom PC aus über den USB in den Controller geladen.
- Es ist kein Programmiergerät für den USB-Controller notwendig.
- Die Firmware kann problemlos auf den neuesten Stand gebracht werden.
- Eine steckbare Controllerplatine erlaubt das bequeme Testen von Komponenten in Experimentieraufbauten und das universelle Einbinden des Interfaces in eigene Schaltungsaufbauten.
- Eine ActiveX-Komponente erlaubt den einfachen Einsatz der Peripherie unter Visual Basic. ActiveX stellt eine genormte Softwareschnittstelle zum Betriebssystem dar. Über ActiceX-Komponenten (nachfolgend nur mit ActiveX bezeichnet) dockt man quasi über diese Schnittstelle an das Betriebssystem an.
- Sehr große, sofort aufrufbare Funktionsvielfalt.

Die Hard- und Software des Entwicklungssystems

Das Entwicklungssystem besteht aus folgenden fertig aufgebauten Komponenten:

- steckbare Controllerplatine, alle relevanten Anschlüsse sind herausgeführt
- kaskadierbare Relaisplatine für erste Gehversuche bzw. allgemeine Schaltanwendungen
- ActiveX zur einfachen Programmierung und Steuerung der Zielhardware
- Testprogramm zum Austesten der über ActiveX bereitgestellten Methoden.

Die steckbare Interfaceplatine

Das Schaltbild des USB-Interface ist in Abbildung 1 dargestellt.

Die Interfaceplatine trägt bereits alle Bauteile, die zum Betrieb des Controllers notwendig sind. Die Betriebsspannungszuführung kann wahlweise über den USB (Auswahl über Jumper) oder per zusätzlichem 5-V-Anschluss (J 5) über die zu steuernde Schaltung erfolgen. Zum Einsatz kommt ein AN2131 der Firma Cypress. Bei Bedarf (höhere Datenrate) ist auch der AN 2135 einsetzbar. Weiterhin beherbergt die Controllerplatine ein 8k-I²C-EEPROM, in welchem die Firmware bzw. andere Einstellungen gespeichert werden können. Die gesamte Schaltung arbeitet mit einer Betriebsspannung von 3,3 V, die über einen auf der Platine befindlichen Spannungsregler konstant gehalten wird, besitzt je-

Bild 2: Das als Fertigbaustein gelieferte USB-Interface.

doch entsprechend des Experimentiercharakters des Aufbaus 5-V-tolerante Ein-/Ausgänge.

Herausgeführt werden die kompletten Ports B u. C sowie die Anschlüsse Port A.4 und Port A.5 des USB-Controllers. Diese bilden für spätere Anwendungen den 8 Bit breiten Adress- und Datenbus.

Mit der hierfür vorhandenen Software sind bei Verwendung des AN2131 Datenübertragungsraten von 100kByte/s und bei Verwendung des AN2135 sogar 1Mbyte/s möglich.

Alle relevanten Signalanschlüsse sind auf der Platine über Lötanschlüsse zugänglich, sodass die mit 58 x 33 mm recht kompakte Interfaceplatine bequem durch einfaches Aufstecken in die eigene Applikation einbindbar ist. Abbildung 2 zeigt die Interface-Platine. Gut zu erkennen sind die USB-Buchse, der Jumper für die Auswahl der Spannungsversorgung und die Lötanschlüsse. Das EEPROM befindet sich auf der Unterseite der Platine.

Natürlich verfügt die Platine über einen Norm-USB-Anschluss Typ B wie jedes USB-Gerät. Ein USB-Kabel für die Verbindung zum Rechner/Hub befindet sich im Lieferumfang der Platine.

Die Relaisplatine

Die Relaisplatine, deren Schaltbild in Abbildung 3 dargestellt ist, eignet sich sowohl zum ersten Kennenlernen und für einfache Versuche mit dem USB-Interface als auch für das Ausführen von allgemeinen Schaltaufgaben. Die acht Relais auf der 100x 160 mm messenden Platine bieten jeweils einen über Schraubklemmen herausgeführten potenzialfreien Umschaltkontakt (max. mit 1,2 A/150 V belastbar) Über Buchsenleisten wird die USB-Interfaceplatine auf die Relaisplatine aufgesteckt. Die Relaisplatine selbst wird über ein externes Netzteil, entweder mit einer unstabilisierten Spannung von ca. 14 V oder mit einer stabilisierten Spannung von 12 V, versorgt.

Zwei dieser Platinen sind kaskadierbar, sodass man über das USB-Interface sogar bis zu 16 Relais steuern kann. Auf der zweiten Platine sind lediglich der im Schaltbild mit IC 3 bezeichnete Treiber und das Widerstandsnetzwerk RN 1 auf die Position von IC 2 und RN 3 zu versetzen. Zur Verbindung beider Platinen ist eine Buchsenleiste vorhanden.

Die Schaltung selbst ist schnell vorgestellt. Über einen invertierenden 8fach-Treiber vom Typ ULN2803 werden die Relais sowie eine jeweils parallel geschaltete LED angesteuert. Pull-Down-Widerstände sorgen für korrekte Pegel an den Eingängen der Treiber, solange der USB-Controller noch nicht initialisiert ist. Die 12-V-Betriebsspannung für die Platine wird über einen Längsregler aus der über die Schraubklemme KL 9 zugeführten Spannung des Steckernetzteiles erzeugt. Wahlweise kann der Spannungsregler aber auch überbrückt und die Platine mit einer schon vorhandenen, stabilisierten Spannung von 12 V betrieben werden.

Zur Ansteuerung der Relais wird bei bestücktem IC 2 (8-Kanal-Platine) der Port B des USB-Controllers herangezogen. Bei bestücktem IC 3 (kaskadierte Platine) ist Port C für die Ansteuerung der Relais verantwortlich. Das ist später bei der Erstellung der Software von Interesse.

Aufbau

Beide Platinen werden, wie gesagt, komplett aufgebaut geliefert. Im Lieferumfang befinden sich bereits Buchsenleisten mit langen Anschlussbeinchen. Diese sind noch nicht bestückt, um den individuellen Einsatz der Boards zu ermöglichen. Für den Einsatz des USB-Interface in Verbindung

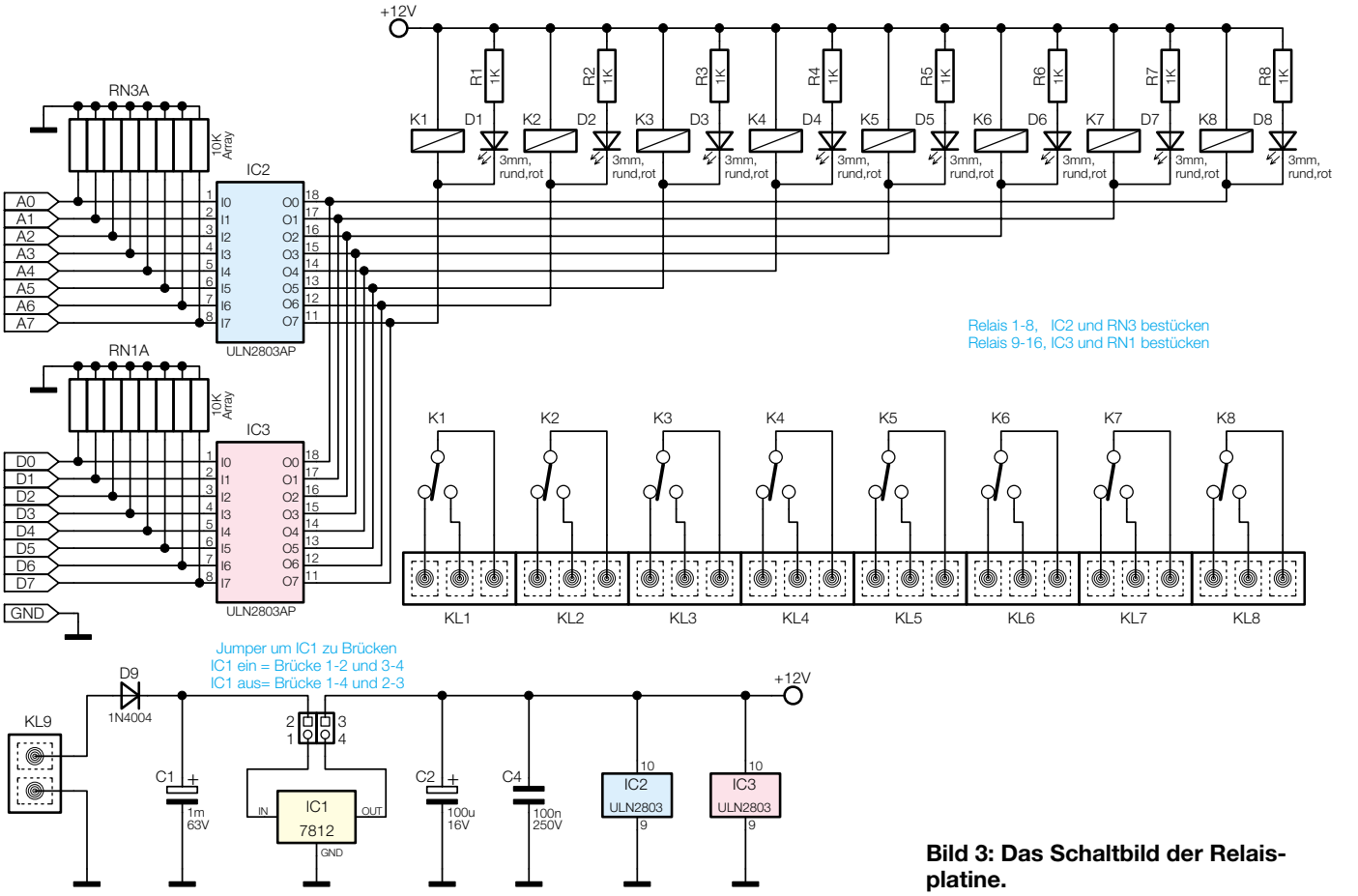

Bild 3: Das Schaltbild der Relaisplatine.

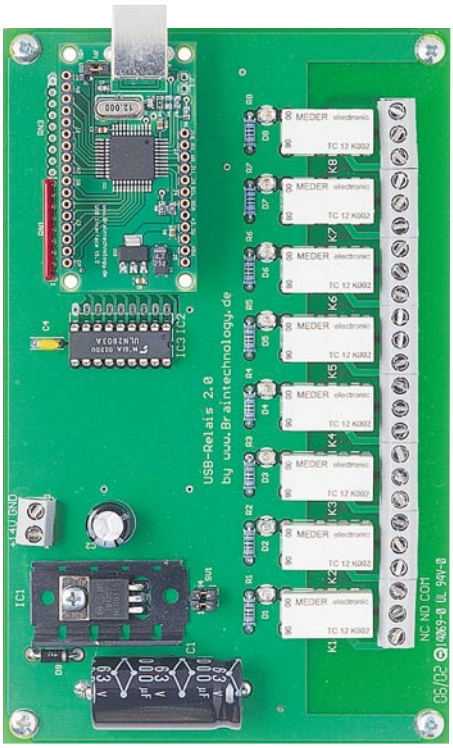

Bild 4: Fertig montierte Relaisplatine mit aufgestecktem USB-Interface

mit dem Relaisboard empfiehlt es sich, die mitgelieferten passenden Buchsenleisten zu verwenden. Dabei sind diese auf die entsprechende Pinzahl zu kürzen und von oben in die Kontakte der Leiterplatten zu stecken. Die Einbauhöhe sollte so gewählt werden, dass die Anschlussbeinchen auf der Unterseite ca. 8 mm herausragen. Die Pins sind anschließend zu verlöten.

Danach kann das USB-Interface mit den Anschlussbeinchen an der Unterseite in die Buchsenleiste der Relaisplatine gesteckt werden (s. Abbildung 4). Durch diese Bauweise können, wie erwähnt, auch zwei Relaisboards übereinander gesteckt werden. Für Messzwecke steht an der Oberseite des USB-Interface eine Buchsenleiste zur Verfügung.

Das ActiveX für die Einbindung in Visual Basic

Während die Hardware noch leicht beherrschbar ist, wird es bei der Software schon deutlich komplizierter. Hier muss zwischen der Firmware (dem Programm für den Controller) und der Hostsoftware, dem Programm für den PC, unterschieden werden. Da das kombinierte Code- und Daten-RAM des Controllers nach Spannungsabschaltung seinen Inhalt verliert, muss nach jedem Zuschalten das Controllerprogramm neu in das RAM geladen und gestartet werden. Die Controllerprogramme befinden sich im ActiveX-Bereich und sind dort als Bytefeld abgelegt. Die Demoversion der ActiveX-Komponente zur Einbindung in die Entwicklungsumgebung von Visual Basic (VB) steht im Internet unter [1] ebenso zum freien Download zur Verfügung wie ein großer Datenpool an Dokumentationen und Beispielprogrammen für das USB-Interface.

Nach der Installation und dem Hinzufügen des ActiveX in die VB-Entwicklungsumgebung ist dieses in der Werkzeugleiste sichtbar. Wird es nun auf einer Form abgelegt, erhält es automatisch den Namen „EZUSB1“ und es können alle Eigenschaften und Methoden eingesehen werden. Tabelle 1 zeigt die wichtigsten Eigenschaften im Überblick.

Da im Rahmen dieses Beitrages nicht alle Methoden des ActiveX ausführlich besprochen werden können, soll hier ein kurzer Überblick gegeben werden. Grundsätzlich kann in mehrere Funktionsgruppen unterschieden werden:

- 8 Bit-Parallelbus mit 8-Bit-Adressbus
- 8 Bit-Parallelbus und frei konfigurierbarer Port C
- allgemeine Methoden des I^2C-Bus
- Methoden für integriertes I^2C-EEPROM
- Methoden zum Einstellen der Datenrichtung sowie zum Setzen und Löschen einzelner Portpins
- Methoden zur Nutzung eigener Controller-Software.

Da nicht alle gewünschten Methoden in einem Controllerprogramm untergebracht werden können, wurden verschiedene Betriebsmodi in der Eigenschaft „MC_Modi“ vereinigt. Unterschieden wird bei der hier vorgestellten Version zwischen 6 Modi, welche im Enumerationstyp „IModi“ zusammengefasst sind:

Tabelle 1:
Die wichtigsten Eigenschaften und Methoden der ActiveX-Komponente

Eigenschaftsname	Datentyp	Klartext
EZUSB1.MC_Mode	IModi	Mikrocontroller-Modus
EZUSB1.Enabled	Boolean	aktiv - nur zur Laufzeit setzbar
EZUSB1.LastErrors	integer	letzte aufgetretene Fehler
EZUSB1.ID	String	Identifikations-Code
EZUSB1.LicenseFile	String	Name der Lizensdatei + kompletter Pfad
EZUSB1.TimeOutIntervale	Long	Timeout - Zeit [ms] für Interface - Methoden
EZUSB1.Device	Byte	Nummer des Cypress-Controllers am Bus *

* Die lizenzierte Version unterstützt bis zu 32 Controller dieser Serie am Bus. Für die Steuerung der Relaisplatine genügt die Shareware-Version. Hier muss als Device-Nummer 0 eingegeben werden.

```
CommonMode - 0 (AN2131/2135)
Imode_1    - 1          (AN2131)
Imode_2    - 2          (AN2131)
Imode_3    - 3          (AN2135)
Imode_4    - 4          (AN2135)
SimpleMode - 5 (AN2131/2135*)
```

* Port B ist beim AN2135 nicht frei verfügbar, sondern für den Fast-Transfer reserviert (1Mbyte/s)

Welche Methoden in welchem Controllermodus zur Verfügung stehen, kann der Dokumentation zur Controllerplatine entnommen werden. Dort sind auch sämtliche Methoden mit Beispielen sowie alle möglichen Fehler aufgeführt und beschrieben. Zusätzlich ist zu jeder Methode ein Aufrufbeispiel in VB angegeben. Der Anwender kann den für ihn günstigsten Modus einstellen. In Abhängigkeit von diesem Modus wird das entsprechende Maschinenprogramm in den Controller geladen. Mit der vorhandenen Funktionalität sind auch komplexe Geräte steuerbar. Beispielsweise wird der hochwertige 100-MHz-Logikanalysator LA 100 USB (16 Kanäle, Abtastfrequenz bis 100 MHz, Abbildung 5) mit Version 1.0 dieses ActiveX betrieben.

Für den ersten Kontakt mit dem ActiveX wird nachfolgend an einem einfachen Beispiel dessen Handhabung beschrieben.

Anwendungsbeispiel Relaisplatine

Das Beispiel beschreibt das Ein- und Ausschalten einzelner Relais der Relaisplatine. Nach dem Aufstecken der Interfaceplatine auf die Relaisplatine werden die Interfaceplatine mit dem Rechner verbunden und das Steckernetzteil an die Relaisplatine angeschlossen.

In der Eigenschaft „MC_Modi“ wird der Wert „SimpleMode” eingestellt. Dadurch sind alle Portpins der Interfaceplatine frei konfigurierbar. Dieser Wert kann natürlich auch im Sourcecode durch folgenden Aufruf gesetzt werden:

```
EZUSB1.MC_Modi = SimpleMode
```

In der Form_Load-Prozedur sollte die Eigenschaft ‘Enabled’ den Wert „True” erhalten. Dadurch wird der Treiber geöffnet, und das zum Controllermodus „SimpleMode” gehörende Maschinenprogramm in den Controller geladen sowie gestartet. Anschließend sollte die Eigenschaft „LastErrors” überprüft werden. War das Setzen der Eigenschaft erfolgreich, so gibt sie den Wert 0 zurück:

```
If EZUSB1.LastErrors <> Err_No Then
  ‘ Fehlerbehandlung
  ‘ eventuell Ausstieg aus der Prozedur
End if
```

Bild 5: Arbeitet ebenfalls mit einer Version des diskutierten ActiveX: 16-Kanal-/100 MHz-Logik-Analyzer LA 100 USB.

Da die Relaisplatine am Port C angeschlossen ist, muss man die Pins dieses Ports als Ausgänge konfigurieren. Das geschieht mit der Methode „SetPortCDir" in folgender Weise :

```
Call EZUSB1.SetPortCDir ( 255 )
```

Soll ein Pin als Ausgang geschaltet werden, so muss das entsprechende Bit in der übergebenen Byte-Variablen gesetzt sein. Pin 0 hat den Wert $2^0 = 1$ ... Pin7 den Wert $2^7 = 128$. Wird der Wert 255 übergeben, so bedeutet das demzufolge, dass alle Pins als Ausgang geschaltet sind. Nach dem erstmaligen Setzen der Pins als Ausgang haben alle Pins L-Pegel. Durch die Invertierung im Relaistreiberbaustein bedeutet das, dass alle acht Relais „abgefallen" sind. Über die Methode „SetPortCPin" kann nun ein Relais eingeschaltet und dementsprechend durch „ClearPortCPin" wieder ausgeschaltet werden.

Die nachfolgenden Code-Zeilen schalten also das Relais 0 ein und wieder aus:

```
Call SetPortCPin (Pin0)
Call ClearPortCPin (Pin0)
```

Der Übergabewert ist ein Enumerationstyp, welcher durch das ActiveX veröffentlicht wird. Somit ist eine bequeme Eingabe im Quelltexteditor möglich.

Die gleiche Verfahrensweise gilt auch für alle anderen Relais (Pins).

Eine andere Möglichkeit, bei der alle Relais gleichzeitig ein- bzw. ausgeschaltet werden, ergibt sich durch die Methode „SetPortC". Wird dieser Methode der Wert 0 übergeben, so sind alle Relais ausgeschaltet. Wird als Übergabewert hingegen 255 übergeben, so sind alle acht Relais eingeschaltet. Der Übergabewert errechnet sich auf die gleiche Weise wie bei der zuvor diskutierten Methode „SetPortCDir". Der folgende Aufruf schaltet die Relais 0 und 7 ein. Alle anderen Relais werden ausgeschaltet.

```
Call EZUSB1.SetPortC (129)
```

Mit diesen wenigen Zeilen ist die Relaisplatine also bereits steuerbar.

Beim Beenden der Anwendung sollte vorher die Eigenschaft 'Enabled' des ActiveX auf den Wert „False" gesetzt werden. Dies kann zum Beispiel in der „Form_Unload"-Prozedur geschehen. Diese sorgt dafür, dass der Treiber wieder geschlossen wird.

Weiterhin sollte man auf jeden Fall regelmäßig die Eigenschaft „LastErrors" abfragen. Wer nicht gleich mit einer eigenen Anwendung beginnen möchte, kann das Testprogramm laden, die Methoden untersuchen und zum Beispiel damit die Relaisplatine ansteuern. Durch diese praktische Beschäftigung mit der Materie wird man schnell die Möglichkeiten des Programms erkennen und beherrschen.

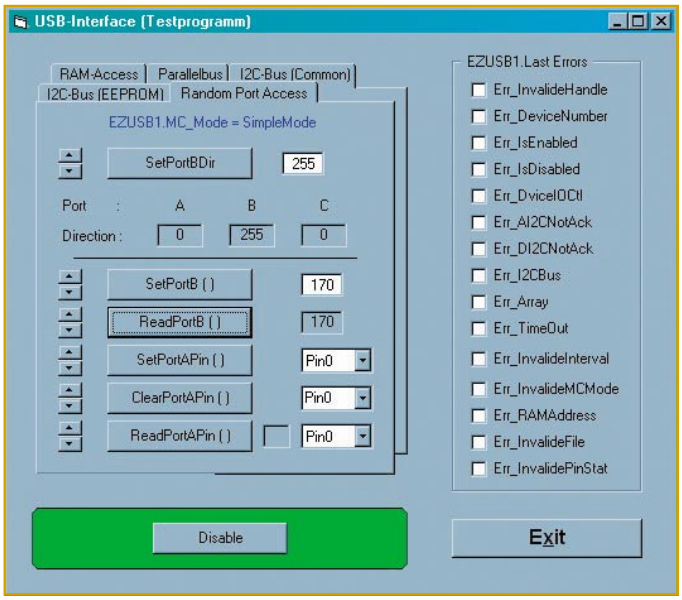

Bild 6: Die Bedien- und Einstelloberfläche des Testprogramms.

Das Testprogramm

Das oben gezeigte Beispiel ist an Einfachheit kaum zu überbieten und lastet den Controller natürlich bei weitem nicht aus. Die weiteren 5 Betriebsmodi bieten mächtige Methoden, die dem Elektronikentwickler eine Menge Programmierarbeit abnehmen. Die Dokumentation zum Interface hält für alle Modi ein Beispiel bereit.

Damit man diese Methoden aufrufen kann, ohne sofort eigene Software erstellen zu müssen, haben wir ein kleines Testprogramm geschrieben, das in Abbildung 6 dargestellt ist. Auf fünf Registerkarten sind die Methoden, in funktionellen Gruppen zusammengefasst, testbar.

In blauer Schrift steht am oberen Rand jeder Registerkarte der gewählte Controllermodus (Eigenschaft MC_Mode). Zur Ansteuerung der Relaiskarte wählt man die Registerkarte „Random Port Access". Damit sind alle Ports frei konfigurierbar. Die auf der Registerkarte befindlichen Einstellelemente erklären sich selbst. Über die links befindlichen Schalter kann eingestellt werden, für welchen der Ports die entsprechende Methode ausgeführt werden soll. Auf den Schaltern ist der Methodenname eingeblendet.

Mit den anderen 4 Registerkarten lassen sich folgende Methoden austesten:

- RAM-Access-Methoden zur Nutzung eigener Firmware
- Parallelbus-Methoden zur Nutzung des 8-Bit-Parallelbus (inkl. 8-Bit-Adressbus)
- I^2C-Bus (Common)-Methoden zur allgemeinen Nutzung des I^2C-Bus
- I^2C-Bus (EEPROM)-Methoden zur Nutzung des 8-k-EEPROM

Für erste eigene Versuche sollte man das Projekt des Testprogramms laden und dort interessierende Teile des Sourcecodes genauer untersuchen. Besonders die Prozedur „Show_Errors" mit den zugehörigen Checkboxen kann im Versuchsstadium eigener Programme schnell Aufschluss über aufgetretene Fehler geben.

So kann man sich Schritt für Schritt einarbeiten und ist bald in der Lage, eigene Programme für Geräte am USB zu erstellen. ELV

Weiterführende Literaturhinweise und Internetseiten:

[1] www.braintechnology.de
[2] Helm, H. J: USB 1.1, Franzis-Verlag
[3] Kainka, B.: MSR mit USB, Franzis-Verlag