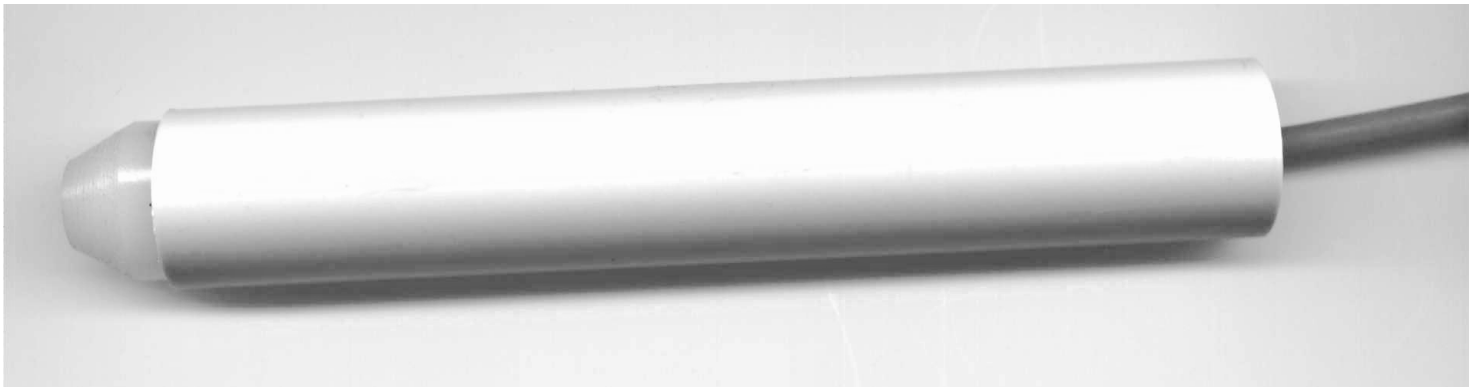
Wenn ein PIC zum Beispiel in einer Timeranwendung (Schaltuhr oder ähnlich) eingesetzt wird, ist der größte Teil der Schaltung für das Benutzerinterface mit Tastern, Schaltern und Display erforderlich. Erfolgen die Einstellungen auf dem PC, kann dieser Aufwand entfallen, dafür wird die Bedienung aber durch die nötige Kabelverbindung zum PC wieder umständlicher. Der Autor beschreibt hier ein einfaches Interface, das mit fast Null Hardwareaufwand das Programmieren einer PIC-Anwendung vom PC aus auf kontaktlosem Weg ermöglicht.

Von H. Bärnthaler

DATEN-BLINKER-INTERFACE

Drahtloses Interface zwischen PC und PICs



Nehmen wir einmal an, Sie wollen zum Beispiel einen komfortablen Timer mit einem Mikrocontroller realisieren. Für die Timerfunktion brauchen Sie nur einen Zähler, der bei einer bestimmten Uhrzeit ein Relais ein- oder ausschaltet. Diese Funktion ist mit wenigen Programmzeilen realisiert.
Wesentlich mehr Programmzeilen und Hardware (Tastatur, Display) brauchen Sie, um den Timer zu konfigurieren. Sie müssen die aktuelle Uhrzeit einstellen , die verschiedenen Schaltzeiten müssen definiert werden, das zu schaltende Relais muß angegeben sein und natürlich können auch verschiedene Optionen ausgewählt werden. Dem Anwender spendieren Sie eine vierseitige Beschreibung , in der wortreich erklärt wird, wie die Einstellungen über das dreistellige Display und die drei zur Verfügung stehenden Tasten vorzunehmen sind.

Die Lösung des Problems

Sie schreiben am PC ein komfortables C-Programm mit schöner Bildschirmmaske und Help-Menü und tragen dort die benötigten Timerdaten ein. Danach halten Sie ein kugelschreiberähnliches Gehäuse mit integriertem Fototransistor auf eine bestimmte Stelle des Bildschirms und übertragen mit einem blinkenden Lichtpunkt auf dem Monitor die 20 benötigten Konfigurationsbytes in den Mikrocontroller.
Die Vorteile sind offensichtlich: Für das komplette Tastatur- und Konfigurationsmanagement wird nur eine Portleitung belegt. Das Display entfällt ebenso wie eine Tastatur, da die Daten am PC eingegeben und grafisch aufbereitet werden. Es braucht kein Kabel an den PC angeschlossen zu werden, PC und PIC-System sind galvanisch vollständig voneinander getrennt. Für das Datenhandling im PIC genügt wenig und einfacher Programmkode, so daß genügend Programmspeicher für die eigentliche Applikation frei bleibt.
Konfigurationsdaten werden komfortabel und übersichtlich am PC verwaltet, die komplette PC-Tastatur steht dem PIC zur Verfügung. Schließlich funktioniert das Verfahren auch auf dem ältesten System, da praktisch jeder PC in der

Lage ist, ein weißes und schwarzes Rechteck auf dem Bildschirm auszugeben. Es wird kein PC-Portanschluß belegt, auch entfällt das leidige Problem der falsch konfigurierten seriellen Verbindung vollständig, und es sind keine speziellen PC-Hardwarekenntnisse notwendig (wie bei serieller PC-Schnittstelle).
Es sollen natürlich auch die Nachteile nicht verschwiegen werden: Der Datentransfer erfolgt nur vom PC zur PIC-Applikation und nicht umgekehrt und ist außerdem langsam (etwa 40 Byte/Minute). Bei vielen Anwendungen ist das aber kein Problem, weil die zu übertragende Datenmenge ohnehin sehr gering ist.

So funktioniert´s

Auf dem PC-Bildschirm wird ein blinkendes Rechteck (mit pulsweitenmodulierter Helligkeit) erzeugt, über das die Datenworte seriell geblinkt werden. Ein Fototransistor leitet die Signale an den Mikrocontroller weiter und dieser setzt das Datenwort wieder zusammen.
Wie die Schaltung in **Bild 1** zeigt, ist die benötigte Hardware sehr einfach. Als Aufnehmer läßt sich jeder Standard-Fototransistor einsetzen, im Schaltbild sind zwei gängige Typen angegeben. Auf den Fototransistor folgt ein einstufiger Verstärker mit einem BC548B, der das Signal so weit auf Pegel bringt, daß das nachfolgende Monoflop an seinem Schmitt-Trigger-Eingang einwandfrei getriggert wird. Die Auswertung des vom Monoflop gelieferten Impulssignals erfolgt direkt durch den PIC-Mikrocontroller am Portanschluß RB0 (Pin 6). Der in dem Beispiel verwendete PIC läuft mit einem 455-kHz-Keramikresonator und ist nur so weit beschaltet, wie es zum Testen des Interface-Programms erforderlich ist.
Wie die beiden Fotos zeigen, läßt sich die Schaltung problemlos auf einem kleinen Lochrasterstreifen aufbauen und anschließend in ein Stiftgehäuse (Kunststoff- oder Metallröhrchen) einbauen.

Software

Für das Senden der Datenbytes vom PC aus wird hier ein Pascal-Listing abgedruckt, das zusammen mit dem (kommentierten) Assemblerlisting für den PIC auch auf der Diskette zu finden ist (ESS986038-1, siehe Serviceanzeige in der Heftmitte).
Das Beispielprogramm “DatenBildschirmBlinker” (Tabelle 1) demonstriert die Funktion des Bildschirmblinkers und überträgt insgesamt 20 Daten-Bytes,

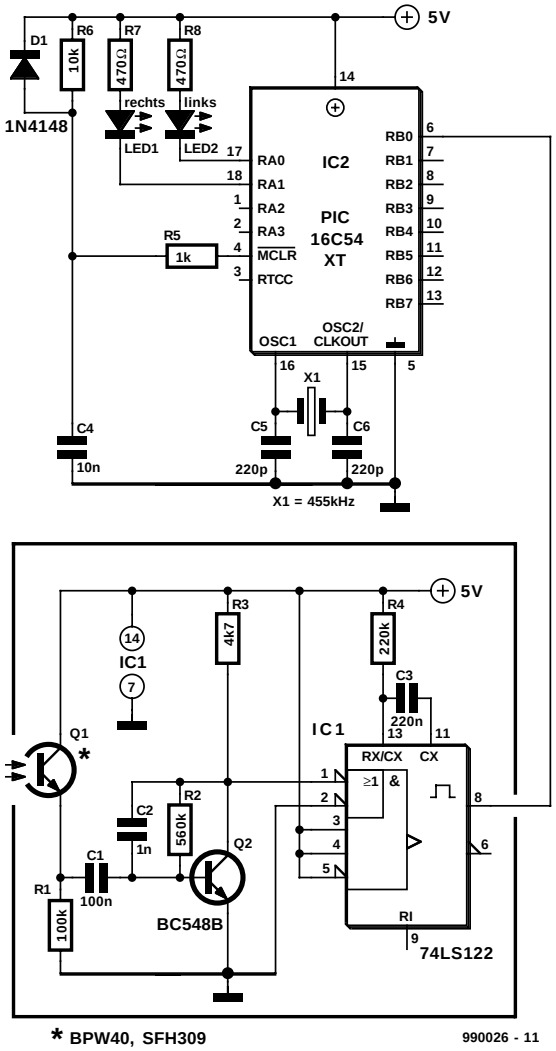


Bild 1. Die für das Interface benötigte Hardware ist sehr einfach und besteht aus einem Fototransistor mit nachfolgendem Verstärkertransistor und einem Monoflop.

Tabelle 1

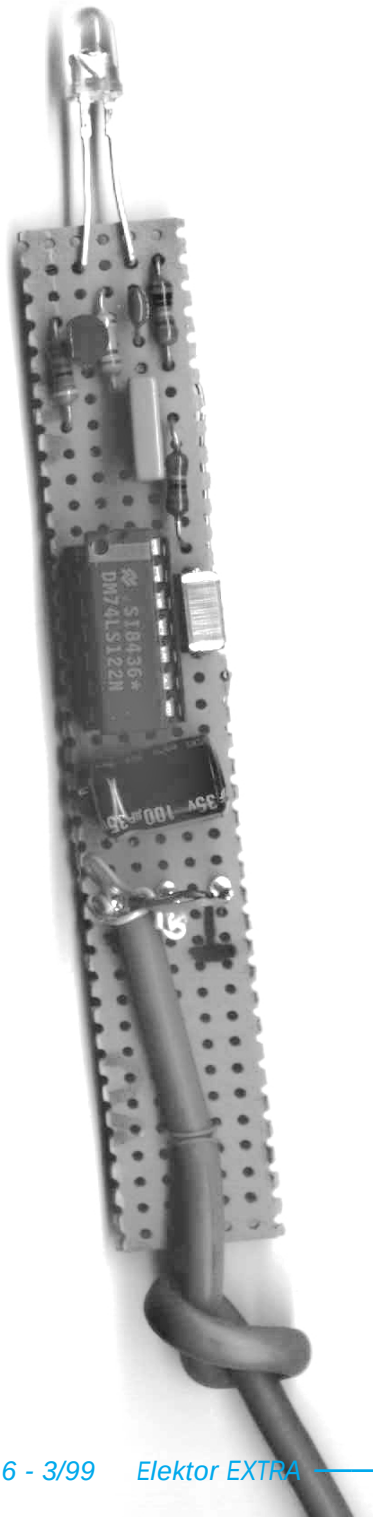
```
Pascal-Listing
*************************************************************}
Program DatenBildschirmBlinker (Input,Output);
Uses  Dos,CRT,Graph;

type
    PointType = record
            x,y: Word;
        end;

Var Var1            : Integer;
    Var2, Var3      : Integer;
    GraphMode, GraphDriver   : Integer;
    x1, y1, Breite, Hoehe    : Integer;
    Messzeitpunkt        : Integer;

const
    Pentagon: array [1..4] of PointType =
    (( x: 50; y:50),
     ( x: 50; y:100),
     ( x: 100; y:100),
     ( x: 100; y:50));
    Schwarz: FillPatternType =   ($00, $00, $00, $00,
                                  $00, $00, $00, $00);
    Weiss   : FillPatternType =  ($FF, $FF, $FF, $FF,
                                  $FF, $FF, $FF, $FF);
    HighLength     = 2000;  {High-Impulslänge}
    LowLength      = 1000;  {Low-Impulslänge}
```

wobei der Dateninhalt zwischen $FF und $00 wechselt. Am Beginn eines jeden Datenbytes wird ein Initbit gesendet, dessen Länge vom Mikrocontroller gemessen wird. Die nachfolgenden Datenbits sind pulsweitenmoduliert.
Alle Impulse, deren Länge über der Initbitimpulslänge liegt, werden als “1” dekodiert und alle kürzeren Impulse als “0”.
Die Datenwortbreite ist variabel und beträgt minimal 1 Bit. Dies muß aber im Mikrocontroller berücksichtigt werden.
Die angegebenen Delayzeiten müssen unter Umständen an den jeweils verwendeten PC und die Bildschirmwiederholfrequenz angepaßt werden:



```
    WaitLength    = 1500;  {Zeit zwischen den Bits}
    InterByteTime = 1000;  {Zeit zwischen den Bytes}

{*********************************************************************
{ In diesem Unterprogramm wird das Startbit geblinkt, dessen Länge
{ vom Microcontroller gemessen wird.
*********************************************************************}
Procedure InitBit;
Begin
    {Initbitlänge berechnen ausgeben}
    MessZeitpunkt := LowLength + ((HighLength-LowLength) div 2);
    SetFillPattern (Weiss, White);
    FillPoly (SizeOf (Pentagon) div SizeOf(PointType), Pentagon);
    Delay (MessZeitpunkt);
    SetFillPattern (Schwarz, Black);
    FillPoly (SizeOf (Pentagon) div SizeOf(PointType), Pentagon);
    Delay (WaitLength);
End;

{*********************************************************************
 Programmstart
*********************************************************************}
Begin
    GraphDriver := Detect;
    InitGraph (GraphDriver, GraphMode, 'c:\pascal\graphix');
    readln;  {Warten auf einen Tastendruck}
    For Var1 := 1 to 10 do
        Begin
                InitBit;
                {8 High DatenBit ausgeben}
                For Var2 := 1 to 8 do
                    Begin
                        {High ausgeben}
                        SetFillPattern (Weiss, White);
                        FillPoly (SizeOf (Pentagon) div
                        SizeOf(PointType), Pentagon);
                    Delay (HighLength);
                        SetFillPattern (Schwarz, Black);
                        FillPoly    (SizeOf    (Pentagon)    div
SizeOf(PointType), Pentagon);
                      Delay (WaitLength);
                     End;
         { etwas Warten bis das nächste Datenbyte ausgegeben wird }
            Delay (InterByteTime);
            InitBit;
                {8 Low DatenBit ausgeben}
                For Var2 := 1 to 8 do
                    Begin
                        {Low ausgeben}
                        SetFillPattern (Weiss, White);
                        FillPoly    (SizeOf    (Pentagon)    div
SizeOf(PointType), Pentagon);
                      Delay (LowLength);
                        SetFillPattern (Schwarz, Black);
                      FillPoly     (SizeOf     (Pentagon)     div
SizeOf(PointType), Pentagon);
                      Delay (WaitLength);
                      End;
                {Warten bis nächstes DatenByte geschickt wird}
            Delay (InterByteTime);
        End;
         { Bildschirm wieder dunkel}
    readln;
    CloseGraph;

End._
```

Dazu erst mit längeren Delayzeiten testen und dann immer weiter vermindern, bis die Daten nicht mehr richtig übertragen werden. Fragen an den Autor Ing. H. Bärnthaler können via E-Mail gestellt werden (hbaernthaler@mail.carinthia.co.at).

990026