OCTOBER 2000
$2.95

ELEKTOR ELECTRONICS

www.elektor-electronics.co.uk

THE ELECTRONICS & COMPUTER MAGAZINE

GBDSO
GameBoy Digital Sampling Oscilloscope

S/PDIF Output for MP3 Players

Versatile Centronics/ I²C Interface

PCI-Hosted Measurement Cards

OSCAR MP3 Player Part 2

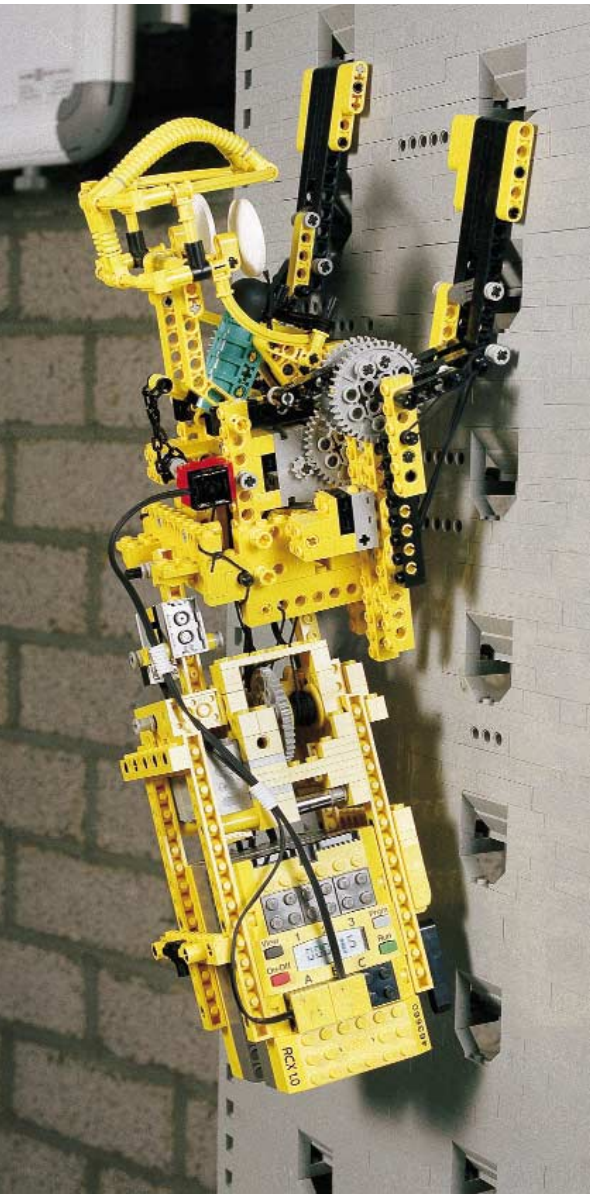
Software MIDI Synthesizers

9 770268 451081 10

Lego Robotics Invention System (4)

part 4: chatting robots

By Hans Steemann and Luc Lemmens



The Robotics Invention System is an outstanding choice for building advanced robots. Various types of sensors can be read out using the RCX module, and three motors or other types of actuators can be driven by the pulse-width modulated outputs. However, you may sometimes want to make a model that needs more power than a single RCX module can provide. Under the motto 'strength in unity', it turns out that RCX modules are able to work co-operatively.

After experimenting with the Robotics Invention System for a while, you will notice that the Lego system is both powerful and versatile. Still, it has its limits. Very complex robots may prove to need more active elements than can be driven or read out by a single RCX block. However, the system is flexible enough to be able to encompass additional possibilities, by using the combined forces of several modules. The demonstration robot shown in the photograph at the head of this article is a clear example of this. No less than three RCX blocks work together in a single model, in order to provided the desired functionality. One RCX module is built into the transport system, while the other two modules work together within the robot on the basis of clearly divided responsibilities. All three modules communicate with each other regarding the progress of their activities. This makes it possible to use a specific

RCX for each separate function, with control being passed to the next RCX block on completion of the task. Of course, two RCX blocks can also carry out tasks independently of each other, using messages to keep track of each other's processes. Even for the novice robot builder, it is not especially difficult to utilise the communication functions of the system within the standard development environment.

Communications

The robots use the infrared transceiver that is incorporated in each RIS for communicating with each other. Codes that are 8 bits wide (with numbers between 0 and 255) can be exchanged using this interface. The transmitting robot can send a message, which will be received by all robots that are within the reception range. Whether the module that receives the signal actually does anything with it depends on the software.

The distance over which the modules can communicate with each other depends on the settings that are made using the configuration screen. If the 'long' option is chosen in this screen, the maximum power level is used to send the command. Naturally, the distance that can be covered is a maximum in this case. This configuration option is comparable to the switch on the infrared transmitter that is connected to the PC. This switch relates only to the output power of the transmitter connected to the PC. The transmitter that is used by the RCX can only be configured using software. The drawback of high transmission power is that the signals may cause interference in other systems that use RCX modules. In addition, high transmission power also causes the battery to be used up more quickly, particularly if the robots exchange a lot of messages. You should therefore choose the setting that best fits the desired setup.

Let's start with message applications within the Lego development environment.

Figure 1. The icons belonging to the message functions in the Lego software.

RCX-message sensor watcher

RCX-message sensor watcher is an option that is associated with the standard commands. This blue function block represents a function that is integrated in the RCX, and that can best be compared to a normal sensor. The only difference is that it does not have to be separately connected.

This function continuously checks whether a message is being received. It causes a subroutine to be executed whenever a message is received from another RCX. The command

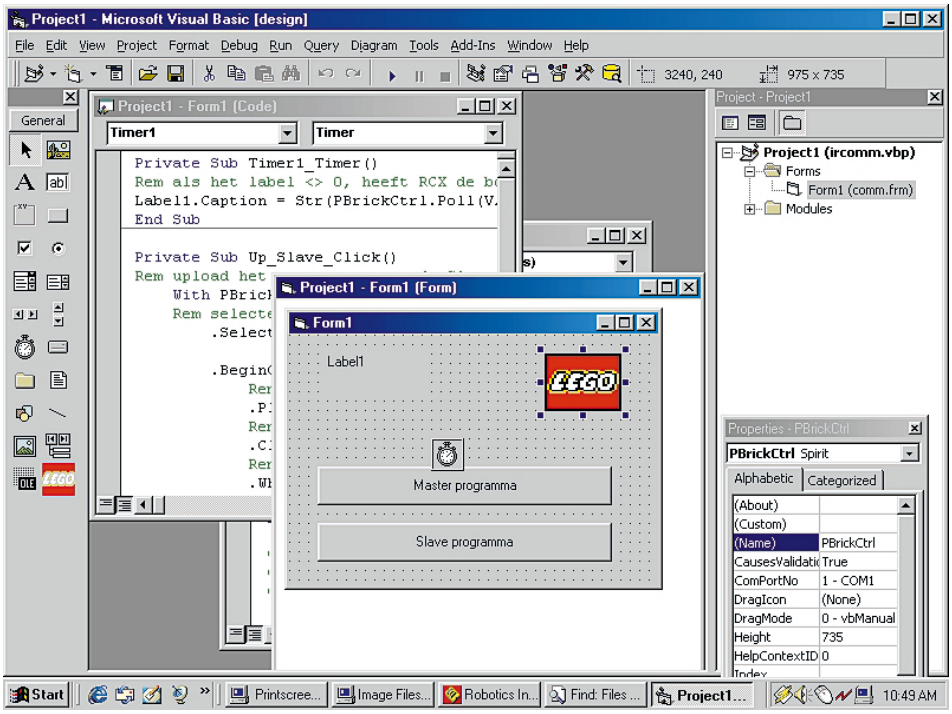


Figure 2. This form can be combined with the Visual BASIC program. The two buttons allow the program to be activated in the master and the slave. The result from Label1 appears at the upper right. Note that the Lego ActiveX module is linked in.

belonging to this function block can be used to specify the messages to which the RCX must respond. Click on the numbers, and enter the lowest and highest message values. The linked stack (the Lego term for a subroutine) will be executed when a message within the message value range is received from another RCX. A message may have a value between 0 and 255. If several RCX modules are used, each module can be assigned its own range. A common number can be used for broadcasting (common messages).

Reset Message
In order to prepare a subroutine to receive messages, it is necessary to first activate the 'Reset Message' block. As long as this has not been done, the RCX will not react to the commands.

Send to RCX
The 'Send to RCX' block allows the RCX to send an infrared message to another RCX. Needless to say, at least two RCX blocks must be used in a project before you can make use of the communications options. When software is being sent from the PC to an RCX, only one RCX at a time may be on!

Table I. The Poll function.

```
Poll(Source, Number)
```

Source and Number are used to indicate where the Poll function has to look in the RCX.

Source	Number	Function
0	0...31	Variables 0...31
1	0...3	Timer 0...3
2	-	-
3	0,1,2	Motor status
		Bit 7: On/Off 1/0
		Bit 6: Brake/Float 1/0
		Bit 5: Output no. HiBit
		Bit 4: Output no. LoBit
		Bit 3: Direction CW/CCW 1/0
		Bit 2: Power Level most significant bit
		Bit 1: Power Level
		Bit 0: Powel Level least significant bit
4	-	-
5	0,1	No function with RCX
6	0,1	No function with RCX
7	2	No function with RCX
8	-	RCX program number. Number actual program in use
9	0,1,2	SensorValue, value measured at input,
		depends on actual mode of operation
10	0,1,2	SensorType, tells what type of sensor the input is set-up for
11	0,1,2	SensorMode, tells what type of sensor the input is set-up for
12	0,1,2	SensorRAW i.e. the analogue value measured at the input
13	0,1,2	Sensor Boolean, returns the Bololean state of the input
14	0	RCX Watch. Integer where MSB=hours and LSB=minutes
15	0	Returns the PBMessage stored internally in the RCX
16	-	No function with RCX

The result of a Poll instruction that requests data or status is a 16-bit signed integer.

Examples of using Poll:

```
Label1.Caption = PBrickCtrrl.Poll 0, 7
```

Label1 takes on the value of Variable 7 in the RCX.

```
Label2.Caption = PBrickCtrl.Poll 8
```

Label2 takes on the value of the program being executed by the RCX.

More possibilities with Visual BASIC

Naturally, the features that support communications between RCX blocks are also available in Spirit.OCX, so we can also utilise them in high-level languages such as Visual BASIC. With such languages, it is also possible to have the PC actively participate in the data traffic. The role of the PC does not have to be limited to generating code for an RCX and sending out program blocks. As we have seen in previous instalments, the PC can also read out RCX registers (to query the battery voltage, for example), and it can even request the data from an RCX data logging session. Unfortunately, though, the PC cannot send and receive messages as a sort of pseudo-RCX. Spirit.OCX does not have any equivalent for the message register that is present in the RCX module. All in all, this means that a PC cannot distinguish between RCX blocks, which means that it can only communicate with a system that contains only one RCX block. Also, only one RCX module may be active whenever program blocks are being transferred from the PC.

Let's first have a brief look at how we can use a high-level language to develop programs that allow RCX modules to communicate with each other. After this, we will discuss

how a PC and an RCX can 'talk' to each other.

From RCX to RCX

Three commands in Spirit.OCX are important with regard to exchanging messages between RCX modules. Two of these (ClearPBMessage and SendPBMessage) have equivalents in the Lego programming environment, and the universal Poll command is used to check for successful reception.

Poll is an exceptionally powerful and versatile routine, which can monitor numerous events in and around the RCX block. **Table 1** shows how it can be used. For example, it is possible to use Poll to examine registers, to control timers, and to monitor motor operation. It can also be used is to see if messages have been received. A simple BASIC program that you can use for experimenting is shown in **Listing 1**.
Actually, this program incorporates two different examples, but in practice, they can almost never work together.

The program consists of two main routines. The subroutine 'Private Sub Up_Master_Click()' is a task that is placed in program slot 0 of an RCX block (RCX program 1).
After this, there has to be an RCX that can receive the message. The routine Private Sub Up_Slave_Click() is provided in Visual BASIC for this purpose. It places the code for receiving a message in the second RCX block, and it utilises slot 3 (SelectPrgm 3, so it can be activated as program 4 using the grey Program button on the RCX. This is a straightforward routine. When it is first activated, it causes the RCX to make a sound (PlaySystemSound SWEEP_DOWN_SOUND), following which it erases the register that may hold a message (ClearPBMessage) and then waits for the reception of a message. If a message is received, the same little tune is played again.
This is naturally only a very simple example, but it can be easily extended to a more meaningful application. In this example, two tasks are carried out in sequence. However, since the RCX can perform several tasks simultaneously, it is also possible to have message monitoring take place in a separate task, in parallel with the main task. The structure of the program will naturally have to take this into account. **Listing 2** shows the approach that must be used.

Finally, it is naturally also possible to wait for a message, and then to start a particular task according to the content of the message.

Listing 1.

```
Private Sub Timer1_Timer()
Rem if label <> 0, RCX has sent the message
Label1.Caption = Str(PBrickCtrl.Poll(VAR, 0))
End Sub

Private Sub Up_Slave_Click()
Rem upload the program for the Slave
    With PBrickCtrl
    Rem select program slot 4
        .SelectPrgm 3

        .BeginOfTask MAIN
            Rem let's hear the program starting
            .PlaySystemSound SWEEP_DOWN_SOUND
            Rem just to make sure, clear all messages
            .ClearPBMessage
            Rem and wait for message from another RCX
            .While PBMESS, 0, EQ, CON, 0
                Rem 50 ms delay
                .Wait CON, MS_50
            .EndWhile
            Rem message received!
            .PlaySystemSound SWEEP_DOWN_SOUND
        .EndOfTask
    End With

End Sub

Private Sub Up_master_Click()
With PBrickCtrl
    .SelectPrgm 0
    .BeginOfTask MAIN
        .ClearPBMessage
        Rem just to be safe, make var0 zero
        .SetVar 0, CON, 0
        Rem send message no. 11
        .SendPBMessage CON, 11
        Rem make variable 0 equal to 11
        Rem PC detects 'message sent' by reading timer
        .SetVar 0, CON, 11
    .EndOfTask
End With
End Sub

Private Sub Form_Load()
    PBrickCtrl.InitComm
End Sub
```

Note: this program utilises the RCXDATA.BAS module, in which the constants (MAIN, EQ, CON etc.) are declared and initialised. Make sure that this module is linked in!

Listing 1. The basic program that can be used to analyse the sending of messages.

Listing 2.

```
:REM structure of a program
  with parallel tasks

PBrickCtrl.BeginOfTask MAIN

  PBrickCtrl.StartTask 1

  PBrickCtrl.StartTask 2

PbrickCtrl.EndOfTask


:REM definition of tasks within
  MAIN

PBrickCtrl.BeginOfTask 1
          .........
          .........
          .........
PBrickCtrl.EndOfTask

PBrickCtrl.BeginOfTask 2
          .........
          .........
          .........
PBrickCtrl.EndOfTask
```

Listing 2. If the program is set up slightly differently, the monitoring of message traffic can be carried out in parallel with the main task

Between PC and RCX

We have already mentioned that the PC cannot participate in the message traffic that can take place between RCX blocks. The actual problem is primarily that an RCX cannot itself send a message to the PC. However, it is possible to use a trick to allow the PC to periodically 'look at' the RCX to see if it has anything to report. Note that this trick works only with a single RCX; it cannot be used in a system in which several RCX modules are active at the same time.

The trick is based on using an auxiliary variable in the RCX (VAR 0 in this example). This variable has the value '0' after the routine is started. After a message has been sent, the variable takes on the same value as the message. This is done by the program line 'SetVar 0, CON, 11'. The number '11' corresponds to the content of the message, since the instruction SendPBMessage CON, 11 is defined in the previous line (which is just a coincidence; other values are also possible in this application).

The PC uses a timer to check the value of VAR 0 once per second. The first few lines of the program show how this works. The routine Private Sub Timer1_Timer() is used for this purpose. With this arrangement, the PC uses the Poll command to request the content of VAR 0 at one-second intervals via PBrickCtrl.Poll (VAR, 0). This is controlled by Timer1, which has a period of one second. The content of the text string obtained in this manner is assigned to Label1, and then displayed on the screen using the form.

Many readers may wonder what this is all good for. The answer is that the PC cannot directly receive a message from the RCX. However, the technique illustrated in this example, in which the content of the message is assigned to an auxiliary variable, makes it possible to check whether something has happened inside the RCX or has been handled by the RCX. This function can be used to send a new task to the RCX after it has carried out a particular task, or to start a new session after the RCX has been read out, for example.

Now you can probably understand why we earlier remarked that these two examples are not compatible with each other. This is because whenever the PC requests the value of the auxiliary variable, it would not be possible for two or more RCX blocks to know which of them should send its Variable 0 value. Every block that receives a command from the PC will send back its own VAR 0 value, which means that no reliable result can be obtained in the PC.

Both of these examples are thus intended purely to illustrate how various things work in the interplay between the PC and the RCX or between several RCX modules. However, it should not be difficult to apply these principles to an actual robot environment.

(000040-4)

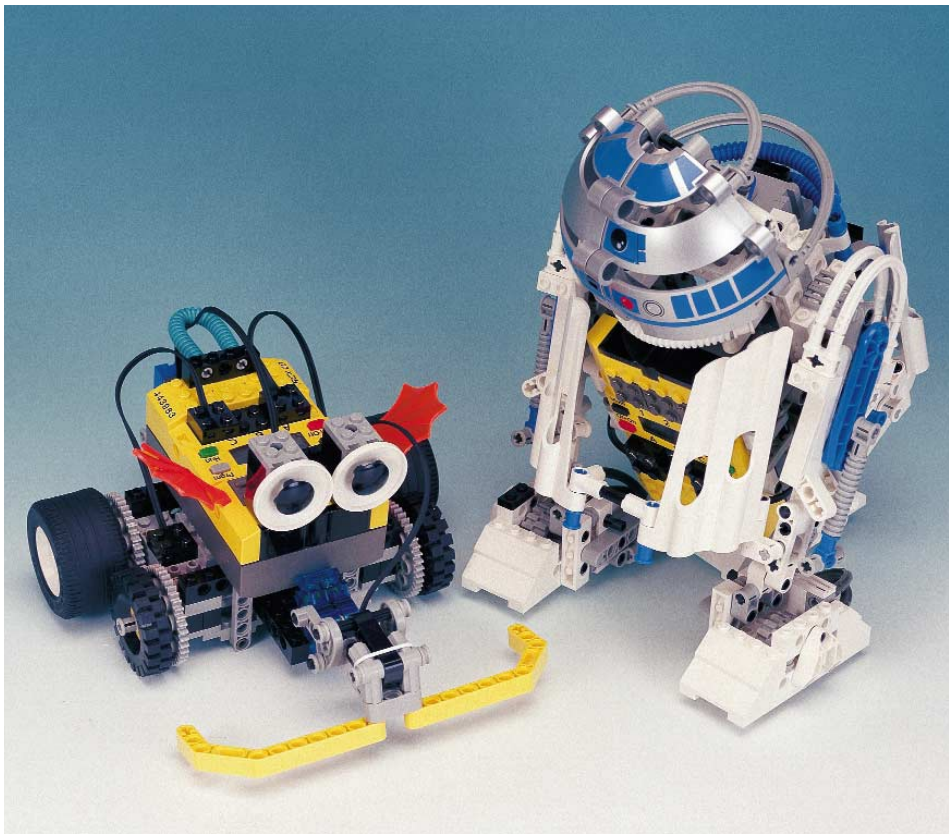


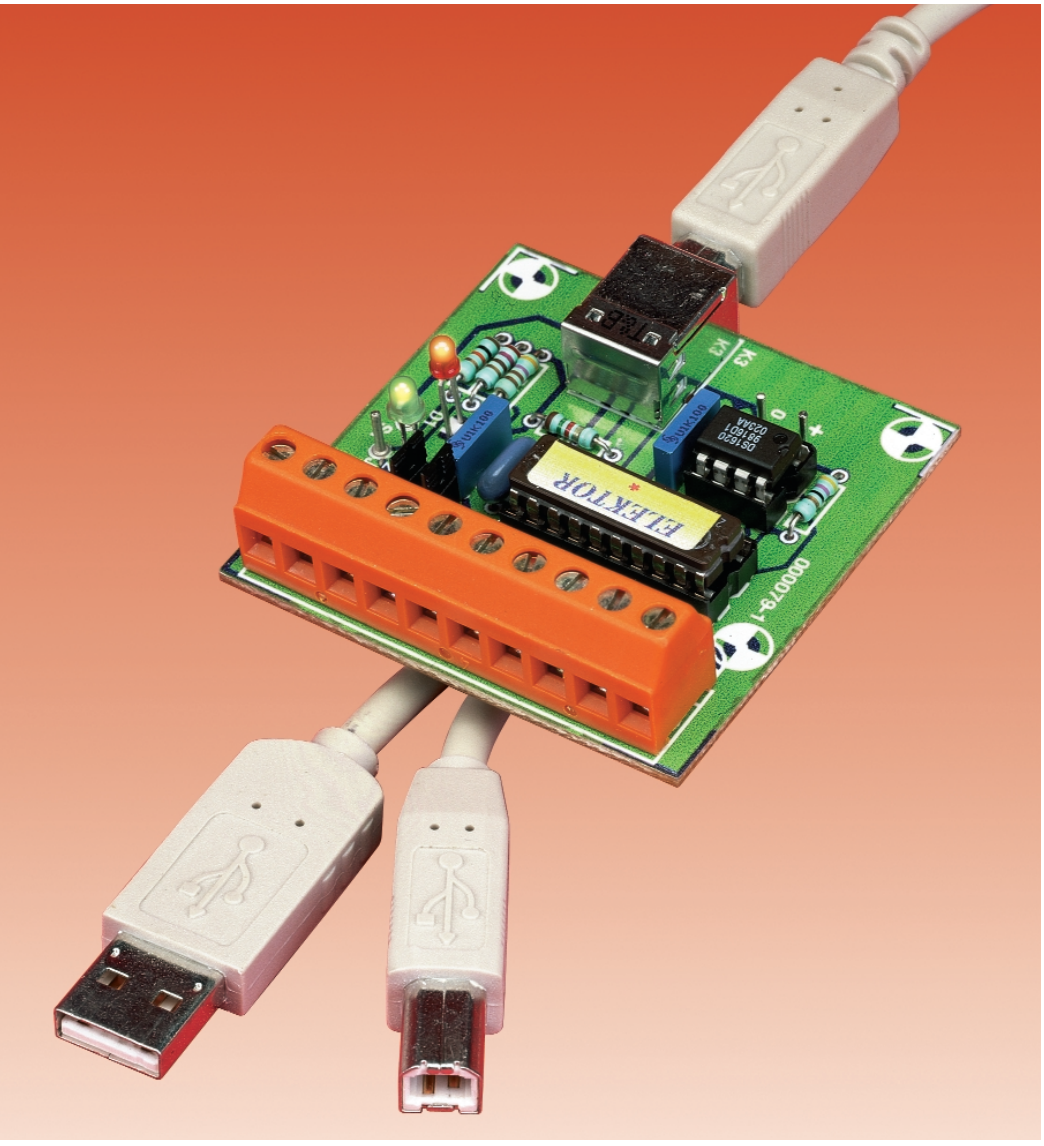
Figure 3. Thanks to the PBMessages that the RCX blocks can exchange with each other, the robots can 'chat' with each other. In this example, R2S2 is communicating with the robot that will be featured in the next instalment.

The Universal Serial Bus (USB)

Part 2: Transfers and Drivers

By B. Kainka

In the first part of this article we looked at the electrical characteristics of the USB standard. In this part we investigate the software and protocols used with this interface. Also included is a practical programming example.



When any new device is first plugged in to the USB connector, it will signal its presence to the hub by pulling up one of the two data wires of the USB cable. Next comes the following sequence of events:

- The hub indicates to the host that a new device has been connected to the bus.
- The host will ask the hub which port the device will use.
- Now the host knows which port the device will use, it enables the port and issues a bus reset command to the hub.
- The hub generates a reset signal on the bus by pulling both data lines D+ and D- low for 10 ms. After reset, the device is ready and will respond using the default address 0.
- Until the device is assigned its own bus address it will use the default address. The host will read the first bytes of the device descriptor to determine the length of the data packet in order to assign a length to the default pipe.
- The host will allocate a bus address to the device.
- The host will now read all the configuration information from the device using its newly allocated bus address.
- The host assigns one of the possible configurations to the device. The current consumption of the device

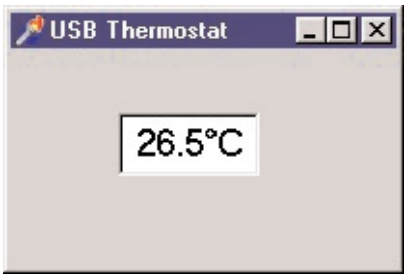


Figure 1. Screen shot of the USB Thermostat.

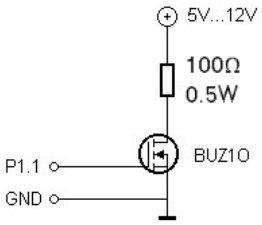


Figure 2. Mini hardware driver for the Heater Controller.

must not be more than that defined in its configuration descriptor. The device is now ready for use.

The microcontroller in the USB device responds to each message from the host. A hardware interrupt will indicate that data has been received and is available in FIFO 0. The data is analysed to determine what type of message it is. Key bytes in the data packet request the device descriptor. The microcontroller will respond by reading the requested information from its ROM and writing it to its output FIFO. From there it will be sent back to the master. The first descriptor sent will be the device descriptor which has a length of 18 bytes. The host will initially take the first 8 bytes. After the host has assigned the device address, it will read the rest of the descriptor. The 18 data bytes will be spread over three frames to enable it to fit into the 8 byte FIFO.

Table 1 shows the Device Descriptor of the USB Thermometer from Cypress, which was featured in part 1 of this series. Each device requires a Vendor-ID and a device ID. Using these numbers, the device is recognised and the corresponding driver software is loaded from the host to the device. The Vendor-ID is assigned under license to a device manufacturer by the USB Organisation. Those who intend to make a serious business of developing USB devices must also arrange for their own vendor ID.

For the hobbyist it is in theory possible to unofficially make up your own VID. This will entail writing your own driver and a .INF file to configure the device. If we carried on down this road you can probably see that eventually the whole 'Plug and Play' concept would be undermined by device manufacturers assigning themselves identical VID's leading to all sorts of confusion.

For our purposes, the best (and perfectly legal) solution is to use the vendor ID of the semiconductor manufacturer corresponding to the USB chip in the device. A device incorporating the USB Soundchip UDA1321 would use the VID of its manufacturer i.e. Philips. For the EZ USB Chip from Anchor you will use the Anchor driver and your own software in the controller.

Types of USB Transfer

There are currently four completely different methods of communicating between a USB device and a PC:

- *Control-Transfer*
Used for control signals to the device, they have a high priority and incorporate automatic error protection. Data rate is high and up to 64 Bytes per request can be sent.
- *Interrupt-Transfer*
Devices such as keyboards and mice would generally use this method. These devices send small amounts of data periodically to the PC. The name of this transfer would suggest that a hardware interrupt is responsible for the data flow but this is not the case, nor is it possible in a single master system. Actually, the system looks for new data every 10 ms or so. The amount of data sent using this transfer would typically be up to eight bytes at a time.
- *Bulk-Transfer*
Used to transfer large amounts of information that requires error protection but is not time critical. Typical applications for this method of data transfer would be passing data to a printer or receiving data from a scanner. The actual data rate is adapted to the capacity of the USB system. This method has a low priority.
- *Isochronous-Transfer*
Devices transferring large amounts of information at a defined data rate, e.g. digitised audio to a soundcard, would use this transfer method. A defined data rate is guaranteed but no error protection is included. With these devices, occasional errors are less important than gaps in the data.

Low-speed devices support control and Interrupt Transfer modes while full speed devices support all four modes. For use in the area of control, measurement and regulation the control transfer method is particularly useful, it offers high data rate together with data error protection. The communication protocol is comparatively easy to implement for this application so for our purposes here we will use the control transfer method for the *Elektor Electronics* USB Interface.

Control Transfers in Delphi

Software drivers are used to control USB devices. Conventional methods of controlling PC peripherals via their hardware port addresses using DOS are no longer possible with the USB interface. In order to control a device it is necessary to study the documentation of the driver function.

Drivers are in principle treated as data files. They are opened, you can read or write data to them and then they are closed. Programming is very akin to accessing data or passing data to and from the RS232 interface.

For example, writing to a USB printer would be performed with WriteFile while reading data from a USB scanner would be performed with ReadFile. In this example of

USB Controllers

When starting in USB development, it is a good idea to choose a USB family, i.e., a microcontroller and/or programming tools (Assembler or C compiler), which you are already familiar with and which is readily available. Equally important is the amount and quality of the supplied documentation. Some companies produce Starter Kits that come complete with volumes of software documentation

Philips Semiconductors

http://www.semiconductor.philips.com

Produce two interesting full speed-USB-Transceivers. These are similar to a UART and can be easily used in existing microcontroller systems or can be interfaced to most microcontrollers.

The **PDIUSBD11** with only 16 pins uses an I²C-Bus-Interface for easy connection to a microcontroller. The chip contains 4 end points and two 8-bit FIFOS for In- and Out-transfers.

- Full speed-USB
- I²C Bus Interface up to 1 Mbit/s
- 12 MHz crystal, multiplied to 48 MHz internally using a PLL.
- 1 Control, 6 generic endpoints (8 Bit Max Packet Size)
- Package outline: DIP-16 and SO-16

The **PDIUSBD12** is a full-speed USB-Transceiver with a parallel interface compatible with an 8051 Controller. It has three endpoints, which can hold from 1 to 128 bytes. The package has 28 pins but is unfortunately only available for SMD mounting.

- Full-speed USB
- Parallel Data bus with data rate up to 2 Mbytes/s
- DMA support
- Up to 1MByte/s in Bulk Mode
- Up to 1Mbit/s in Isochronous Mode
- 1 Control, 4 Generic (Dependent on Mode)
- 320 Bytes (in total) deep FIFO
- On board 3.3 V voltage regulator
- Package outline: SO28 and TSSOP28

AnchorChips

www.anchorchips.com

The **AN2131** is a versatile USB-Controller with a 8051 compatible core. The in-built USB Machine is capable of self-enumeration. With this controller you can build a USB device without writing a single line of code. The chip announces itself as an Anchor Default Interface and has all the functions and further program code for loading and starting. This simplifies the development in the controller. Program download via USB means an end to the annoying burn/wipe EPROM cycle normally associated with program development.

- Full speed USB
- 12 MHz crystal, multiplied to 48 MHz internally using a PLL
- 8051 Core.
- Programme Download using USB.
- I²C interface
- 8 K internal RAM, expandable to 64 K externally.
- Package outline: PQFP-44, PQFP-80

Infineon

http://www.infineon.de

Produces the C541U, an 8051 compatible full speed Controller. The primary application for this controller is intended to be telecommunications e.g. ISDN modems. The controller has external RAM and program memory that simplifies development.

- Full speed USB, selectable low speed
- 12 MHz crystal
- 8051 core (C500)
- 5 V supply with internal 3.3V voltage regulator.
- SSC (SPI) interface
- 8 K OTP
- external up to 64 K RAM and ROM
- Package outline: PLCC 44, PSDIP 52

Cypress

www.cypress.com

Cypress supply the low cost, low speed **CY7C63000** Family controller, mainly intended for HID's. The CY7C63001 is used in the Elektor-Electronics Interface. One advantage for developers is the PDIP package outline. Cypress provides documentation for different 'reference designs' which makes development easier.

- Low speed USB
- 1 Control endpoint, 1 Interrupt endpoint. (per 8 Bytes)
- 8 Bit RISC-Processor
- 6 MHz crystal, internal clock 12 MHz.
- 128 bytes of RAM
- 2K OTP (CY7C63000A)
- 4K OTP (CY7C63001A)
- Programmable current output port with a 4 Bit D/A
- Package outline: PDIP 20, SOIC 20

Motorola

http://sps.motorola.com

Friends of the HC05-Family are not left out. Motorola produce the 68HC705JB3 low-speed USB Controller with 2 Interrupt endpoints. The 68HC705JB4 also has an A/D converter.

- Low speed USB **68HC705JB3**
- HC05 core.
- 2560 Bytes User ROM
- 144 Bytes RAM
- Low Speed 1.5Mbps
- 1 Control, 2 Interrupt endpoints
- Package outline: PDIP-20/28 and SOIC

- Low speed USB **68HC705JB4**
- HC05 core
- 3584 Bytes ROM
- 176 Bytes RAM
- 1 Control, 2 Interrupt endpoints
- 6 channel, 8 Bit A/D converter
- Package outline: PDIP 28 and SOIC 28

Microchip

http://www.microchip.com

If you prefer PIC controllers, Microchip produce these low-speed USB controllers. They include an A/D converter and a serial interface UART.

PIC16C745
- Low speed USB with 6 endpoints
- 8 Bit A/D converter, 5 Inputs
- 8k Program Memory
- UART
- Package outline: PDIP 20

PIC16C765
- Low speed USB with 6 endpoints
- 8 Bit A/D converter, 8 inputs
- 8k Program Memory
- UART
- Package outline: PDIP 40

```
unit USB1;

interface

uses Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ExtCtrls, StdCtrls;

type
  TForm1 = class(TForm)
    Timer1: TTimer;
    Edit1: TEdit;
    procedure FormCreate(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
  end;

type _lIn  = record
    bFunction : Byte;
    bValue1 : Byte;
    bValue2 : Byte;
    bValue3 : Byte;
end;

type _lOut  = record
    bAck : Byte;
    bValue1 : Byte;
    bValue2 : Byte;
    bValue3 : Byte;
end;


var
  Form1: TForm1;
  Temp : Real;

implementation

{$R *.DFM}

procedure Thermo;
var lIn: _lIn;
    lOut: _lOut;
    DeviceHandle: THandle;
    TemplateHandle: THandle;
    nBytes: DWord;
    bresult: Boolean;
begin
  DeviceHandle := CreateFile
('\\.\Thermometer_0',Generic_write,File_Share_write,nil,open_existing,0,TemplateHandle);
  lIn.bFunction := 11;   {Read Thermometer}
  bResult := DeviceIoControl(DeviceHandle,$04,@lIn,sizeof(lIn),@lOut,sizeof (lOut),nBytes,nil);
  Temp := lOut.bValue1/2;
  lIn.bFunction := 23;   {Write RAM}
  lIn.bValue1 := 47;     {Adr. 47 = Port 1}
  if Temp > 37 then   lIn.bValue2 := 0 else lIn.bValue2 := 15;
  bResult := DeviceIoControl(DeviceHandle,$04,@lIn,sizeof(lIn),@lOut,sizeof (lOut),nBytes,nil);
  CloseHandle (DeviceHandle);
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  Timer1.Interval := 100;
  Timer1.Enabled := true;
end;

procedure TForm1.Timer1Timer(Sender: TObject);
begin
  Thermo;
  Edit1.Text := FloatToStrF(Temp,ffGeneral,3,2) + '°C';
end;

end.
```

Listing 1. Control access in Delphi.

Table I. Format of the Device Descriptors

Fieldname	Length(in bytes), Description	Example
Blength	1, length of the Descriptor in Bytes	12h
BDescriptorType	1, Descriptor type (01h=Device Descriptor)	01h
BcdUSB	2, USB Version (V. 1.0)	00h, 01h
BDeviceClass	1, Class Code	00h
BDeviceSubClass	1, Subclass Code	00h
BDeviceProtoco	1, Protocol Code	00h
bMaxPacketSize0	1, EP0-FIFO size	08h
IdVendor	2, Vendor ID (04B4h=Cypress)	B4h, 04h
IdProduct	2, Product ID (02=Thermometer)	02h, 00h
BcdDevice	2, Version number (V.09)	09h, 00h
IManufacturer	1, String Index for "manufacturer"	01h
Product	1, String Index for "Product"	02h
ISerialNumber	1, String Index for "Serial number"	00h
BNumConfigurations	1, The number of possible configurations	01h

a printer, data will only be flowing in one direction so the USB bulk transfer method would be employed — control transfer would not be suitable here because it allocates time for data to flow in both directions.

For control transfer methods the Windows function DeviceIoControl is used. **Listing 1** shows the Delphi program that reads the temperature from the *Elektor Electronics* interface. The Windows Function CreateFile supplies a Handle to the device or rather its driver. The device name \\.*Thermometer_0* is defined in the driver. With the valid handle, it will access the temperature reading. Lastly the device will be closed with CloseHandle. All these three Windows functions are declared in the Delphi 4 Windows unit. For more precise information, see Windows.pas and further information is in the help file WIN32.HLP.

Each access with DeviceIoControl requires a control code that is used to call one of the possible driver functions. Additionally two buffers need to be initialised, one input buffer and one output buffer. Both buffers require a maximum length of four bytes. In the listing, these buffers are defined as byte records. The flow direction of the data in the buffers is not defined, and a driver could use both buffers to send data in the same direction. In our example 1In sends data to the USB controller, while 1Out receives data from the device.

To read the temperature, the value 11 (or 0Bhex) is sent as the first byte to the in buffer — this is done in the lIn.bFunction. The following positions bValue1 to bValue2 are used by the *Read Thermometer* function, not by the driver. If everything works correctly, the driver will return four bytes in lOut. The driver status indication is contained in bAck (1 indicates success). bValue1 contains the temperature, bValue2 the sign of the temperature value and bValue3 is the status of the pushbuttons.

This example program implements a thermostat to maintain a temperature of 37°C (ideal for incubating chicken eggs!). The output appears on the four output lines of Port 1 together with the green LED. While the Function 11 reads the temperature, Function 23 writes to a RAM address in the controller. Using Address 47, the *Elektor Electronics* firmware allows access to port 1. After each temperature measurement, a value of either 15 (all 1's) or 00 (all 0's) will be sent out depending on whether the temperature is too high or too low. To control the heater, a Power FET type BUZ10 can be used at the output to switch the current (**Figure 2**). For test purposes a 100-Ω resistor (physically in contact with the temperature sensor) will serve as the heating element. Using a supply voltage of 5 V it will be possible for the resistor to reach the selected temperature.

The complete Delphi Program can be downloaded from the author's homepage:

http://home.t-online.de/home/B.Kainka

The USB Driver

Developers of USB devices can write the necessary device driver or they can build a device which already has an existing class driver. This option is however not available for applications of control, measurement and regulation. The necessary prerequisites for developing a USB driver are C++ development environment and the Driver Development Kit (DDK) from Microsoft. This kit includes an example USB driver together with helpful additional information. Writing your own driver is however not a trivial exercise, so the more examples of driver code you can study, the better. Included along with the EZ-USB Starter kit from Anchor Chips is the complete source code for a versatile USB driver. Also worth a visit for a good introduction to USB is the website run by Craig Peacock (www.beyondlogic.org). He has also written a driver for the USB thermometer that can be used with the *Elektor Electronics* interface.

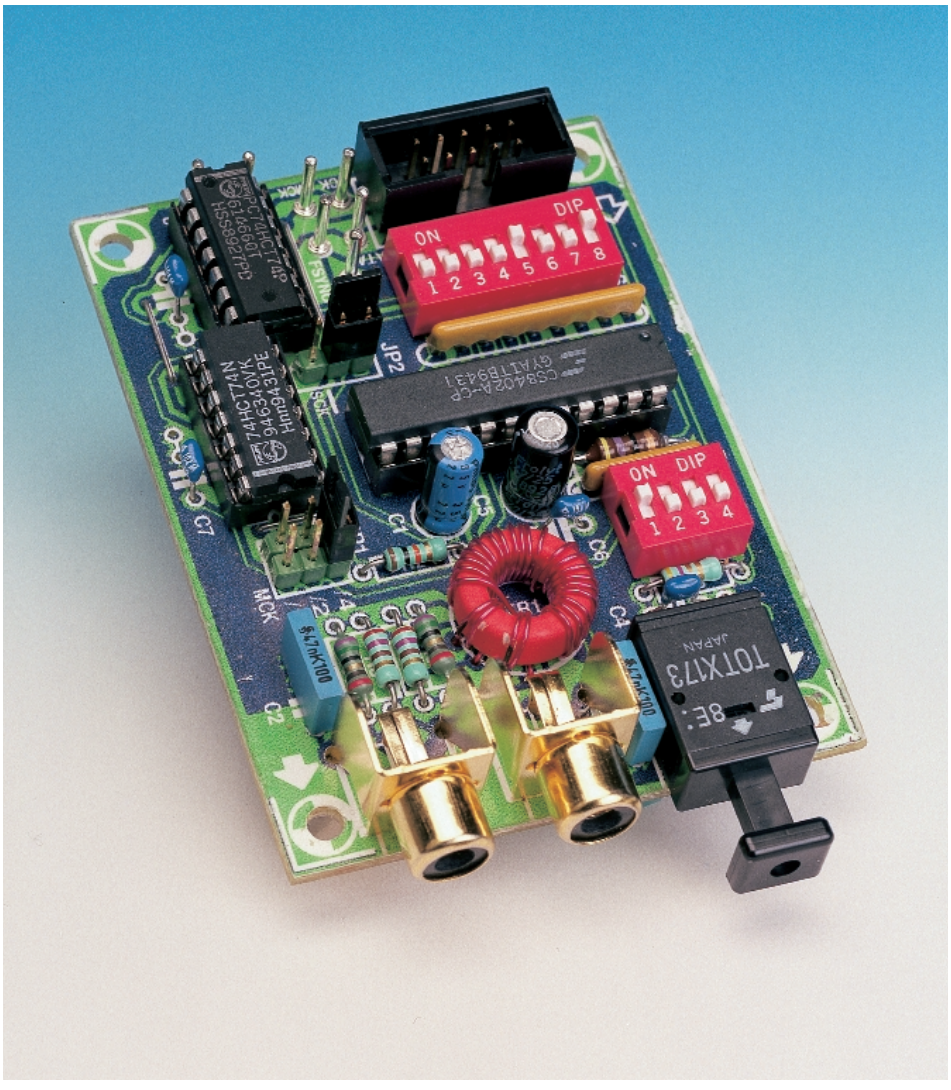
(000094-2)

S/PDIF Output

For OSCAR and other digital audio-equipment

Design by T. Giesberts

The standard MP3 CD player is not provided with an S/PDIF-output that would have enabled the transmission of digital audio signals to other equipment. With the aid of this circuit, designed around an IC from Crystal, an optical and coaxial S/PDIF signal is generated from a kind of I^2S signal in the OSCAR MP3-player.



Althoiugh the OSCAR MP3 player is fitted with analogue audio outputs that may be connected to other hi-fi equipment or the line input of a sound card, it would be easier and less lossy to transfer the signal in digital form to another piece of equipment with digital input, especially when recording tracks using, for instance, a DAT- or MD-recorder.

As standard, OSCAR does not possess an output that provides a digital audio signal in usable form. However, on the main circuit board is a connection that provides digital audio data in a 'raw' format. This signal must first be correctly encoded with additional information bits and then transmitted with the correct bi-phase modulation and amplitude.

The CS8402A from Crystal is an IC that can do all these tasks by itself, hence the name 'digital audio interface transmitter'. Around this IC is some circuitry that translates the 'raw' signal from OSCAR to S/PDIF.

In the first instance, this circuit was designed for use with OSCAR, however it is universally applicable because almost all the hardware modes of the CS8402A are available to be selected. Two DIP switches and two jumpers enable the circuit to support various formats.

The CS8402 has already appeared in various projects (such as the Sampling Rate Converter from October '96, 20-bit A/D Converter from December '96 and the S/PDIF Test Generator from July/August '99), so we will limit ourselves to the most important aspects. In any case, to enable a better insight into the overall functionality of the IC, **Figure 1** shows once again the block diagram of the CS8402A. Additional information can be found on the Cirrus Logic web site.

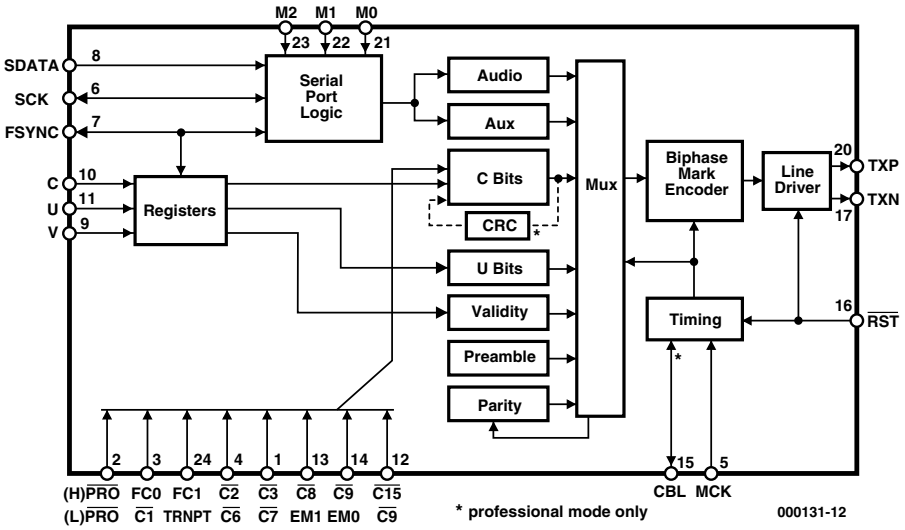


Figure 1. Internal architecture of the CS8402A.

The circuit

In **Figure 2** you can see the entire schematic for the converter circuit. As previously mentioned, the CS8402A takes care of the complete conversion from raw audio data to S/PDIF. Besides the TOSLINK module for the optical output, the only other active elements are two HCT logic ICs (dual D flip-flops).

The master clock signal MCK for the CS8402A arrives at pin 3 of K1 (or on the PCB pin of the same name). It is then routed via header JP1 to pin 5 of IC1. To make the circuit as universally applicable as possible, two divide-by-two circuits (IC2a and IC2b) are connected to pin 3 of K1. Normally (when transparent mode is not selected) the master clock MCK amounts to 128?Fs. Because of the presence of the divide-by-twos MCK may also take the values of 256?Fs or 512?Fs. This is selectable with Jumper 1 (JP1). The divisors /4, /2 and /1 can be seen next to JP1 on the PCB overlay. The latter connects the MCK input of the CS8402A

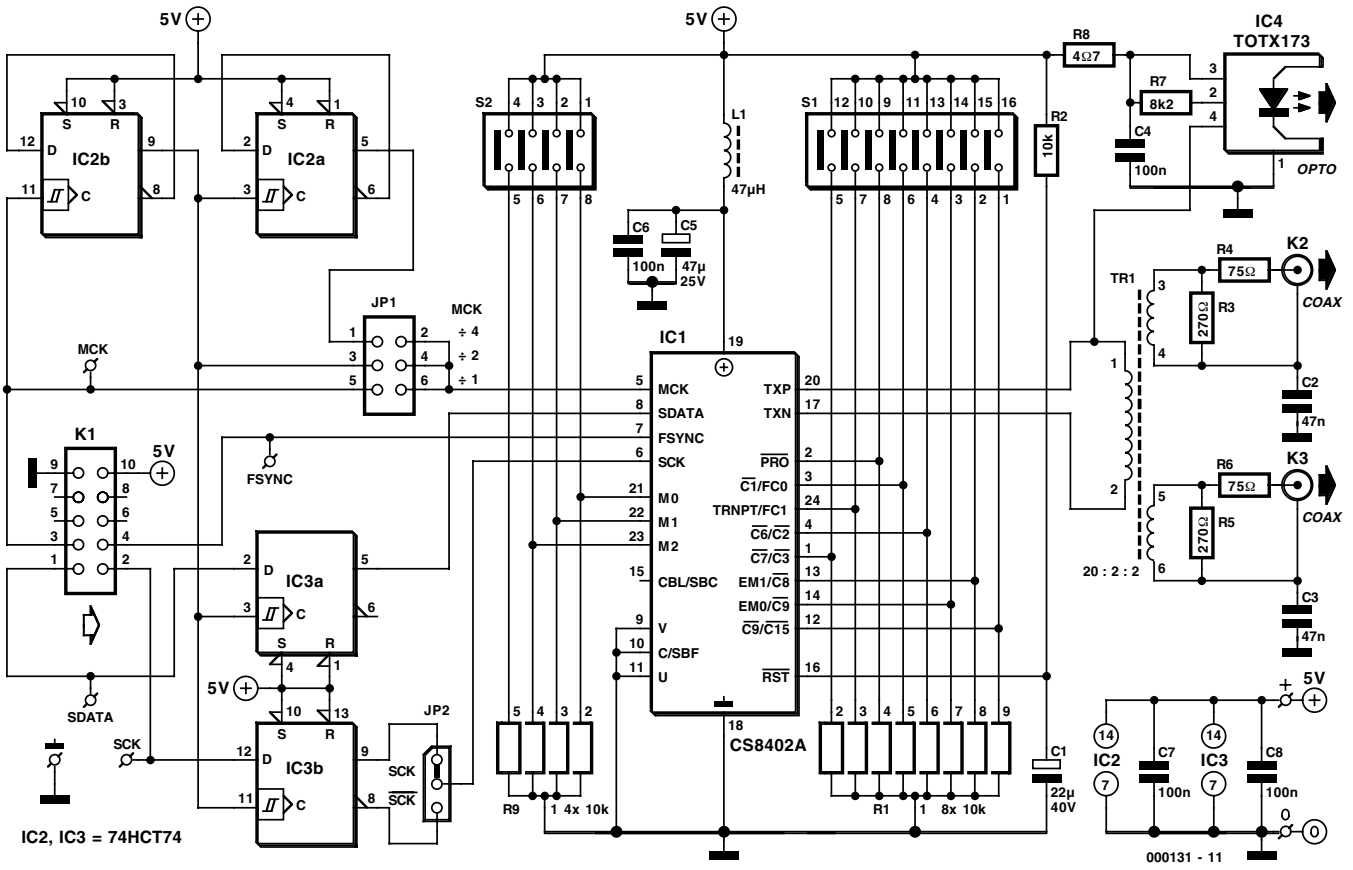


Figure 2. Complete circuit diagram of the S/PDIF-converter.

COMPONENTS LIST

Resistors:
R1 = 8 x 10kΩ resistor array
R2 = 10kΩ
R3,R5 = 270Ω
R4,R6 = 75Ω
R7 = 8kΩ2
R8 = 4Ω7
R9 = 4 x 10kΩ resistor array

Capacitors:
C1 = 22µF 40V radial
C2,C3 = 47nF
C4,C6,C7,C8 = 100nF ceramic
C5 = 47µF 25V radial

Inductors:
L1 = 47 µH choke

Semiconductors:
IC1 = CS8402A-CP (Crystal)
IC2,IC3 = 74HCT74
IC4 = TOTX173 (Toshiba)

Miscellaneous:
JP1 = 3-way pinheader + jumper
JP2 = 2 x 3-way pinheader + jumper
K1 = 10-way boxheader
K2,K3 = cinch socket, PCB mount (e.g., T-709G from Monacor/Monarch)
S1 = 8-way DIP switch
S2 = 4-way DIP switch
TR1 = ferrite ring core, Philips type TN13/7,5/5-3E25, primary 20 turns, secondary 2 x 2 turns, lacquered wire 0.5 mm dia.
PCB, order code **000131-1** (see Readers Services pages)

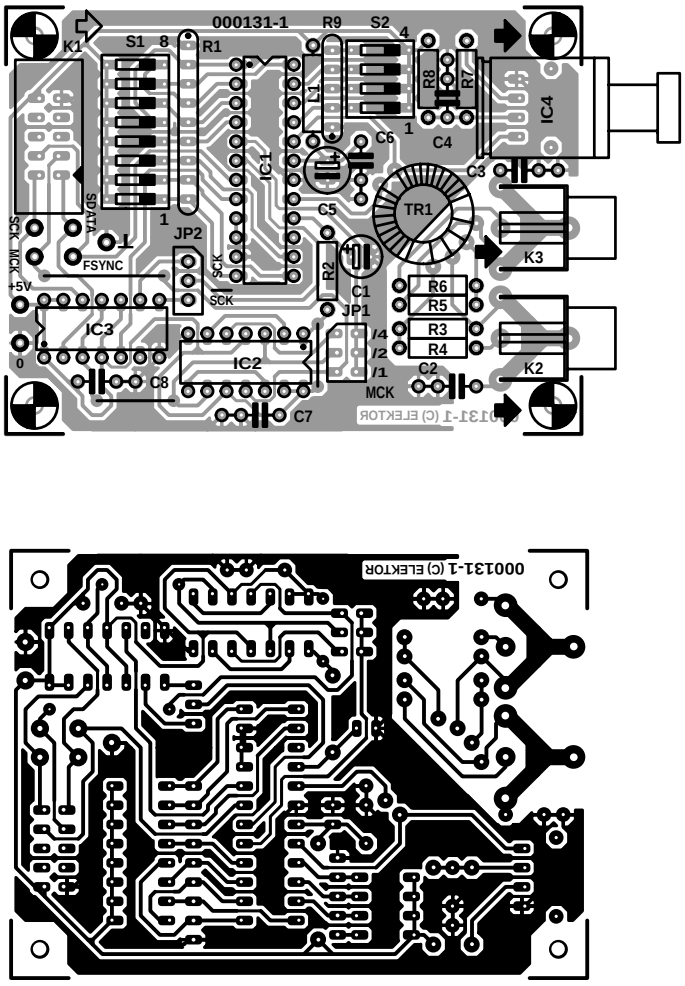


Figure 3. The single-sided PCB designed for the converter. Be sure to fit the three wire links!

directly to K1 and associated PCB pin.

Depending on the mode, data may be clocked on the rising or the falling edge of serial clock SCK. Consequently, an additional D flip-flop (IC3b) is provided to allow the possibility of inverting SCK. Jumper J2 selects the polarity of SCK. To ensure the correct relationship between data and SCK, the data is also clocked through a D flip-flop (IC3a). The divide-by-two MCK clock signal (output Q of IC2b) is used for this.

The decision to use HCT logic was a conscious one because it is possible that the input signals originate from 3-V logic (such as the MP3 CD-Player). The CS8402A interprets signals greater than 2 V as a logic high level. This arrangement avoids the need for a level shifter. The L/R clock signal FSYNC (frame sync) is connected directly, without level shifting, from K1 to pin 7 of IC1.

SDATA, SCK and FSYNC together form the audio serial port. Mode select pins M0, M1 and M2 define the format for this port. Seven different formats may be selected this way and the addition of Jumper 2 allows several more (refer to the sidebar for details). A quad DIP switch S2 and a quad resistor array R9 are used to select the mode of these 3 pins. Although the fourth switch and resistor remain unused, a quad package is used simply because it is easier to obtain.

The CS8402A is provided with eight pins that define the operation of the transmitter. These are selected with the aid of S1/R1. Every pin has two functions depending on whether professional- or consumer mode is selected (pin 2, S1-8). **Table 1** lists the preferred values for use with OSCAR. The functions shown for S1 apply to consumer mode.

The output of the CS8402A is symmetrical. The 10 V_{pp} drive signal for Tr1 for the coaxial outputs permits a relatively large primary/secondary turns ratio (10:1). A coaxial cable (transmission line) is normally terminated at both ends into its characteristic impedance. Reflections are avoided this way. An output signal of 1 V_{pp} is required on the secondary side in order to deliver 0.5 V_{pp} into 75 Ω. The large transformer ratio provides a lower output impedance and greater bandwidth (better coupling because the primary winding covers a large part of the toroidal core).

The application of a transformer for the coaxial outputs is motivated as follows: full electrical isolation between various devices will, to a large extend, avoid ground loops and other possible sources of trouble.

A second advantage of the symmetrical drive signal for the transformer is that by using a toroidal

core, the primary winding can be split easily into two equal halves. The terminals for the primary and secondary windings are now on opposite sides of the core and the connection between the two halves of the primary winding acts as a virtual ground point. This suppresses crosstalk (and improves electrical isolation at RF).

Resistors R3 and R5 damp the outputs of the secondary windings and ensure that the transformers present an resistive load to the CS8402A at all times. R4 and R5 mainly define the output impedance (75 Ω). Capacitors C2 and C3 provide RF grounding for the screen of the coax cables.

The circuitry around the TOSLINK module IC4 is of a conventional nature and consists mainly of power supply decoupling. R7 is required for the internal adjustment of the module.

Practical matters

A small circuit board was developed for the S/PDIF converter. It is shown in **Figure 3**. The construction will require little effort. Winding transformer Tr1 is probably the most difficult part.

The output transformer is wound with 0.5 mm diameter enamelled copper wire on a 13?5.5 mm toroidal core from Philips (TN13/7.5/5-3E25). The primary winding consists of 20 turns and the two secondary windings of two turns each (refer to the drawing of **Figure 4**). A 0.5-metre length of wire should be adequate.

The connection between the circuit and the audio data source is made with a ribbon cable plugged into K1 (for connection to OSCAR) or by using the PCB pins next to K1 and individual wires for other applications. With OSCAR, the ribbon cable connects via a 10-way insulation displacement connector (IDC) to pin-header JP6 (AUX) on the main circuit board. Pay careful attention to the correct orientation.

In the case of the MP3 player, the supply voltage arrives via the ribbon cable connected to K1. In the event that the connections are made using individual wires connected to the PCB pins next to K1, a separate 5 V power supply may be connected to the power supply pins next to IC3.

A few remarks before finishing. Unfortunately there was no space on the PCB next to S2 to provide a quick reference to the function of the individual DIP switches. S2-1 is the LSB M0, S2-2 is M1 and S2-3 is M2. Between R1 and C1 you may find enough room on the board to stick a small label with the default settings for the functions of S1.

(000131)

Internet address:
Cirrus Logic: http://www.cirrus.com

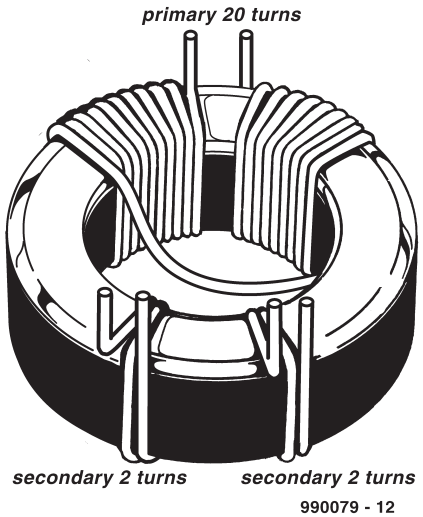


Figure 4. Showing how the transformer is wound.

Main settings on S1 & S2.

Audio Port modes

S2-3	S2-2	S2-1	
M2	M1	M0	Format
0	0	0	FSYNC & SCK output
0	0	1	Left/right, 16-24 bits
0	1	0	Word sync, 16-24 bits
0	1	1	Reserved
1	0	0	Left/right, I²S compatible
1	0	1	LSB justified, 16 bits
1	1	0	LSB justified, 18 bits
1	1	1	MSB last, 16-24 bits

Sample rates in professional mode

S1-8	S1-4	S1-5
PRO	C6	C7
0	0	0 not defined
0	0	1 48 kHz
0	1	0 44.1 kHz
0	1	1 32 kHz

Sample rates in consumer-mode

S1-8	S1-7	S1-6
PRO	FC1	FC0
1	0	0 44.1 kHz
1	0	1 48 kHz
1	1	0 32 kHz
1	1	1 44.1 kHz, CD-mode

Category code

S1-8	S1-2	S1-3
PRO	C8	C9
1	0	0 general format
1	0	1 PCM encoder/decoder
1	1	0 CD
1	1	1 DAT

('1'= switch closed, '0'=switch open)

Table 1. Default settings for OSCAR.

S1	Pos.	Signal description
-1 off	C15\	generation status
-2 off	C8\	category code
-3 off	C9\	category code
-4 off	C2\	copy prohibit/permit
-5 on	C3\	pre-emphasis
-6 off	FC0	sample frequency
-7 off	FC1	sample frequency
-8 on	PRO\	professional/consumer mode

S2-1	on
S2-2	off
S2-3	off
S2-4	n.c.

JP1: /4

JP2: SCK

Top of the Range Volume Control

Switches instead of Pots

Design by G. Haas

Those who spare no expense for perfection, can now exchange their volume potentiometers for high precision step resistor switches

High-grade switches assembled with precision resistors have the advantage of absolute linearity and an extremely high channel separation. Hi-Fi fans can now link these switches together with a mute selector and forego the need to have a balance control. With a well-chosen speaker positioning to match the room acoustics, the need can be entirely dispensed with.

Figure 1, the circuit diagram, shows the volume switch set-up coupled to the mute switch. What is being used is not any run of the mill switch. The rotary switch must have a smooth gliding effect so as not to introduce any clicks, pops or sudden loudness during the operation. In order to have a uniform response, 24 switching steps have been chosen. The contacts must be gold plated to avoid clicks, which will arise during switching if the contacts become contaminated.

Common switches don't claim to

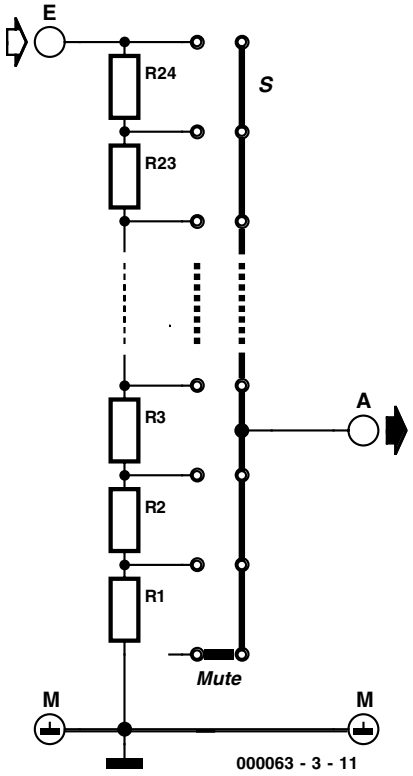


Figure 1. Circuit diagram of the 'top of the range' volume control.

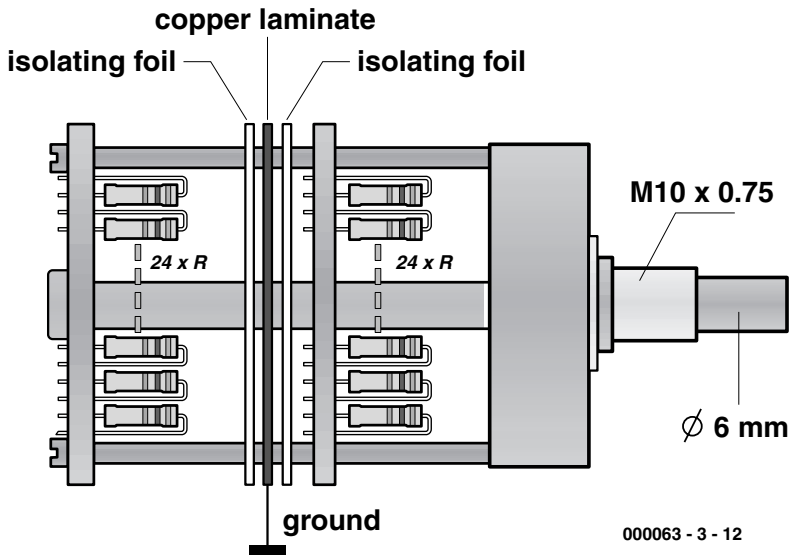


Figure 2. The modified stepped switch.

R	Pot 2.2 k log.	4.7 k log.	10 k log.	22 k log.	47 k log.
24	470	910	1.8 k	4.3 k	8.2 k
23	390	750	1.5 k	3.3 k	7.1 k
22	300	620	1.2 k	2.7 k	5.6 k
21	270	510	1 k	2.2 k	4.7 k
20	220	430	820	1.8 k	3.9 k
19	160	330	680	1.5 k	3.3 k
18	130	270	560	1.2 k	2.7 k
17	120	240	470	1.0 k	2.2 k
16	100	200	390	820	1.8 k
15	82	160	330	750	1.5 k
14	68	130	270	620	1.3 k
13	56	110	220	470	1.0 k
12	47	91	180	400	820
11	39	75	150	330	715
10	33	62	120	270	560
9	27	51	100	220	470
8	22	43	82	180	390
7	16	33	68	150	330
6	15	27	56	120	270
5	13	22	47	100	220
4	10	16	33	75	150
3	5.6	10	22	47	100
2	5.6	5.6	12	27	56
1	5.6	5.6	10	22	47

have optimal channel separation features, so it's necessary to take it apart as depicted in **Figure 2**. A pre-requisite is to disassemble the switch and place between both layers a well insulated copper foil, to which the ground point will be connected. By doing this modification, the channel crosstalk is less than the noise threshold generated by the resistor itself.

Selectively measuring a 1-% metal film 10-kΩ logarithmic switch will show a linear gain greater than 0.1 dB, a channel separation >100db at 1 kHz and 96 dB at 10 kHz as well as self-induced resistor noise of –95 dBV (20 Hz - 20 kHz) or 17,7 µV.

These values have been measured with the switch entirely isolated. At 10 kHz, the cross-talk attenuation is 1 dB (wide-band measurement) above the internally self-generated noise. As soon as a normal Audio filter is added, the results look even better. It's safe to assume that from this configuration the measurements show that cross-talk attenuation is always greater than the internal noise and thus becomes non-existent. In addition, while measuring the crosstalk values with selective filters, white noise coming from the signal source was clearly identifiable. In the table you will see the values in Ohms of the E-24 Resistor series corresponding to the equivalent potentiometer position.

(000063-3)rg

CONSTRUCTION GUIDELINES

Elektor Electronics (Publishing) does not provide **parts and components other than** PCBs, fornt panel foils and software on diskette or IC (not necessarily for all projects). Components are usually available form a number of retailers – see the adverts in the magazine.

Large and small values of components are indicated by means of one of the following prefixes :

E (exa) = 10^{18}	a (atto) = 10^{-18}
P (peta) = 10^{15}	f (femto) = 10^{-15}
T (tera) = 10^{12}	p (pico) = 10^{-12}
G (giga) = 10^{9}	n (nano) = 10^{-9}
M (mega) = 10^{6}	µ (micro) = 10^{-6}
k (kilo) = 10^{3}	m (milli) = 10^{-3}
h (hecto) = 10^{2}	c (centi) = 10^{-2}
da (deca) = 10^{1}	d (deci) = 10^{-1}

In some circuit diagrams, to avoid confusion, but contrary to IEC and BS recommandations, the value of components is given by substituting the relevant prefix for the decimal point. For example,

3k9 = 3.9 kΩ 4µ7 = 4.7 µF

Unless otherwise indicated, the tolerance of resistors is ±5% and their rating is ⅓–½ watt. The working voltage of capacitors is ≥ 50 V.

In **populating a PCB**, always start with the smallest passive components, that is, wire bridges, resistors and small capacitors; and then IC sockets, relays, electrolytic and other large capacitors, and connectors. Vulnerable semiconductors and ICs should be done last.

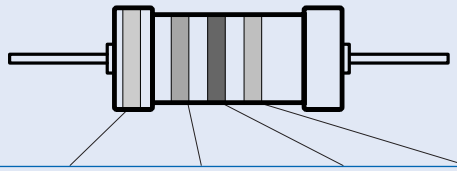
Soldering. Use a 15–30 W soldering iron with a fine tip and tin with a resin core (60/40) Insert the terminals of components in the board, bend them slightly, cut them short, and solder: wait 1–2 seconds for the tin to flow smoothly and remove the iron. Do not overheat, particularly when soldering ICs and semiconductors. Unsoldering is best done with a suction iron or special unsoldering braid.

Faultfinding. If the circuit does not work, carefully compare the populated board with the published component layout and parts list. Are all the components in the correct position? Has correct polarity been observed? Have the powerlines been reversed? Are all solder joints sound? Have any wire bridges been forgotten?

If voltage levels have been given on the circuit diagram, do those measured on the board match them – note that deviations up to ±10% from the specified values are acceptable.

Possible corrections to published projects are published from time to time in this magazine. Also, the readers letters column often contains useful comments/additions to the published projects.

The value of a resistor is indicated by a **colour code** as follows.



color	1st digit	2nd digit	mult. factor	tolerance
black	–	0	–	–
brown	1	1	×10^{1}	±1%
red	2	2	×10^{2}	±2%
orange	3	3	×10^{3}	–
yellow	4	4	×10^{4}	–
green	5	5	×10^{5}	±0,5%
blue	6	6	×10^{6}	–
violet	7	7	–	–
grey	8	8	–	–
white	9	9	–	–
gold	–	–	×10^{-1}	±5%
silver	–	–	×10^{-2}	±10%
none	–	–	–	±20%

Examples:
brown-red-brown-gold = 120 Ω, 5%
yellow-violet-orange-gold = 47 kΩ, 5%

GBDSO

Gameboy Digital Sampling Oscilloscope (1)

Convert a Nintendo games console into a portable oscilloscope

By Steve Willis

The application of dedicated games consoles to more varied functions other than just playing games is fraught with difficulties. Not only do the consoles contain highly integrated and custom-made components, there will often be no publicly available development tools or applications manuals. However the Nintendo Gameboy© is one such system which has received considerable attention from the hobby designer and enjoys a large selection of both technical and software support through a network of websites.

Key Features:

Dual trace display
Sampling Rate: DC to 1 Msps
Time Base: 100 s to 5 µs/Div
Inputs: AC/DC 1 MegOhm
Input gain: 50 mV to 10 V/Div
Line or chart recorder trace modes
Real-time FFT mode with dB scale
Variable persistence XY mode
PC link for screen or data transfer
5 hrs operation from NiMH batteries
Averaging and Auto trigger functions
Reference trace storage

FFT — the basics

The following details the specific implementation of the GBDSO FFT for those requiring a greater explanation of its operation. Initially, the input data is sampled using the standard scope capture programs. The data is then passed through a Hamming window to reduce artefacts caused by the finite sample points (rectangular window). A Hamming window was chosen as it gives a good compromise between main lobe and side lobe widths. The windowed data is recorded using a bit-reversal algorithm to suit the decimation in time (DIT) process of the FFT. The data is then processed by a discrete Fourier transform (DFT), which is implemented using a 256-point radix-2 fast Fourier transform (FFT).
16-bit arithmetic is used for the majority of the calculations, but since the GB has no multiply instruction this has to be calculated longhand by assembler software. The complex real and imaginary values of the DFT are combined to produce 128 x 32 bit values.
Finally, a logarithm is taken (1 bit = 6 dB) and the data is displayed on the screen. The entire processes (excluding sampling) takes 0.8 seconds of which the FFT accounts for 90% of the time requiring 4096 signed 16-bit multiplications and 6144 signed 16-bit additions.

The attraction of using a mass produced console soon becomes clear when the time and cost involved in producing a general purpose portable instrument with processor, LCD display and user interface, not to mention the moulded case, sound system and serial port, are considered.

The Gameboy Oscilloscope

The GameBoy 'GB' digital sampling oscilloscope 'GBDSO' converts a GBpocket or GBcolour into a multipurpose test instrument which should prove invaluable to the hobby designer. Design of GBDSO posed many contrasting requirements: high speed, low power, low cost, small size and versatility. In order to simplify the hardware as much as possible, real-time software is used to perform the majority of control functions e.g., triggering and variable-rate sampling. Once captured, the sampled data is displayed using a standard oscilloscope 10 by 8 screen format that is designed to maximise screen use. A simple four-position menuing system (one for each key) in combination with the joypad provides easy setting of the standard scope functions.

A number of advanced options are also possible for the more experienced user, these include FFT analysis, XY mode, averaging and reference storage. A serial data link allows the displayed screen or trace data values to be transferred to a PC computer via the computer's printer port. The PC software is MS Windows 95 compatible.

The oscilloscope has two input channels CHA/B, with independent software-controlled variable-gain amplifiers. The inputs have a 1-MΩ input impedance so they can be connected directly to 1:1 or 10:1 scope probes or to an audio jack lead. (limitations on size made BNC type connectors impractical). The amplifiers can be AC or DC coupled by means of a switch. The variable-gain amplifiers give an input sensitivity of 50 mV to 10 V per division with 10:1 probes and an input bandwidth of DC to 100 kHz.

The displayed trace has two modes of operation. For low frequencies (100 s to 100 ms/DIV), a chart recorder style of display is produced with the trace scrolling across from the right hand side. For high frequencies (50 ms to 5 µs/DIV), an entire screen of data is captured before it is displayed (standard mode). The number of samples captured per scan may be set to either 240 or 600 points per channel, allowing the visible screen window to be moved relative to the initial trigger. Sampling of the two input channels occurs simultaneously, except for the 10 and 5 µS/DIV ranges, when the channels are sampled on alternate traces relative to the trigger (alternate mode). CHA may also be saved as an on-screen reference, thus allowing a total of three traces to be displayed at once.

Gateway: the ROM cartridge interface

All the signals required for the oscilloscope are available on the external ROM cartridge slot, so no modifications need be made to the GB console itself.

The oscilloscope cartridge plugs into a custom 32-pin card edge connector on the underside of the GB and interfaces directly to

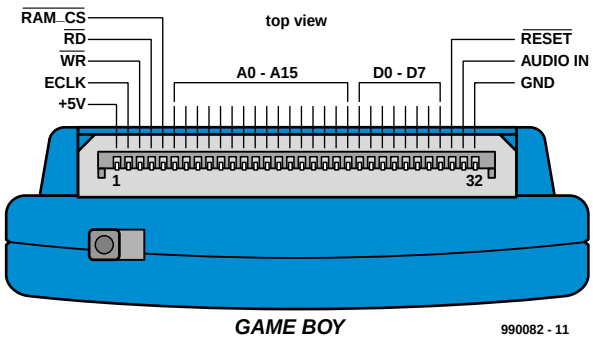


Figure 1. Gameboy card edge connector (end view of ROM cartridge).

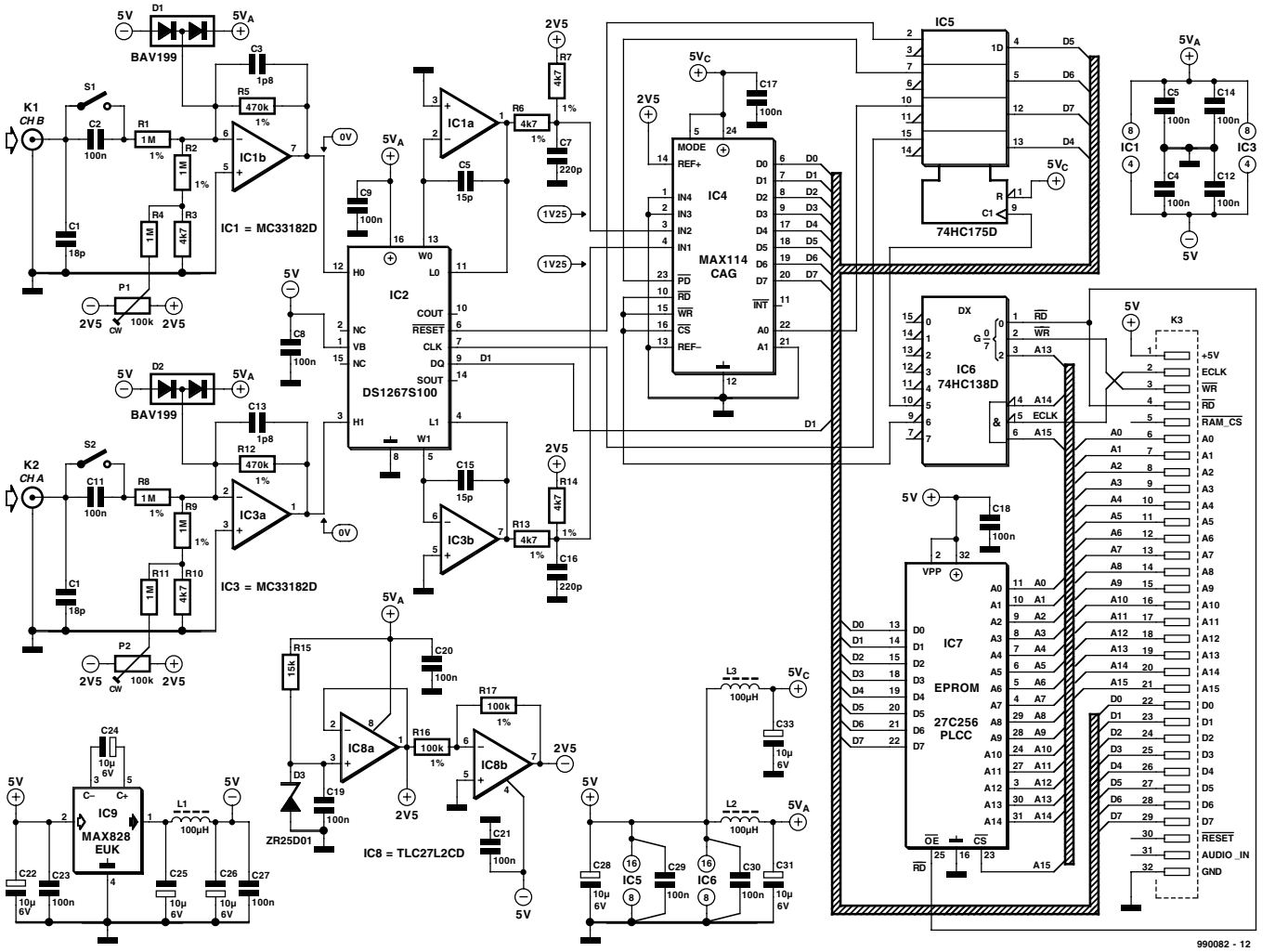


Figure 2. Circuit diagram of the GBDSO cartridge. Stop playing games — turn the Nintendo Gameboy into a sampling oscilloscope with a large clear display!

the data and address busses of the GB Z80 (-like) main processor. The pin functions of the card edge connector are shown in **Figure 1**.

Circuit diagram

As you can see from the circuit diagram in **Figure 2**, the GBDSO cartridge, despite its rather complex function, contains a relatively small number of integrated circuits.

The GB is controlled by an 8-bit Z80-like processor that gives a direct addressing range of 64 kB. However internal devices e.g., LCD, RAM, sound etc. use most of the top 32 KB leaving 0000-7FFF available for external ROM and A000-BFFF for external RAM. The GBDSO program is stored in a low power 27C256 32 kB EPROM, IC7, occupying lower memory at 0000 to 7FFF which is executed upon power-on. The ADC and input gain control chips occupy a single memory location at A000.

Address decoding consists of a 74HC138 1-of-8 decoder, IC6, that decodes the address range down to A000-BFFF and provides two enable signals:

1. **A000./RD** at pin 9 accesses the ADC and reads CHA/B 8-bit data value while reinitialising the ADC for the next sample.
2. **A000./WR** at pin 10 accesses the 74HC175 quad latch and writes data to the DS1267 gain control circuit, or selects the input sampling channel of the ADC.

The DS1267 from Dallas Semiconductor is a dual channel 10-kΩ digital gain control potentiometer with 256 wiper positions per channel. Data is transferred to the device through a serial interface via the 74HC175 latch, this updates both pot values simultaneously. Once updated, the values are held until the next update or the power is removed. A condensed datasheet of this interesting IC may be found elsewhere in this issue.

The GBDSO analogue circuitry requires ±5 V to operate, so an inverting capacitive charge pump circuit is used to provide the negative supply. The device used for this is a MAX828EUK (IC9) which has an internal oscillator that operates at 12 kHz. This allows relatively small capacitors to be used for smoothing. However, to prevent any high frequency noise being passed to the sensitive input amplifiers, additional LC filters are used.

As already mentioned, the CHA/B input amplifiers are designed to be compatible with 1:1 or 10:1 (0 or

9 MΩ) probes, each input amplifier having a relatively high input impedance of 1 MΩ. This is achieved by feeding the signal via a 1-MΩ resistor (R1; R8) into the negative input of a CMOS opamp (IC1b/IC3a) type MC33182D. This device is marked by low power, high slew rate and JFET inputs. By utilising the negative input of the amplifier, the signal is fed to a virtual ground, reducing bandwidth limitations imposed by stray capacitance on the inputs. Input protection is provided by D1/D2, which are type BAV199 dual low-leakage diodes. The negative input also forms a current summing junction for the DC zero preset (P1/P2) which is adjusted to reduce DC errors at high gain settings.

The variable-gain amplifier section consists of IC1a/IC3b and the electronic potentiometer inside the DS1267. The amplifier is configured as an inverting amplifier with the potentiometer forming the input and feedback resistors (wiper is W0). By changing the wiper position through software, the gain of the amplifier can be varied between 0.004 and 255, considerably reducing the amount of input circuitry required and eliminating the need for mechanical switches.

Interfacing to the 8-bit ADC type MAX114 from Maxim (IC4) requires an input signal in the range 0-2.5 V. To achieve a stable reference voltage with varying supply, a 2.5-V bandgap diode type ZR25D01 is used. DC biasing of the CHA/B amplified signal to match the ADC input is achieved with two 4k7 resistors which transform the ±2.5-V amplifier signal into 0-2.5 V for the ADC. The MAX114 is identified by Maxim as a '1-Msps, 4 &8-Channel, 8-bit ADC with 1-µA Power-Down'. The device is good for a conversion time of 680 ns per channel, and features internal track/hold circuitry that does not require an external clock.

Finally, the power supply is rigorously decoupled to prevent high-frequency noise affecting the operation of the sensitive input circuitry. A combination of electrolytic capacitors and smaller solid ones of 0.1 µF, plus two 100-µH chokes are employed to keep supply-borne noise to a minimum.

Table I. Key Function Menu

		Highlighted Function Key		
Joypad	***SELECT-'TRIG'***	***START- 'Timebase'***	***B - 'CHB'***	***A - 'CHA'***
△ UP ▽ Down	Trigger level △▽	Screen window Position ◁ ▷	Y Position △▽	Y Position △▽
▷ Right ◁ Left	Trigger mode[1]	Timebase scale[2]	Input gain Scale[3]	Input gain scale[3]

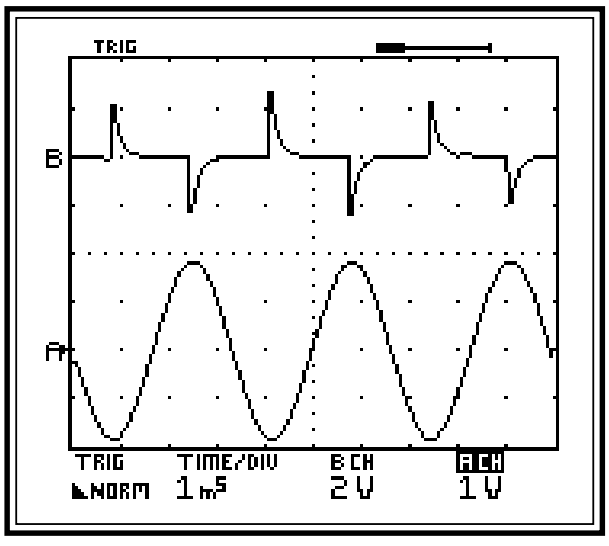
Notes.
1. Trigger modes for CHA are ▽Auto, △Auto, ▽Normal, △Normal.
   Auto trigger — produces a trace if a trigger has not occurred within a set time
   Normal trigger — holds the trace until a trigger occurs
2. Timebase scale range is 500,200,100,50S etc. up to 5 µS/DIV.
3. Gain scale range is 10,5,2,1V,500,200,100,50 mV/DIV,GND,OFF

Gameboy software

The GBDSO software that resides in the EPROM on the cartridge board was written in a mixture of assembler language and 'C', with assembler used for the real time data acquisition and display and 'C' provides the user interface. The 'C' compiler used was GBDK V. 2.17, which is a public domain program written by Pascal Felber and Michael Hope[5]. GBDK is an excellent means of quickly developing dedicated software for the Gameboy, as it contains many predefined functions for interfacing to dedicated GB hardware e.g., joypad, screen and sound generator. For more information on software development have a look through the list of references which will be given at the end of next month's concluding instalment.

User interface

On power up the GB starts to execute the GBDSO interface program that resides in external EPROM. The program initially displays an intro screen and a set of four predefined start options which initialise the scope settings: Single/Dual/Logic/AC. The scope is now ready for use. **Figure 3** shows a typical display screen with both channels enabled,



990082 - 13

Figure 3. Typical view from dual trace screen.

Standard menus
By pressing A/B/Start/Select, the appropriate function will be highlighted on the bottom line of the scope (note CHA is highlighted). Once a function is highlighted, the joypad control can be used to adjust the values associated with that function, as shown in **Table 1**.

Advanced menus
The function key menu provides a quick means of changing the standard scope settings with as few key presses as possible. However, if the highlighted function key is pressed a second time then an advanced menu will be displayed. The advanced menu options may be found in **Table 2**.

Each advanced menu has three options and an exit. When an option is selected with the function key it will be enabled immediately and the screen will return to normal operation. Some options switch between one of two modes e.g., average on/off. The normal screen will display which of the two modes is currently selected. The following section describes in greater detail the operation of each option.

'A' Advanced menu
600/240pt, selects the number of points sampled each trace scan. By moving the screen window position with the 'timebase + ↑↓' menu, it is possible to scroll through up to three screens of data in 600pt mode. The current window position and size are shown in the top right corner of the screen. The default mode is 240 points as this gives the fastest screen update rate. In FFT mode this is fixed at 240 points.

Auto Trigger calculates the average value of CHA relative to the current timebase setting, the trigger is set to this value.

Average, averages CHA (and CHB if enabled) using previous trace scans. The average is calculated from X=X+(Xnew–X)/8. This can be useful for removing uncorrelated noise from a trace, so the required signal can be measured.

'START' advanced menu
CHA⇒REF, Stores the current trace of CHA to the reference channel and displays it allowing up to three traces on the screen at once. Selecting this menu again will clear the reference. The letter R will appear in the left-hand column indicating a reference trace.

SCREEN⇒PC, The current scope screen will be transferred to the PC computer via the link lead. When the start of transfer has been acknowledged, the GB will beep and the progress indicator will change on the PC. At the end of transfer the GB will beep again. The screen will be converted to a black and white 160x144 .BMP format by the PC and can be saved to disk.

DATA⇒PC, The current data values (not screen) of CHA and CHB will be transferred to the PC computer in a similar method to that above. Data can be saved to disk as raw data or in MathCad© 6.0 format for importing back into the computer for analysis. See the section on the PC Link Interface (part 2) for more details.

'SELECT' advanced menu
FFT, changes scope operation to FFT mode. FFT mode produces a spectrum analysis of the sampled data, where the horizontal axis represents frequency and the vertical axis represents amplitude in dB (10 dB/DIV). See the FFT section for more information on the FFT mode.

XY, changes the scope operation to XY mode with the horizontal trace deflection set by CHA and the vertical deflection set by CHB. The trace persistence (how long each point is

Table 2. Advanced Menu Screens

Function Key	Advanced menu screens
Select	EXiT FFT XY TiME
Start	CHA →REF EXiT SCREEN →PC DATA →PC
B	Not used
A	600/ 240PT AUTO TRIGGER AVERAGE EXiT

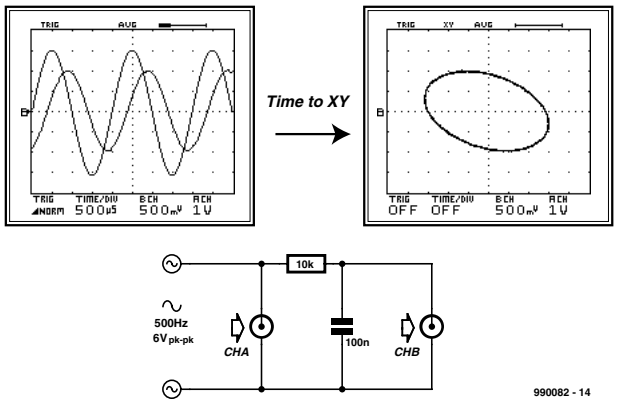


Figure 4. RC network and XY mode.

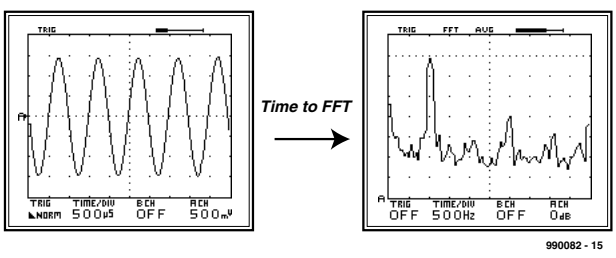


Figure 5. FFT of 1 kHz 0 dB sinewave.

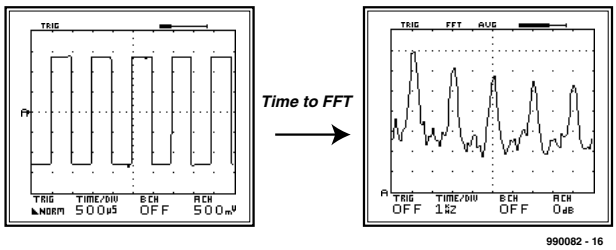


Figure 6. FFT of 1 kHz 0 dB squarewave.

displayed) can be set to 100 ms, 1 s, 10 s, 100 s or OFF (infinite). The maximum number of points plotted on the screen at any time is 600. See the XY section for more information on the XY mode.

TIME, changes the scope operation to normal scope mode with the horizontal axis representing Time after an initial trigger point and the vertical axis representing amplitude in Volts.

XY Mode

The XY mode allows the horizontal and vertical deflection to be controlled by CHA and CHB respectively. This can be used to display how one circuit parameter is changing with respect to another.

A simple RC network demonstrates how phase shift is introduced into a circuit by the capacitive component, see **Figure 4**. The input sinewave moves the trace horizontally and the 'phase advanced' output from the RC network moves the trace vertically resulting in a circular image. Changing the frequency and waveform type gives some interesting effects.

In XY mode the trace persistence (how long each point is displayed) can be set from the timebase to 100 ms, 1 s, 10 s, 100 s or OFF (infinite). When the persistence is set to OFF, the points plotted on the screen remain visible until the screen is cleared by pressing one of the function keys. When the persistence is set to one of the time values, a maximum of 600 points can be displayed at a time and they will remain visible for length of time defined by the timebase setting. Therefore, the 100-ms setting is used to measure fast changes and the 100-s setting for measuring slow changes.

FFT Mode

Normally, signals are considered as varying in amplitude with time and are therefore measured as a function of voltage versus time. However, all continuous signals can be constructed from a fundamental and harmonically related (Fourier) components which are best represented logarithmically as magnitude (dB) versus frequency (Hz). This is more commonly known as a spectral analysis and is often used in audio systems to measure the performance of an amplifier, or the purity of a signal. The GBDSO computes the spectral analysis using an algorithm known as the fast Fourier transform or FFT which is a mathematically intensive method, normally associated with high speed digital signal processors – DSPs. Nevertheless, by coding the algorithm entirely in assembly language and optimising its performance, the FFT is performed in just 0.8 seconds.

A full discussion on frequency analysis is beyond the scope of this article, but by considering a few examples it will soon become clear how useful the FFT mode is and in what circumstances it should be used. **Figure 5** shows a 1-kHz sine wave of amplitude 0 dBV_{rms} or 1.414 V_{pk} (0 dB = 20log(1V $_{rms}$), rms=pk x 0.707). The vertical scale is fixed at 10 dB/DIV and the reference marker (represented by a dotted line near the top of the screen) is determined by the ACH gain setting in dB. Since the sine wave contains only one frequency component (the fundamental), the frequency spectrum shows a single peak at 1 kHz. With the reference marker set to 0 dB, the noise floor is about –45dB down. By comparison, the 1-kHz square wave of **Figure 6** contains a fundamental and odd harmonics at 3, 5, 7 kHz, etc. which reduce in amplitude as the frequency increases.

Closer observation of Figure 5 shows a spurious spectral peak at 3 kHz which is 30 dB down on the fundamental, this is in fact produced by the signal generator and represents distortion of the pure sine wave. The time response gives no indications of this problem and it is clear that without the FFT mode this imperfection would not be measurable but it could cause problems when analysing circuits.

Using the FFT mode can often give unexpected results and some experimentation and experience is required to obtain meaningful results. Here are some points to consider.

Make full use of the 8bit samples, if possible adjust the signal amplitude or reference marker so that the input signal is at its maximum i.e.: the highest peak is just below the reference marker. The Reference Marker range is 35, 30, 20, 15, 10, 0, –5, –10 dB, GND. Do not increase the signal above this point as this will saturate the scope amplifiers and introduce spurious harmonics.

If a signal contains a wide spectrum of frequencies, often the frequency components that extend beyond the FFT will be reflected back so that they become visible again, which can be misleading. By increasing the timebase frequency you can reduce this effect. The Timebase scale range is 10, 25, 50 Hz etc., up to 100 kHz/DIV.

To lower the noise floor, use the averaging mode.

Measuring small signals with a large fundamental, is possible if you use a notch filter circuit (e.g., twin-T) to remove the fundamental component before the signal is applied to the scope. This effectively increases the scope's dynamic range.

(990082-1)

Next month we continue with the PC Link Interface, the construction of he cartridge, testing and calibrating the instrument.

Warning

When using 10:1 probes, the maximum input voltage applied to GBDSO must not exceed ±50 V; with 1:1 probes the maximum input must not exceed ±16 V. Under no circumstances should 230/110 V mains voltages be applied to the GBDSO.

‘OSCAR’ MP3 Player (2)

Part 2 (final): operation and measurement results

Design by A. Kurpiers and V. Pantelic

The OSCAR MP3 player is operated according to a new, intuitive concept which is vastly different from anything you’ve seen on conventional audio equipment.

Figure 1. Front panel of the ready-built unit with all controls and the LCD.

Although there are no hard and fast ‘rules’ for the format of a CD or hard disk when it comes to song titles and the order in which they are stored, it is useful and convenient to arrange for an MP3 CD or a hard disk filled with MP3 tracks to have a hierarchical structure as we know it from working with a PC. Using a fixed structure ensures that you do not loose track (literally…) of your favourite songs. After all, an MP3 CD may contain over one hundred tracks, and a hard disk, thousands, so you have to get organised!

When in playback mode, OSCAR not only indicates the song title, but also the directory structure. This allows meaningful names to be assigned at all three levels (Directory/Subdirectory/File). For example,

Artist/Album/Song.mp3.

Another example is shown in the screendump in **Figure 3**.

Keypad

To prevent an unwieldy number of functions and options coupled to an equally large number of key combinations, most functions and settings may be reached via a hierarchical menu structure. Only five keys are required, each with a specific function, plus a rotary encoder with a key function. You will also look in vain for a display that’s only capable of showing a couple of numbers and some special symbols. OSCAR has a 4-line LC display that allows all relevant information on song title or operation to be shown in plain text.

On the front panel of OSCAR (**Figure 1**) you’ll find the five pushbuttons, the rotary encoder and the LC

Measurement results

Parameter	MP3-Player	Audio-DAC 2000 with S/PDIF-extension
Nominal effective output voltage	1 V	2.1 V
Level 20 kHz	–0.95 dB	–1 dB
Output impedance	1 kΩ	100 Ω
Signal/Noise ratio (no signal)	–96 dB	–111 dB
THD+N (1 kHz, B = 80 kHz)	0.17 %	0.0037 %
IMD (60 Hz:7 kHz = 4:1, 0 dB)	0,06 %	0.009 %
Channel separation 1 kHz	>100 dB	>115 dB
20 kHz	>76 dB	>90 dB
Lower roll-off frequency (10 kΩ load)	17 Hz	DC-coupled

display, besides the mains switch and the IR remote control 'eye'.

EJECT
This allows the CD-ROM drive tray to be opened and closed. This function also works while a track is being played from the hard disk.

SELECT and **UP**
SELECT gives access to the rotary encoder, which also acts as a normal pushbutton. The rotary encoder is the central control element, allowing quick access to the (vast number of) songs on the CD or the hard disk, as well as to the plethora of functions and settings offered by the menus. The operation is not unlike that of a mobile phone, and you will soon get the hang of it. To select a song or a directory, you simply turn the knob until the desired entry is found. Next you press (click) the encoder knob to select a directory, song or function.

Once you are in a menu, menu options may be picked by turning and clicking. If you end up in the wrong directory or subdirectory, you can go to the upper directory by pressing the UP button, and from there, go to the main menu. If OSCAR is fitted with a hard disk, the UP button allows you to choose between CD player and hard disk at the top directory level.

While in 'play' mode, turning the rotary encoder enables you to skip tracks or go back in the track list. The menu allows you to assign either function to the knob.

MENU
This is where the central control of OSCAR is stored. The menu allows all settings to be performed and stored. The complete overview of available commands may be found on the Elektor website (free downloads section, October 20000). You can leave the menu by using MENU, UP or ESC•.

STOP
Ends playback mode.

SOFTKEY
Although this key does not have a proper name, it does give access to a number of functions. The currently active function is shown in the right-hand bottom corner of the display:

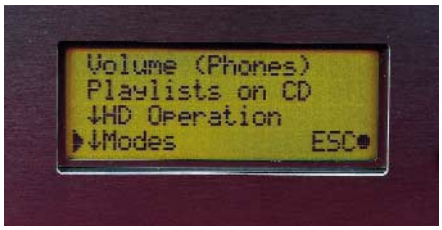
NXT•	*Go to next track*
PRG•	*Add title to playlist*
Enter•	*Quit text entry*
ESC•	*Terminate operation*
DEL•	*Clear title storage in radio mode*

LC Display

The LCD can operate in one of three different modes: selection display, playlist display or menu display.

Menu display (Figure 2a)
The readout appears after pressing thre MENU button, and enables you to navigate through the equipment settings. The dark triangle ('Play' sign) points at the menu entry. A 'down' arrow in front of an entry indicates that a subdirectory is available. The current function of the SOFTKEY is shown in the right-hand bottom corner of the display.

Selection display (Figure 2b)
In this mode you can select a certain songtitle or directory. The 'down' arrow then indicates a directory, the character ≡ stands for .m3u-playlist,

a


b

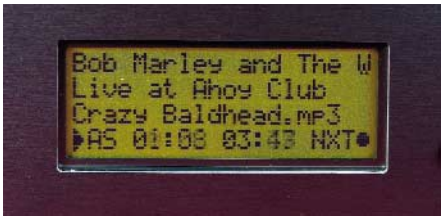

c


Figure 2. LC Display in three different modes:
a) Menu Mode
b) Select Mode
c) Playback Mode
The current application of the Softkey always appears in the lower right-hand corner.

while the musical note sign points to the MP2/3 track.

The bottom line indicates the selected playback mode and its current status. The symbols used are the same as seen on conventional audio equipment. So, the dark square will be recognized as the STOP sym-

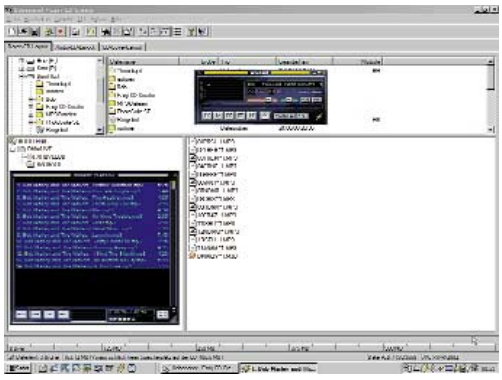


Figure 3. Ready to burn! The Explorer shows the directory structure (lower left), with beside it the lowest directory level containing all mp3 tracks and the m3u playlist.

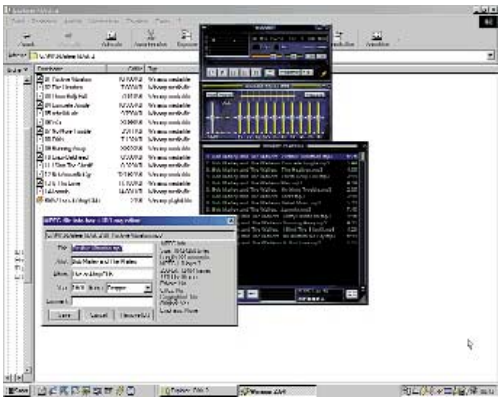


Figure 4. Using the Winamp ID3 Tag Editor (lower left) you can save information like Artist, Song Title, and Album. If required, this information is indicated on the LCD.

bol. Next to the STOP symbol is the play mode (1st position) as selected in the Menu/Mode, or the shuffle mode (2nd position). For example, 'AN' on the display means All Songs, Normal order (see table).

Playback mode (Figure 2c)
This is the normal readout during playback. The first three lines originate from the MP3 tag. In the fourth line, the display shows the current status (here, a Play symbol), followed by playback/shuffle mode (PS = random selection from a playlist), the track duration and the remaining time. The current function of the SOFTKEY is again shown in the right-hand bottom corner of the display (NXT = next track).

Playlists and Programmes

Besides the direct selection of tracks it is possible to define the order in which tracks are played in a so-called playlist, which was created before the CD was 'burned' or the hard disk was given its structure on the PC. Thanks to Winamp, M3u playlists now abound on the web. They may, however, also be made with other software MP3 players. When creating an MP3 CD, it is best to save the playlist in the lowest directory level, together with the MP3 files. This is illustrated by **Figure 3**, see the sub-window in the right-hand bottom corner of the CD burner program. Although this method provides structure and order, OSCAR is also capable of handling playlists separately from the tracks. In fact, they may be selected via a dedicated menu option (Playlists on CD).

It is not absolutely necessary to compile the playlist beforehand, however, as OSCAR can also do this. Simply go to the Program mode (via MENU – MODES – PLAYMODE – PROGRAM) and select songs or entire subdirectories in the usual manner using UP/SELECT. Using the SOFTKEY (PRG•), the selected track may be included in the playlist. As soon as a programme exists, only the programmed tracks are played (press SELECT to start). To play other tracks, you may either select another play mode, or start a non-programmed title using SELECT. When OSCAR is switched off, the programmed playlist is cleared. To prevent this, you may save a programmed playlist on hard disk using MENU – PROGRAM – SAVE PROGRAM. OSCAR will then prompt you for a name to be given to the playlist. You may enter the name by turning and clicking SELECT. The storage operation is initiated by the SOFTKEY. The target directory for the playlist may be selected before saving, via MENU – HD OPERATIONS – SET DESTINATION. The contents of the current playlist may be examined via MENU – PROGRAM – VIEW PROGRAM.

Figure 5. A look inside the case of a fully loaded OSCAR.

VBR and VB3

The acronym VBR stand for Variable Bit Rate and allows the degree of compression to be optimised depending on the programme material. If 'not much' is heard in a piece of music, the amount of data will be relatively small. Consequently the bit rate of the MP3 format is reduced. Conversely, a loud crescendo with its large amounts of data will cause the bit rate to go up again. To prevent the time display to change abruptly as a result of the continuously changing bit rate, OSCAR handles a VBR header loaded at the start of the MP3 track, thereby ensuring a steady and reliable readout of the song duration.

At the end of an MP3 file one often finds the so-called ID3 tag. This is special data set containing information not only on artist, album and song title, but also on production date and musical classification. This information may be quite useful if the file name does not say it all!

ID3 tags may be created with the ID3 tag generator function of Winamp. The screendump in Figure 4 shows that the *ID3 tag generator* not only saves text, but also infor-

mation on the file format, in the *MPEG file info box*. If available, and if so desired, textual information is indicated when the song is played.

Select, Wind, Control

As already mentioned, the rotary encoder has several functions while a track is played on OSCAR. These functions may be defined via MENU – MODES – WHEELMODE. In 'Select' mode, you may search and select the next track. In 'Cue' mode, turning the encoder knob produces the 'fast forward' and 'fast rewind' functions. The 'Speed' mode, finally, enables you to control the playback speed without affecting the pitch. This is made possible by OSCAR's special way of pre-processing the MP3 data in, and is **unique** as far as we know. Within a span of ±10%, audible degradation is hardly noticed, while the function should induce experimentation! The current speed is shown in the right-hand bottom corner of the display. With normal audio CDs, the settings 'Cue' and 'Speed' are associated with the functions 'fast forward' and 'fast reverse'.

CD-ROM or hard disk?

In the (not unlikely) case of your collection of MP3 CDs starting to cause serious storage problems, OSCAR should be fitted with a hard disk for increased storage capacity. Currently available storage capacities of modern hard disks should provide ample for a vast music collection stored in a single piece of equipment. The hard disk may be formatted in OSCAR. By pressing a button, individual songs, whole directories or even complete MP3 CDs may be copied on to the internal hard disk. OSCAR also allows directories to be created, song titles to be cleared, and the HD free space to be indicated. Alternatively, the hard disk may be formatted and 'recorded' in the PC, for use later in OSCAR.

OSCAR supports the widely used FAT16 and FAT32 file systems.

OSCAR will work with any modern hard disk with a capacity greater than 1 Gbyte and supporting LBA mode. The actual capacity

Construction variants

OSCAR may be obtained from Baycom as a ready-built unit or a kit, including remote control. The kit, priced at £149 contains the stuffed main and front boards (including display) and an extensive construction manual. The more extensive variant priced at £249 also includes a CD-ROM drive, a steel case and the PSU board. A 20-GB hard disk is available as a further option. Because there is hardly any solder work to do, we can concentrate on a few details. The photograph shows the ingredients of the kit. The following items should be added: case, cables and a few mechanical and electromechanical parts, a power supply and at least one drive unit, that is, you need to add your own CD-ROM player and/or hard disk drive (unless, of course, you buy the extensive kit).

PSU:
The power supply unit (PSU) has to be capable of supplying voltages of +5 V at 2 A and +12 V at 2 A.

CD-ROM and hard disk:
Both drive units are connectd to the main board by a 40-way IDE cable with a maximum length of 50 cm. If no hard disk is used, the CD-ROM player is set to Master mode. If a hard disk is used, this is assigned the 'Master' function — the CD-ROM drive then becomes the Slave. To be able to play normal audio CDs you'll need a screened cable between the CD-ROM drive and the main board.

LCD and Keyboard
A 20-way flatcable with a maximum length of 50 cm connects the keyboard and the main board. The link between the keyboard and the display is two times 6-ways. To be able to adjust the LCD contrast with the case closed, a small hole should be drilled below the cermet preset.

Additional parts to fit on the case:
Front side: mains switch (illuminated), stereo headphones socket
Rear side: IEC mains appliance socket (with in-built fuseholder), 2 Cinch sockets, 9-way RS232 chassis-mount socket (female).

Obviously, there is nothing against turning OSCAR to the standard audio source in your car! Unfortunately, CD-ROM drives are susceptible to strong mechanical shocks and vibrations that may occur in car. If you are firmly intent on equipping OSCAR with a CD-ROM drive in your car, be sure to use a spring suspension system. Hard disks, too, should benefit from rubber suspension, although no problems were encountered in practice. We recommend using 2.5-inch HD drives as applied in notebook computers. In a car, the cable between the main board and the display should not be longer than about 50 cm. Essential, but also problematic in a car, is the availability of a sufficiently stable supply voltage. Because the terminal voltage of a car battery may vary between 8 V and 14 V, you'll need to use a combined step-up/step-down converter to generate regulated +5 V and +12 V rails. A suitable circuit is planned for publication on the OSCAR homepage as well as in Elektor Electronics.

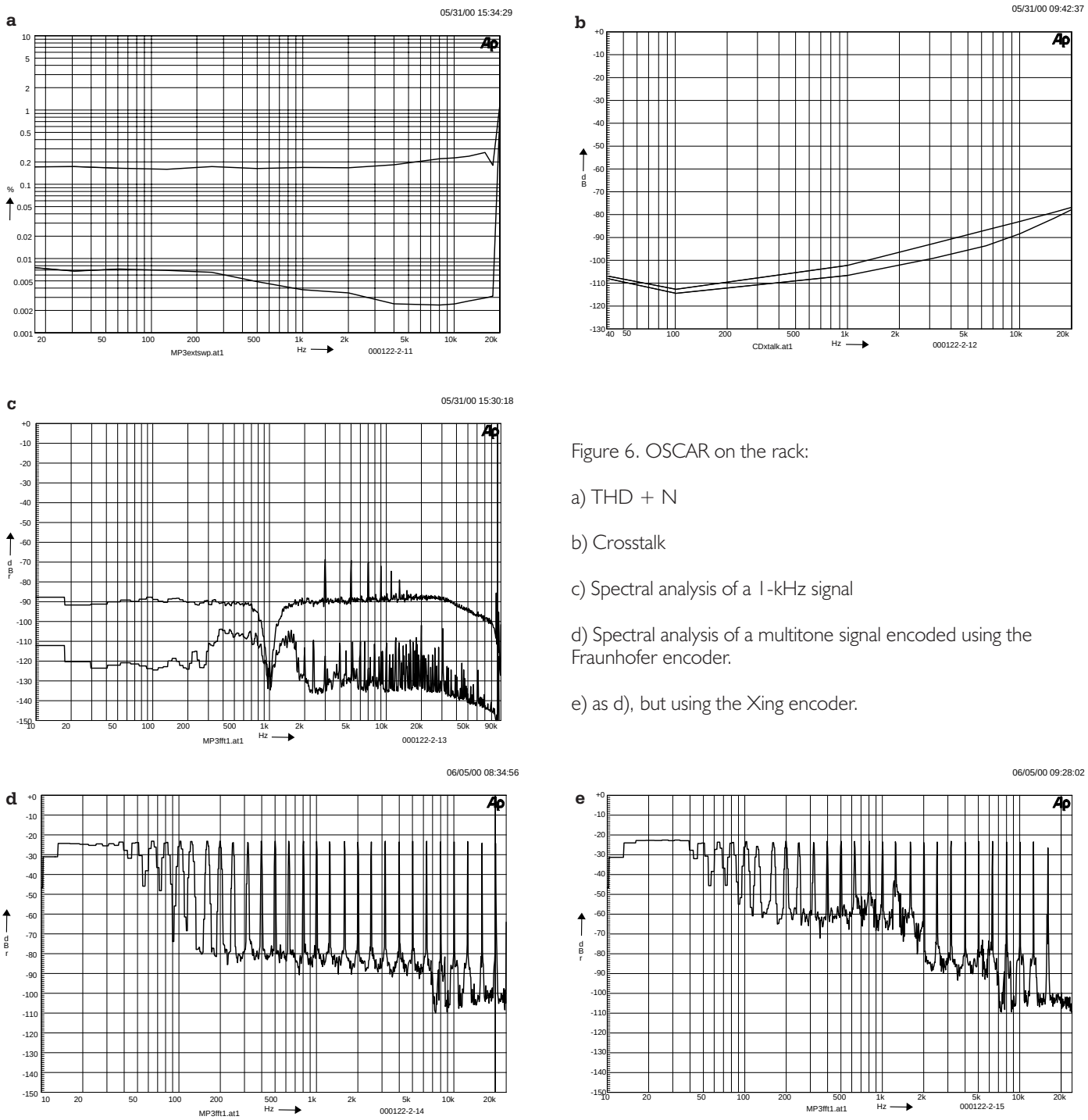


Figure 6. OSCAR on the rack:

a) THD + N

b) Crosstalk

c) Spectral analysis of a 1-kHz signal

d) Spectral analysis of a multitone signal encoded using the Fraunhofer encoder.

e) as d), but using the Xing encoder.

of the hard disk is immaterial — even 60 Gbyte ones will work without problems. **Figure 5** shows a fully loaded OSCAR with an internal hard disk.

Update

With PCs, software updates seem to be required on a monthly basis. By contrast, the update phenomenon is unknown in the field of hi-fi equipment. The operating system for OSCAR is subject to constant extension and improvement. Updates may be downloaded free of charge from the OSCAR homepage http://www.oscar-mp3.de An update may be downloaded and either burned on a CD, or sent to OSCAR via the PC and the RS232 serial port. The communication settings are 115 kbit/s, 8 data bits, 1 stop bit, no parity.

Many suggestions and wishes of existing OSCAR users have already found their way into the current version of the operating system, and many more are expected after publication of this article. However, note that it is not (yet) possible to send MP3 files to the OSCAR hard disk by way of the RS232 port.

Measurement results

As a matter of course, OSCAR was subjected to an extensive series of

(gruelling) tests in the *Elektor Electronics* audio laboratory. A number of test CDs were purposely encoded as MP3 files. The encoder used was L3enc V. 2.72 from the Fraunhofer Institute, set to the highest resolution (128 kbit/s, f_s = 44.1 kHz).

The measurements were not only carried on OSCAR itself, but also in conjunction with the high-end *Audio DAC 2000* published by this magazine. For this to work, a special S/PDIF extension had to be devleoped first. This design appears elswewher in this issue. As you can see from the measured values and curves, it makes sense to employ a high-end external converter, even though the measurement results are a far cry from the behaviour of the Audio DAC 2000 with real S/PDIF signals (see *Elektor Electronics* January 2000).

Curve 1
A comparison of the harmonic distortion plus noise indicates that the analogue audio circuitry in OSCAR is not up to the performance of a high-end converter. At 1 kHz, the distortion of the Audio DAC 2000 is 45 times lower than that of OSCAR. The measurements were carried out in a bandwidth of 80 kHz. The jump in the curves at about 20 kHz is caused by the discrete measurement points at 18 kHz and 20 kHz. In reality, the curves extend to almost 20 kHz.

Curve 2
shows the channel separation at the linear output of the MP3 Player. Up to 1 kHz, the signal is clearly affected by noise.

Curve 3
The top curve shows the frequency spectrum of the MP3 Player at a 1-kHz signal at maximum volume (0 dB). The noise floor is relatively high at –90 dB, and harmonic distortion is clearly visible.

The lower curve shows the result of the same measurement, but then at the output of the audio DAC (i.e., OSCAR acting as a digital signal source). The noise floor is 20-30 dB lower than with Curve 3, and harmonic distortion is down by no less than 40 dB. The curve also shows the effect of masking in MP3: the noise floor around the (suppressed) 1-kHz test signal is clearly higher than in the other ranges. Moreover, a series of mixing products is created.

The test session also included measurements using multitone signals (31 frequencies, 1/3th octave distance and 20 kHz). Below 100 Hz, the curves provide little useful information — the 'flat' response is caused by the linear resolution. In reality, however, there are also a couple a discrete frequencies with a noise floor as in the rest of the frequency range.

Curve 4
Shows the frequency spectrum with a similar multitone measurement, this time captured at the output of the MP3 Player. The spectrum was also measured at the output of the Audio DAC (via the above mentioned S/PDIF converter). The curve, which is not shown here, was not significantly different — only the noise floor was about 7 dB lower overall. When compared with the 1-kHz measurement, the noise floor with complex signals is about 30 dB higher than with a single discrete frequency.

Curve 5
Here you can see that the playback quality is heavily dependent on the quality of the MP3 encoder. The test setup is the same as that for Curve 4. However, we used the (fast) Xing MP3 Encoder V1.5 to encode the multitone spectrum. A quick comparison indicates that frequencies above 16 kHz are simply suppressed, although *High Frequency Mode* was set to 20 kHz. The noise floor is very irregular — even below 2 kHz, noise is 20 dB higher than with the Fraunhofer encoder. Here, quality comes at the cost of time: whereas the Xing encoder requires just 15 minutes per CD, the DOS-based Fraunhofer encoder takes ten times longer to complete the job.

One final note regarding the MP3 Player. If you decide to buy OSCAR as a ready-made unit (see the Baycom advert elsewhere in this issue), you'll find that the analogue outputs are located on the switch-mode power supply board, rather close to the transformer. Our measurements indicate that the resulting 50-Hz stray signal was clearly noticed at about 20-30 dB above the noise floor. For the measurements discussed here, the Line signals were taken directly from the decoder board using short cables terminated in RCA (Line) sockets, as far away as possible from the PSU board.

(000122-2)

The menu in detail

A table listing all control menu options and their effect is available as a .pdf document from our website at http://www.elektor-electronics.co.uk.
You'll find the table in the Free Downloads section, October 2000.

Software and links

Software to create and process MP3 files on a PC may be found at

AudioCatalyst	http://www.audiocatalyst.com
MusicMatch Jukebox	http://www.musicmatch.com
Playback software for the PC	
Wimamp	http://www.wimamp.com
website dedicated to MP3	http://www.mp3.com
Information on MP3	
Info, tips, help	http://www.mpex.net
Basics (file 000044-11, May 2000	http://www.elektor-electronics.co.uk/dl/dl.htm

urls available as hyperlinks on the Hyperlinks page of the Elektor Electronics website.

PCI-Hosted Measurement Cards

Towards the computer-driven test lab

J. Häuser

Computer systems in test environments are no longer a rarity. Many applications fall back on the use of a PC as test equipment and end up producing a system that would not be feasible with conventional test equipment. With a suitable package consisting of sensors, test hardware, PC and standard test software it is possible to build a highly flexible system.

Figure 1. Analogue and digital scales.

To use any PC as a test instrument it must first be capable of inputting measured values during the test procedure, working with these values and then outputting the results in a form that can be used by other laboratory equipment, or if necessary to alter the test procedure itself. To convert a PC into a complete test laboratory three additional components are necessary:

A sensor that can take the measurement and output it as a proportional voltage.
Suitable test hardware.
The measurement software.

Finding suitable PC test hardware is relatively easy, but tracking down a user friendly program to capture test data proved to be not so simple. The previous article gave advice for choosing test hardware. A future article in *Elektor Electronics* will give advice on choosing suitable software products for the test environment. A follow up article dealing with typical sensors to take physical measurements is also planned.

Signal Conversion

The world around us is characterised by analogue behaviour. An analogue signal is one that can have an infinite number of positions between two measurement points. A moving-coil meter for example, is an instrument whose pointer is deflected in direct proportion to the current flowing in its coil (**Figure 1**).

Using a digital computer, microprocessor or DSP to work directly with an analogue signal would in theory require infinite computing resources and is therefore not possible. Instead, it is only possible to read the signal value at fixed time intervals, or so called discrete time intervals. The process of making a discrete time signal from a continuous time signal is called sampling. Sampling occurs at equi-distant points on the analogue waveform i.e. at regular time intervals. The time between any two samples is called the sampling period. This is also known by its inverse value, the sample frequency and is of critical importance when choosing test hardware.

To make a properly informed decision when choosing test hardware for a particular test job it is important to list all the important hardware requirements.

This list should include the following criteria:

Interface choice for the test hardware.
The type of converter.
The need for sample & hold
Signal resolution
speed (sampling rate)

After this process, your choices are already becoming limited but there

are other factors that also need to be brought into the equation. For example the number of analogue channels needed, the output voltage range of the sensors, the number of any D/A converters required, the necessary digital interface, the type of stop and start signals for the A/D converter, the need for galvanic isolation (usually achieved by optocoupling), all need to be added to the list of requirements. Last but not least are the questions of budget, anticipated equipment delivery time and the reliability of the product supplier.

Expansion Socket Architecture

There are many manufacturers of PC test cards that use standard expansion cards architectures. So you can take your pick between PCI, Compact PCI, PCMCIA, ISA, USB, Parallel-Port and COM-Port. But for which application should you use which architecture? The following explanation should help to make things clearer.

PCI

The key factor governing the power of any PC test environment is the type of bus used. Despite the relentless increase in processor speed and memory capacity the main speed bottleneck occurs at the communication channel with the expansion card.

For many years the ISA bus was one of the most common methods of interfacing to the PC but today it does not have the capacity to make use of the power of current processors. The PCI (Peripheral Component Interconnect) bus was developed to address this problem. It offers a maximum data rate of 132 MB/s.

The specification for this bus ensures that the interface can keep pace with increased processor speeds and upgrade investments made to the PC will not be wasted.

This interface standard is quickly becoming accepted by the industry, its capacity to accept future processor enhancements means that it will be an enduring standard.

The PCI solution is recommended for use in the following situations:

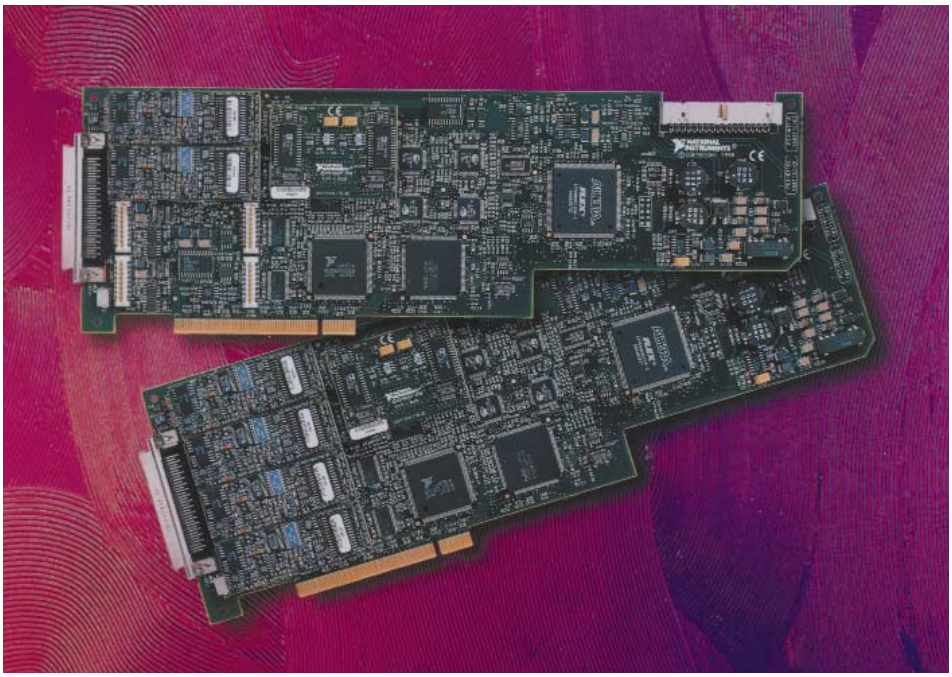


Figure 2. PCI Test card.

For use in stationary test equipment using the standard operating systems.

Wide range of industrial standard connectors between the PC and test hardware.

Price range £180 - £2500 approx.

One example of the fast analogue data capture possibilities of this interface is the data acquisition card PCI-6110E from National Instruments shown in **Figure 2** along with its block diagram in **Figure 3**. This device achieves a data capture rate of up to 5 MHz. The main feature of this card is its ability to

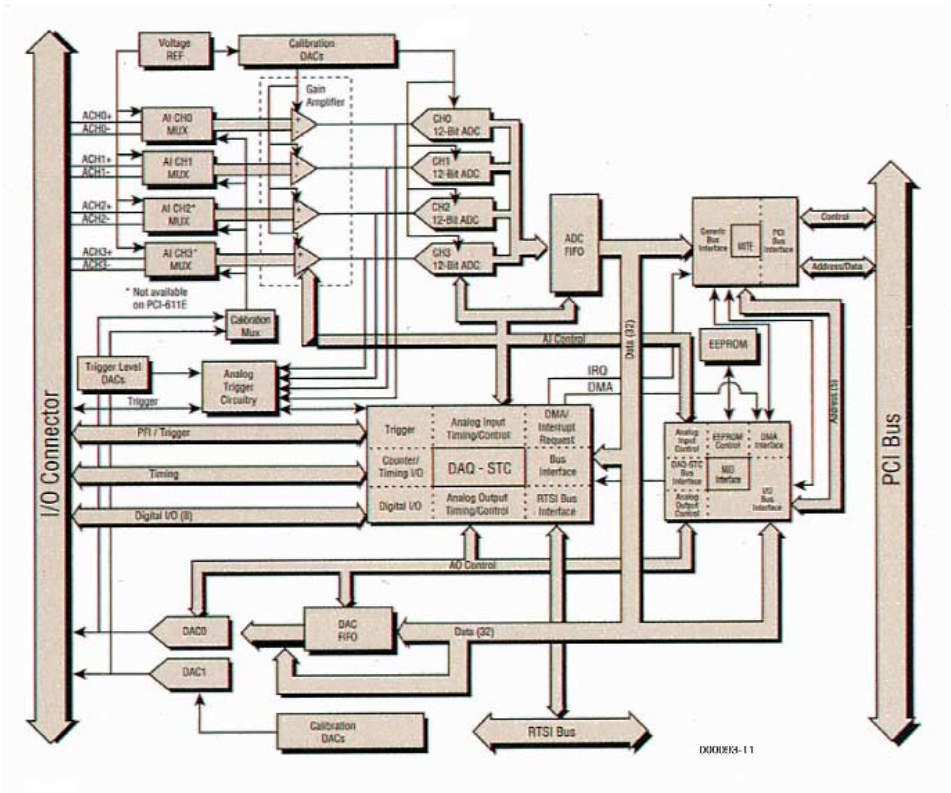


Figure 3. Block diagram of a fast PCI test card.

Figure 4. Compact PCI system.

sample the four channels simultaneously and stream the measurement data directly to the system hard drive. Each of the four differential analogue input channels (CH0...CH3) has its own 12 bit A/D converter. The sample values are stored in a FIFO that is capable of storing 8192 values. This FIFO allows the four channels to measure simultaneously and then transfer data direct to the PC memory or stream to the hard disk. Also included on this card are two 16 bit D/A converters (DAC0, DAC1). Eight digital input /output lines (Digital I/O). Two 24 bit Counter/Timer I/O, together with a versatile triggering system allowing either analogue or digital signals to start and stop the A/D converters (PFI/Trigger).

Compact PCI

Predominant in the field of industrial test and measurement is the compact PCI system. This standard combines the benefits of the PCI system architecture together with a robust, modular and reliable computer hardware built using the Eurocard rack system and complying with EMC standards.

This Industrial computer is built using 3U and 6U height Eurocards, unlike the cards you would normally plug into an office PC the compact PCI offers outstanding mechanical integrity and simple installation. All of the system cards are easily accessible because they slide out from the front of the computer. The operating conditions of this system are more precisely defined than a standard PC and include maximum shock loading, vibration, air humidity and temperature of its environment. It is anticipated that this standard will be in use for the next 10 to 15 years.
A Compact PCI-solution would be recommended if the following features are important:

Reliable bus/slot connectors.
Vibration and impact resistant card retention mechanism. Using locking screws and card extraction levers.
Robust industrial housing suitable for 19" Rack Mounting
Cooling occurs by natural convection or industrial fans.
Access to the system is always from the front of the unit, including connections to test or interface cables.
This simplifies monitoring of test signals changing cards.
Passive backplane, using an unpluggable CPU. (Compact PCI CPU card) without changing the I/O-Systems (Test card and connectors, or Interfaces)
7 or 13 free Slots for Compact PCI cards. Expandable using bus bridges/bus-expanders.
Runs on Intel-Processors under Windows 95 / 98 / NT / 2000 operating system.
Price range £650 - £3100 approx.

Figure 5. USB Test hardware.

PCMCIA

The PCMCIA (Personal Computer Memory Card International Association) Standard specifies the size, maximum permissible power, signal paths and the programming constraints for cards using this inter-

face. It was specifically designed to allow simple plug-in memory expansion or I/O for portable PC's. There are three different types of cards available:

Type I card with a maximum thickness of 3.3 mm.(originally for memory cards only, too narrow for test equipment hardware).
Type II with a thickness of 5.0 mm.
Type III with a thickness of 10.5 mm.

In portable computers there are often two PCMCIA Type II slots implemented next to each other, this gives space for two type I or type II cards or one type III card. The interface to the PC is well specified.

PCMCIA cards can however be sensitive to other software running on the system e.g. power saving applications ("Power.exe") or multimedia drivers. This can have the effect of slowing down data flow handling between the card and the PC and may cause interruption to the measurement process. These problems can be difficult to detect and fix. Another problem that can occur, especially if you are using Windows 95/98 is that the card is not properly recognised after installation and pops up under "other devices" in the device manager. What follows will probably be many hours of tinkering before the system finally accepts the card. Some notebooks are also very sensitive to the presence or absence of these cards and it is especially important to deactivate the slot when the card is removed otherwise it could lead to system crash.

A PCMCIA solution can only be recommended in the following situations. A USB system would be an alternative solution:

The test equipment supplier allows the interested customer to try the test equipment in the PC system before committing to purchase. This should ensure that most of the problems described above are avoided.
Portable test systems using notebooks running Windows 95 / 98 / NT or 2000 operating system.
The connector between the PCMCIA-test hardware and the sensor connection box does not need to be industrial grade.
Capture of analogue signals with a maximum sampling rate of 100 kHz with a resolution of 12 to 16 bits.
PCMCIA cards have an external slot and can be connected without opening the portable PC.
Price range £210 - £750 approx.

Parallel Port

Many years ago the company Centronics developed an interface standard for printer control, over the years this has come to be used as a general purpose PC interface. This works with parallel data and can have a cable length of 8 m, cable capacitance and crosstalk reduces the usable speed/length of the cable. These days each data line in the cable is twisted with an earth wire allowing a maximum cable length of 3 m. The maximum data rate is largely dependant on the hardware but it can handle up to 1 MByte/s with a cable length of 1 m.

Using the parallel port is not necessarily straightforward and many PC's can crash on boot up if the peripheral device is powered up before the PC is ready. In this case the solution is to turn on the equipment after the PC has booted. Often it is possible to achieve a continuous data transfer by reading the parallel port through the system BIOS. There seems little point recommending a parallel port solution when the USB is the better option.

USB

Virtually every PC based system produced today comes complete with a USB (Universal Serial Bus) connector. This system has been developed jointly by leading companies in the PC industry. This system offers true Plug-and-Play capability to peripheral devices. There is no need to add any additional cards to connect a peripheral device. While this interface standard was originally designed for consumer orientated applications today we find it has a wide acceptance in many other areas of application. The PC based test equipment has benefited from this development.

The USB bus uses a four wire bus cable with connectors of different type at either end. This is because the USB uses a star topology and a loop back connection is not allowed and is not possible with this cable. This interface has a continuous data rate of 12 MB/s with a guaranteed latency and is especially useful for the test environment. This data rate is more than sufficient for most test applications and means that costly local memory storage does not need to be designed into the equipment to store fast test data. The USB cable also supplies power to the peripheral device. So with USB able to supply the power to the device, all that is needed to connect the test device to a PC is a single cable. Using controller software on the PC it is extremely quick to produce a test environment to make the measurements without the necessity of reconfiguring the software or adding a special card to the PC. If the device driver software has already been loaded into the PC then it will automatically recognise that the device has been connected or disconnected to the bus. A USB solution should be considered if the following points are important:

Allows for portable test equipment using notebooks or portable PC-equipment running the Windows 95 Rev. C / 98/2000 operating system.
Wide range of industrial standard connectors between the PC and the USB device.
Capture analogue signals using a sample rate of 100 kHz with a signal resolution of around 12 – 16 Bit
USB Test hardware can be attached simply to the PC no need for a screwdriver here, just plug it in.
Price range £90 - £1500 approx.

ISA

The ISA (Industrial Standard Architecture) Bus allows a transfer rate up to 16.7 MB/s. Test equipment using this interface are becoming more rare largely due to its replacement by the PCI. In most of today's computer systems it is becoming more difficult to find machines supplied with an ISA socket. Investment in test equipment using this interface standard could therefore be risky.
Price range approx. £100 - £5000

The COM Port

Practically every computer system has an RS232 interface and its not surprising that it has been put to uses that were not anticipated when the interface was originally spec-

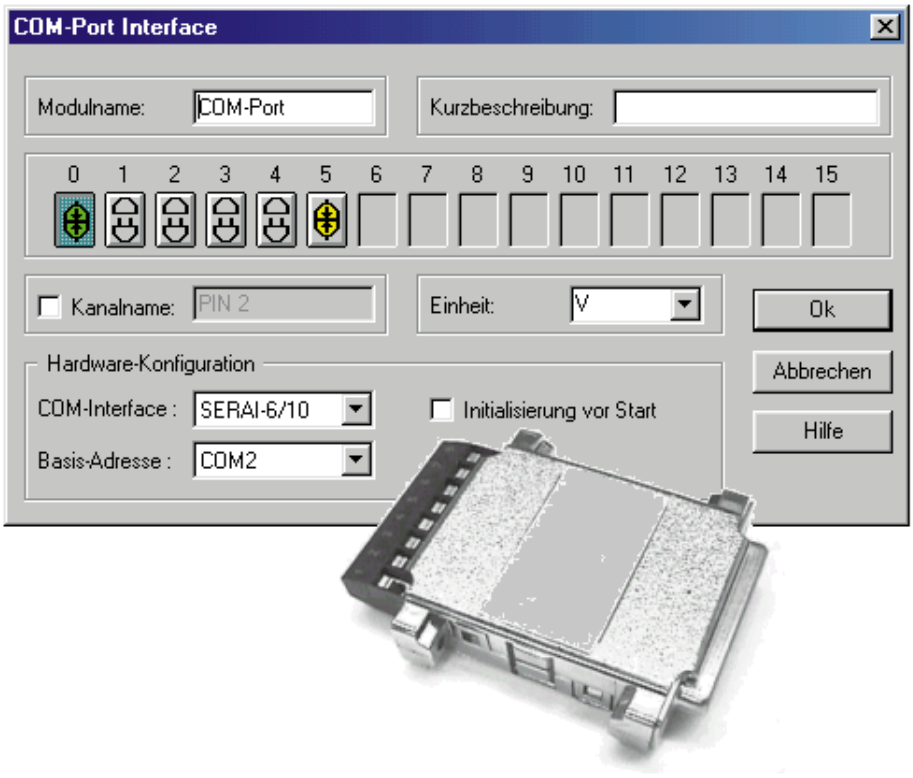


Figure 6. COM port test hardware with DASYLab driver.

ified. The advantage of this port is that equipment developed for use with this interface can be controlled by totally different computer systems, the handshaking protocol can however create some problems with this interface. Test equipment using this port are generally low cost and low speed. A COM port interface can be recommended If these criteria are important in your application:

Economical solution for measuring analogue signals with a sampling rate of 1 Hz – 1 kHz and a signal resolution in the range of 8 - 12 Bits Using the Windows 95/98/NT/2000 operating system.

Test equipment using the COM Port can be connected without opening the PC case.

It uses a relatively slow data transfer rate. This offers very good data protection and generates very low interference over a relatively long cable.

This uses a very simple cable assembly that can if necessary, be provided with galvanic isolation.

Price range approx. £20 - £200

A/D and D/A converters

Another important criterion when choosing the test card is the number of signals that must be recorded at the same time. Together with the question of speed, there is also a difficult decision as to how the measurements are to be made. The most expensive option for fast measurement is to capture each input signal at the same time using a dedicated high speed A/D converter on each input line. The more economical path is to use one A/D converter and switch each channel in sequence into the input of the A/D converter. This last solution is more suited for less critical testing.

Sample & Hold

In critical cases it may be necessary to take a snapshot of the input signals and hold their values in a sample and hold circuit, until the conversion (see the paragraph on speed) has been completed on all input channels. The rule applies here that the effective sample rate of a channel is inversely proportional to the number of channels that are being measured.

Resolution

The resolution of a test system is defined as the smallest change that can be detected by an A/D converter or the smallest output voltage step that can be produced by a D/A converter. The resolution can be given as a percent of the full-scale range or more usefully as the number of bits "n", where 2^n equals the number of levels that can be resolved.

Speed and Sampling Rate

Two important measurement criteria are speed and accuracy. There is always a trade off between them. Speed of measurement is the inverse of the conversion time this is the time that the A/D converter requires to convert the analogue signal into its digital equivalent. The greater the resolution required, the greater this time will be. The width of the data output (e.g. 8, 10, 12 or 16 Bit) will therefore be one factor that influences the conversion time. The conversion process also influences the conversion time choosing a converter that uses a fast conversion technique will obviously be more expensive than one using a slower method.

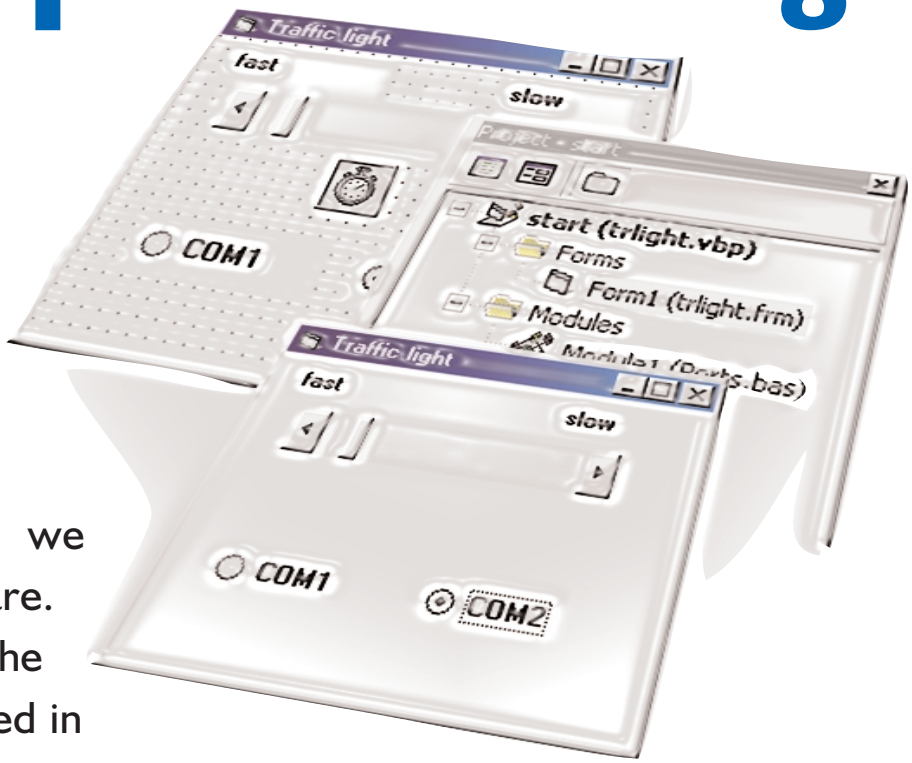
Assuming there are no secondary limiting factors (e.g. processing speed) then the sampling rate of an A/D converter will be inversely proportional to its conversion time. Sampling theory states that to measure a pure sinusoidal signal without losing any information it is necessary to take at least three samples for each period of the wave. However for non ideal signals and to get a better appreciation of what the signal is doing the rule of thumb states that there should be 10 measurement samples for each period of the sampled signal. This means in practice that the fastest analogue signal to be measured should be 1/10 of the sampling rate of the converter.

A practical example will show the constraints of typical test hardware. A test card has an A/D converter with a sample rate of 100 kHz. It has 16 input lines that can be switched to the input of the A/D with a multiplexer. This gives an effective sampling rate of 100/16 kHz = 6.25 kHz per channel. Using the above rule of thumb, this gives us a maximum input signal of 625 Hz per channel. The card in the above example would therefore be suitable for measurement of low frequency AF signals.

000093e

PC Peripheral Design (2)

Part 2: Software



By B. Kainka

In this second part we concentrate on software. We take a closer look at the IOtest program presented in the first article and use simple program examples in Visual Basic to control a model traffic light and a clock generator.

In the first article in this series we introduced the IOtest program. The majority of readers with some programming experience will be interested in how we access the serial interface using Visual Basic. Here we will use a library of program subroutines called PORT.DLL written by H.-J.Berndt. This DLL file is available from the Free Downloads section of the *Elektor Electronics* website at http://www.elektor-electronics.co.uk. For those with no access to the Internet, the file is also found on the diskette containing the course software (see Readers Services pages).

In Visual Basic, all the procedures and functions defined in the DLL are declared and must be in an external module. In our case this is called PORTS.BAS (**Listing 1**).

The experienced (Visual Basic) user will probably recognise the most commonly used routines for serial communications, e.g., OPENCOM which is used to open the interface for communication and indicates that it is available to the program. SENDBYTE and READBYTE are used to transfer data over the serial interface. For our purposes here, the more important routines are those that access the control and status lines of the interface. The procedures DTR, RTS and TXD control these output signals and CTS, DSR, RI and DCD read the state of these input signals. One input line that you cannot read directly is RXD. This is because it would normally be used to carry the received serial data. Also the PORT.DLL file contains routines for time measurement that we will be using later, as well as many other functions for the other PC interfaces (Parallel port, Joystick, Sound and Video).

I/Otest

The application program I/Otest is shown in **Listing 2**.

The essential core of the program is the timer procedure `Timer1_Timer`, which is automatically called at predetermined intervals. The check boxes 1 to 4 are acti-

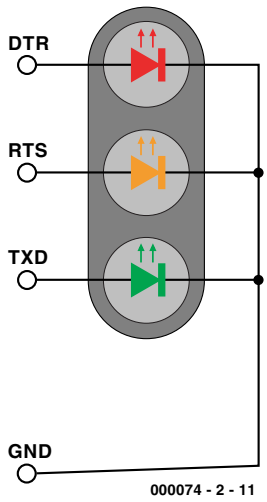


Figure 1. The traffic light LEDs.

Listing 1

Declarations for PORT.DLL

```
Declare Function OPENCOM Lib "Port" (ByVal A$) As Integer
Declare Sub CLOSECOM Lib "Port" ()
Declare Sub SENDBYTE Lib "Port" (ByVal b%)
Declare Function READBYTE Lib "Port" () As Integer
Declare Sub DTR Lib "Port" (ByVal b%)
Declare Sub RTS Lib "Port" (ByVal b%)
Declare Sub TXD Lib "Port" (ByVal b%)
Declare Function CTS Lib "Port" () As Integer
Declare Function DSR Lib "Port" () As Integer
Declare Function RI Lib "Port" () As Integer
Declare Function DCD Lib "Port" () As Integer
Declare Sub DELAY Lib "Port" (ByVal b%)
Declare Sub TIMEINIT Lib "Port" ()
Declare Sub TIMEINITUS Lib "Port" ()
Declare Function TIMEREAD Lib "Port" () As Long
Declare Function TIMEREADUS Lib "Port" () As Long
Declare Sub DELAYUS Lib "Port" (ByVal l As Long)
Declare Sub REALTIME Lib "Port" (ByVal i As Boolean)
```

Listing 2

User program IOtest

```
Private Sub Form_Load()
 i = OPENCOM("COM2,1200,N,8,1")
 If i = 0 Then
    i = OPENCOM("COM1,1200,N,8,1")
    Option1.Value = True
 End If
 If i = 0 Then MsgBox ("COM Interface Error")
 TXD 1
 RTS 1
 DTR 1
 TIMEINIT
End Sub

Private Sub Form_Unload(Cancel As Integer)
  CLOSECOM
End Sub

Private Sub Option1_Click()
 i = OPENCOM("COM1,1200,N,8,1")
 If i = 0 Then MsgBox ("COM1 not available")
 TXD 1
 RTS 1
 DTR 1
End Sub

Private Sub Option2_Click()
 i = OPENCOM("COM2,1200,N,8,1")
 If i = 0 Then MsgBox ("COM2 not available")
 TXD 1
 RTS 1
 DTR 1
End Sub

Private Sub Timer1_Timer()
  Check1.Value = CTS()
  Check2.Value = DSR()
  Check3.Value = DCD()
  Check4.Value = RI()
  If Check5.Value Then TXD 1 Else TXD 0
  If Check6.Value Then DTR 1 Else DTR 0
  If Check7.Value Then RTS 1 Else RTS 0
End Sub
```

vated and will show a tick when the corresponding input line is switched to a '1'. The three output lines are switched when the user activates the corresponding check box.

The other parts of the program are used to select and open the interface and we will look at them a little later. We can see that the interface will be initialised with a communication rate of 1200 baud, no parity bit, 8 data bits and 1 stop bit. For our purposes here, these communication parameters are entirely superfluous because we are only interested in the control lines that form part of the interface. Windows will however configure the port as if it were to communicate with, for example, a modem. But all we need to do here is to read the input status lines and write to the output control lines.

Traffic Light controller

If you are new to Visual Basic it is a good idea to choose some hardware that you can actually see working. For this reason a model traffic light with three LEDs was chosen (**Figure 1**). The effects of current limiting and reverse voltage on the LEDs were discussed in the first article of this series.

The model lamp is well suited to experimenting for our first excursion into programming with Visual Basic. For the program development you will need to start with a new blank form, and using the mouse, select and drag graphic control elements from the list of tools onto the form (**Figure 2**). The size of each element and its position can be easily altered. Each element has a complete range of properties that can be attached to it, including size, colour, text and much more. Those that you are not sure of yet can be simply left as they are. For our application we use the following elements:

- Two labels with the properties (the text) Caption = "fast" and "slow" respectively.

- A horizontal slider or scroll bar (HScrollBar) with the properties Min = 50, Max = 500 and Position =100
- A timer (Timer) with the property Interval = 100, i.e. 100 ms

- Two Option Buttons with the Caption = "COM1" and "COM2" respectively, the button for COM2 is "true"
- The form itself contains the Caption = "Traffic Light"

Figure 3 gives an overview of the project. The aforementioned module PORTS.BAS is also included in the project and contains all the

Listing 3

Traffic lights program

```
Dim Time As Integer

Private Sub Form_Load()
 i = OPENCOM("COM2,1200,N,8,1")
 If i = 0 Then
    i = OPENCOM("COM1,1200,N,8,1")
    Option1.Value = True
 End If
 If i = 0 Then MsgBox ("COM Interface Error")
 TXD 0
 RTS 0
 DTR 0
 Time = 0
End Sub

Private Sub Form_Unload(Cancel As Integer)
  CLOSECOM
End Sub

Private Sub HScroll1_Change()
  Timer1.Interval = HScroll1.Value
End Sub

Private Sub Option1_Click()
 i = OPENCOM("COM1,1200,N,8,1")
 If i = 0 Then MsgBox ("COM1 not available")
 TXD 1
 RTS 1
 DTR 1
End Sub

Private Sub Option2_Click()
 i = OPENCOM("COM2,1200,N,8,1")
 If i = 0 Then MsgBox ("COM2 not available")
 TXD 1
 RTS 1
 DTR 1
End Sub

Private Sub Timer1_Timer()
  Time = Time + 1
  If Time = 1 Then red
  If Time = 40 Then redyellow
  If Time = 50 Then green
  If Time = 90 Then yellow
  If Time = 100 Then Time = 0
End Sub

Sub red()
RTS 1
DTR 0
TXD 0
End Sub

Sub redyellow()
RTS 1
DTR 1
TXD 0
End Sub

Sub yellow()
RTS 0
DTR 1
TXD 0
End Sub

Sub green()
RTS 0
DTR 0
TXD 1
End Sub
```

necessary declarations for PORT.DLL. It needs to linked with the software in each of the projects so that they can all have access to the procedures and functions that are needed to control the serial interface.

In a Visual Basic program, events are produced by procedures that are isolated from each other. The overall flow of the program is controlled by Windows. The procedure 'Private Sub Form_Load()' will be called right at the beginning of the program (**Listing 3**) This procedure contains all the instructions needed to initialise the serial interface. In this case, the serial interface will be open and all the outputs of the interface will be switched off. Lastly, the global variable 'Time' is reset to zero.

The function OPENCOM in PORT.DLL returns a value indicating if the interface has been successfully opened. It will not be possible to open it if it is already in use by another program. Initially the program will use this function to open COM2. If the return value from this function is 0 i.e. it is busy then COM1 will be opened. This will be displayed on the screen with the value of option button 1 (COM1) 'True'. If both COM1 and COM2 are busy then a failure message will pop up in a MessageBox.

In the normal course of events the program will open COM2. If however you want to use COM1 then you can point to the corresponding button and click. This will cause Windows to call the procedure Option1.Click which will open COM1 and check for a successful returned status. This automatic selection and manual override has already been used in the IOtest program and will also used in the upcoming program examples. It would be a simple matter to add extra buttons to control the COM3 and COM4 interface.

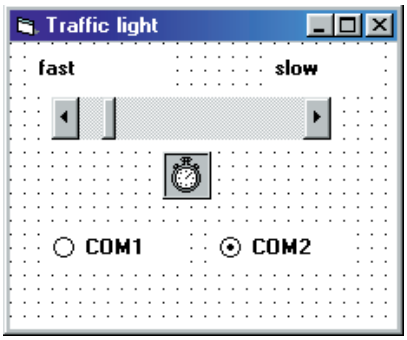


Figure 2. The traffic light controller form.

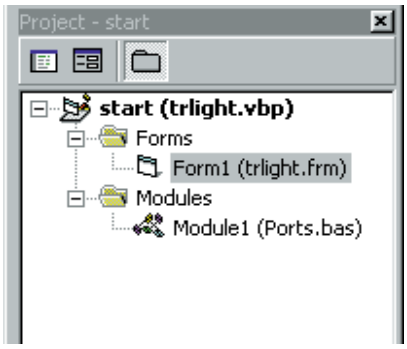


Figure 3. Project overview.

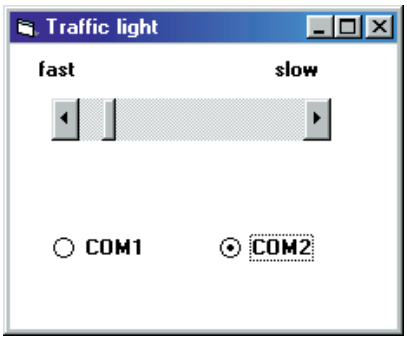


Figure 4. The traffic light program during run time.

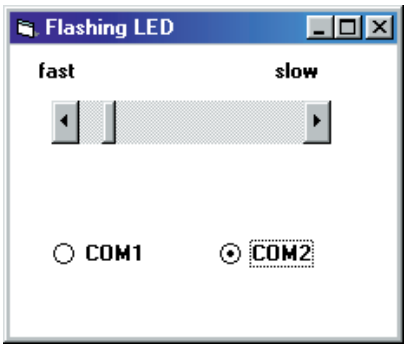


Figure 5. The Blinker Program.

For the traffic light controller we will need some timer function to introduce a delay between changing the lights. In the DLL there is, for example, a procedure called DELAY. However, unlike programming in DOS, individual Windows programs do not have the entire processing time devoted to them. It would be pointless to write a procedure containing a simple timing loop to give us the delays that we need for the traffic light. Instead, the traffic light timer will be event controlled. For this we will use a Windows timer. The timer period is set to 100 ms. The procedure will be called every 100 ms.

In the timer procedure there is a variable called "Time" which is incremented and gives the time in tenths of a second, so the procedure can be called to find out if a specific period has elapsed before, for example, the traffic light LEDs are changed. The speed of switching has been chosen arbitrarily and can be easily altered. The IF statements are used to compare the time values and switch the corresponding LED.

The form (**Figure 4**) also contains a horizontal slider or scroll bar. This slider is used to alter the speed of the traffic light changing. As soon as the slider is moved, the procedure *Hscroll1.Change* is called. This procedure will change the interval in the timer corresponding to the actual position (Value) of the slider control. The slider is calibrated from 50 to 500; the timer can therefore be altered in the range from 50 ms to 500 ms.

Listing 4

Timerprocedure for blink program

```
Private Sub Timer1_Timer()
  Time = Time + 1
  If Time = 1 Then
    RTS 1
    DTR 0
  End If
  If Time = 2 Then
    RTS 0
    DTR 1
  End If
  If Time = 2 Then Time = 0
End Sub
```

Blinker/Clock generator

This next application is a blinker or clock generator with adjustable frequency (**Figure 5**). The circuit is the same as for the traffic light experiment. This time, the TXD output is permanently switched on while DTR and RTS are switched in anti-phase, i.e., when one is on the other is off and vice versa. It is not too difficult to see how this program can be expanded to turn it into a mini running light display.

The software from the traffic light program is used again for this exercise. The timer procedure is, however, new (**Listing 4**). One difference here is that no additional procedures are called to control the output lines, they are switched directly from inside 'timer-procedure'.

The LEDs on the DTR and RTS lines should now blink alternately. The speed of the blinking can be increased or decreased. As you increase the speed you may notice that the timing for each blink is not precisely regular. The reason for this is that Windows cannot maintain a time interval of exactly 50 milliseconds because it is a multitasking operating system many other processes will be running at the same time. For this reason Windows is generally considered to be not capable of 'real time' operation. However, there are some tricks and techniques that we can use and we shall be investigating them in the coming articles.

(00074-2e)

Software Synthesisers

Nostalgia with the MIDI Parameter Box

By Luc Lemmens

In the February issue we described a MIDI Parameter Box, in which the emphasis was placed on the hardware. In this article we will revisit this project, but will now pay special attention to software synthesisers: programs that imitate a '70s era synthesiser on a PC or Mac. The Parameter Box makes the operation of this software remarkably easier.

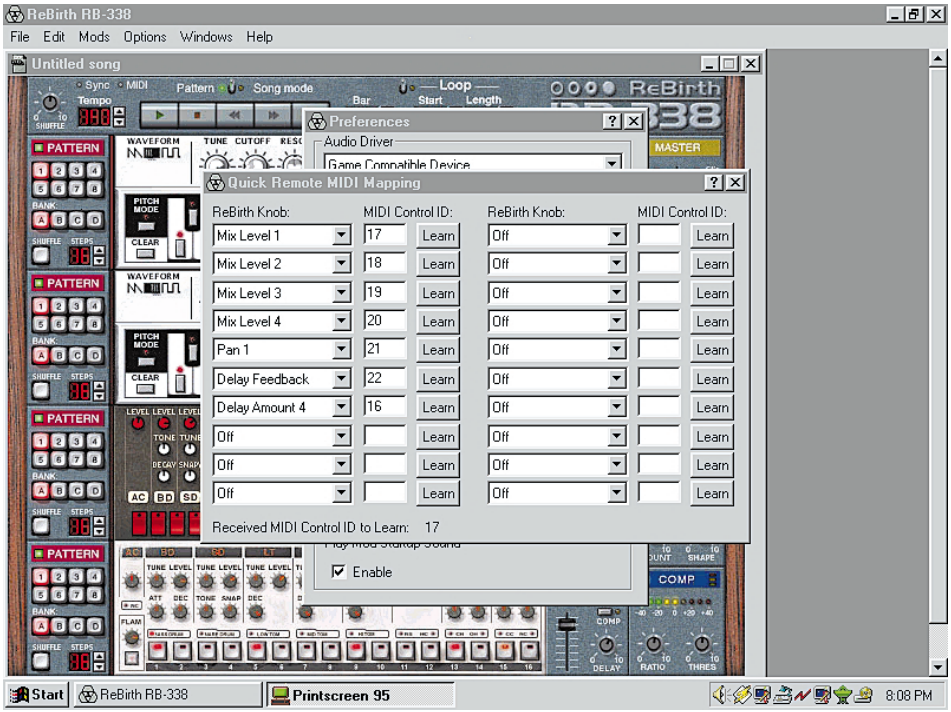


Figure 1. Adjusting the correct MIDI channel number in Rebirth.

In all likelihood, many readers will immediately switch off upon reading the term 'MIDI', thinking it is something for dyed-in-the-wool musicians who will hook up various all-electronic instruments, using this digital interface. This impression in not correct, however. Anyone who owns a PC or MAC with a sound card can play around with the Parameter Box.

To most users, the sound card is just a 'passive' aid to make the (standard) sounds from the operating system audible, play audio CDs or to enliven presentations and games with sound effects. These capabilities are standard features of the existing software. Few people will realise that the computer can also be used as a musical instrument. Excellent programs exist, called *software synthesisers*, that enable even beginning musicians to play capably and make full use of all the features that a modern sound card has to offer.

Back to the past...

We have in mind the synthesisers of the same generation as the legendary **Formant** published in *Elektor Electronics* during the '70s and '80s. They consisted of large boxes, filled with adjustable oscillators, mixers, VCOs, VCAs and filters, which could be interconnected in various ways with patch cables to produce the strangest, and often unexpected, sounds and sound effects. The musician could set the controls of any individual module (such as frequency, amplification, modulation, mixing levels, etc.) with a whole array of buttons and sliders. These days, things can be made much simpler, in so far as it affects the hardware, with software synthesisers such as Rebirth from Propellerhead or Dynamo from Native Instruments. The patch panel and adjustments can now be found on the computer screen and the generation of sound effects is delegated to the software and the sound card.

The hardware may have changed, but the operation still demands as much skill as was necessary some 25 years ago. The musician is still required to operate the buttons and sliders and herein lies the weakness of the 'bare' software synthesiser for a computer. The adjustments are made using the mouse, and given the fact that a computer with two mice has yet been developed, we effectively have to operate the control panel with one hand tied behind our back. In addition, reaching for a real button is still faster and simpler than the slide and click of a mouse. And here is where the parameter box comes in: we have our buttons back!

The parameter box provides us with trusty, physical potentiometers and buttons, all within actual reach. However, one way or another, they need to be connected to the virtual control panel on the monitor. Fortunately, this is simple to achieve. The software synthesiser has extension capabilities that its illustrious predecessors lacked (for the simple reason that it did not yet exist): a MIDI interface.

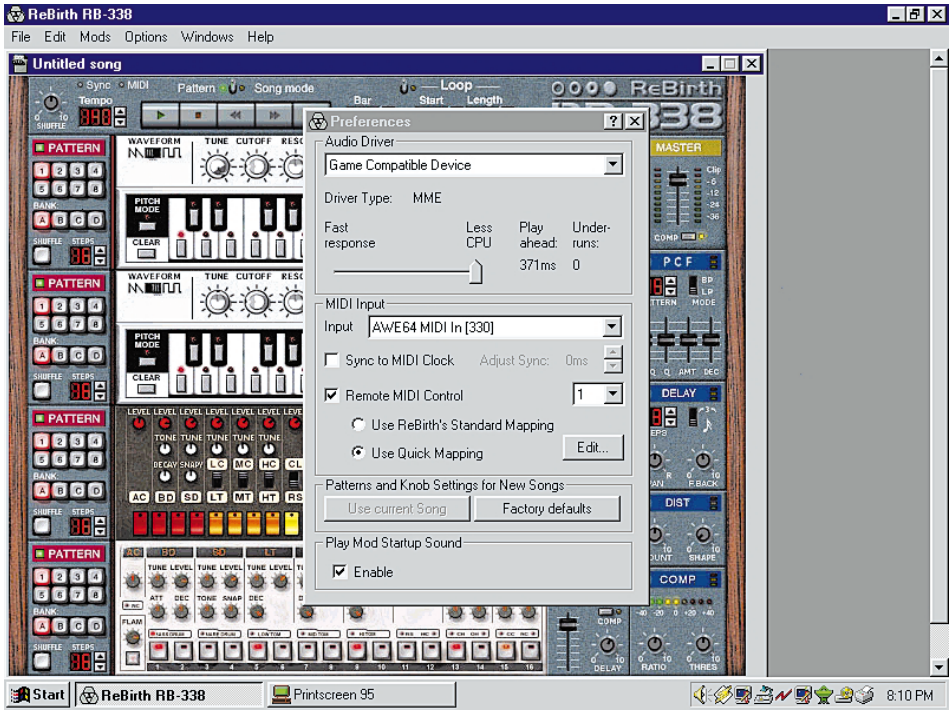


Figure 2. Linking the potentiometer with a button on the screen.

In this article we will not deal with the intricacies of MIDI, this has been sufficiently covered in previous publications and books from Elektor Electronics (Publishing). Here, the only important thing to know is that using MIDI, it is not only possible to send musical notes but also control codes for modern synthesisers. And that is exactly what the parameter box does: sending codes that are received and interpreted by the software synthesiser. Obviously, we first need to 'teach' the synthesiser which physical button belongs with which virtual adjustment. After that, we need only fool around with the buttons to enable us to play the instrument quickly and comfortably. This learning process is explained in the description for each of the two software synthesisers, a bit further on.

Naturally, we are limited to providing technical help with the interconnection of the parameter box and the software synthesisers. Arranging the synthesiser, the actual operation and the playing are beyond the scope of this article. This learning process we leave gladly to the ingenuity and musical ability of the reader. Even if you do not have the pretension or intention to compete with Jean Michel Jarre, it will still be a rewarding experience to put that sound card to work yourself. A small warning is appropriate: we are dealing here with two very professional software packages. So do not expect to quickly master these synthesisers.

Demo versions of these software packages may be downloaded from the Internet for free. So 'professional' need not mean that you

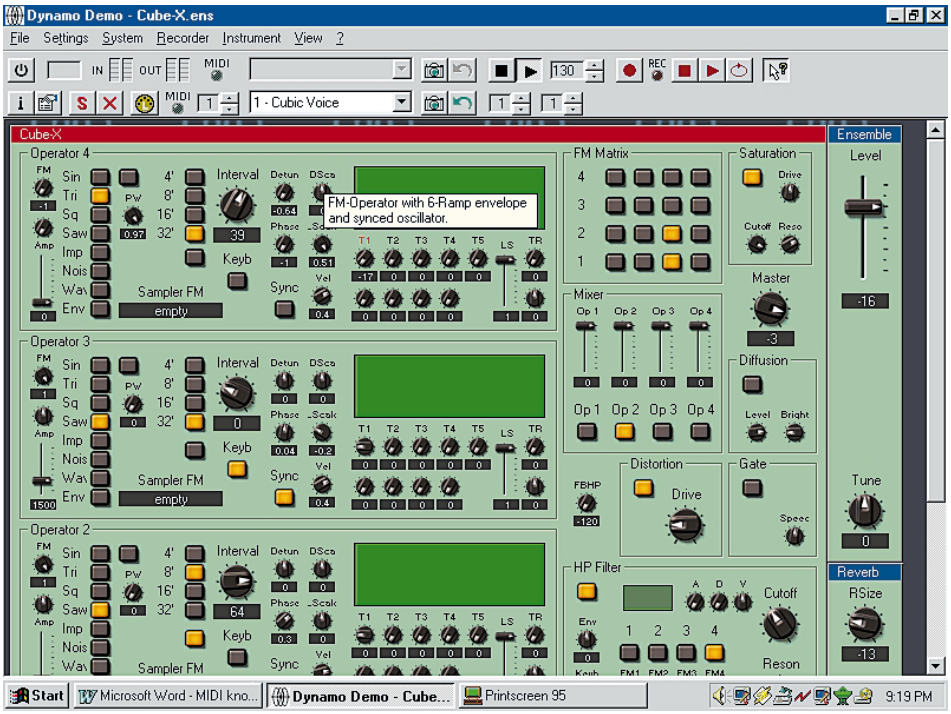


Figure 3. The Dynamo software, the Learn button is at top right (a DIN-symbol with an L).

immediately have to reach deeply into your wallet to become acquainted with it.

Rebirth

The most recent demo version of this synthesiser from **Propellerhead** is version 2.0. It may be possible, with a little effort, to find version 1.5 on a mirror site. Demo version 2.0 may be found on *www.propellerhers.se* and works for 15 minutes at a time — by contrast, version 1.5 used to work for half an hour. In addition, a number of options, available in the full version, have been disabled. Fortunately, the control panel settings, the topic of this article, are automatically saved when time is up. Therefore, it is not necessary to start all over again each time. The complete version of this software costs 159 Euro.

After starting the software, a song has to be selected (file extension .RBS), after which the buttons from the parameter box may be linked to the buttons on the screen. In the menu Edit|Preferences, check 'Remote MIDI control' and select the radiobutton 'use quick mapping'. Make sure that the MIDI channel number (a number between 1 and 16) corresponds with the channel selected on the parameter box (see **Figure 1**). Following that, click on the Edit button. The most recent data that has been received from the MIDI controller, via MIDI-in, is displayed at the bottom of the window that pops up. The control code should change when you turn a different potentiometer. Assign this control code to a control from the synthesiser by selecting this from the list and clicking Learn. A potentiometer on the parameter box is then linked to a button on the computer screen, as shown in **Figure 2**. Repeat this process until all the desired controls are assigned. If necessary, select another layer using S1-S6 to enable a maximum of 48 buttons to be controlled from the parameter box. It may be a good idea to make a list or table showing which potentiometer in which layer is assigned to a particular slider control.

Initially, it may be difficult for the beginner to associate the name of a control with the 'physical' adjustment on the screen and its effect. However, after a little experimentation and practice you will quickly get the hang of it.

Generator/Dynamo/ Reaktor/Transformator

Just before this article was due for printing, it was realised that Native Instruments (*www.native-instruments.de*), the creators of **Generator** and **Transformator**, stopped the sale and support of these products. They have been replaced with **Dynamo** (DM 299), a beginners version, and **Reaktor** (DM 799) for advanced users. Both software packages are available as demo versions from their website. Besides the similar limitations of Rebirth (works for 15 minutes at time, for example) there is a very irritating 'bonus' to encourage the demo user to purchase the full version: every few seconds an unwanted note is added to the music…

In comparison with Rebirth, the co-operation with the Parameter Box is very simple to set up. It is sufficient to click on the desired control, followed by clicking the Learn button on the tool bar and turning the desired potentiometer. The Learn button is identified by the 5-way DIN connector symbol with a capital letter 'L', also shown in **Figure 3**. Again, pay attention to the MIDI channel number from which the synthesiser expects to receive the control codes. This is to the right of the Learn button and obviously has to be the same as the MIDI channel that has been selected on the Parameter Box.

Conclusion...

After some fooling around, you will either have had enough of it or decide to purchase the complete version of the software. The limited play time and the omission of some vital functions (such as Save) will no doubt prevent you from doing serious work with this software.

There is obviously much more to the operation of software synthesisers, but perhaps this brief introduction has awakened sufficient interest to further explore the land of software synthesisers yourself.

(000068)

One-Minute Light

Elementary Circuit

by B. Kainka

Loss-free switching, using a Field Effect Transistor (FET) and a capacitor's discharge function, is the base theme of this elementary circuit.

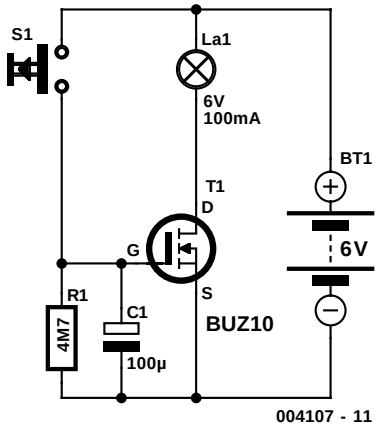


Figure 1. One –minute light circuit diagram — there are just of six components.

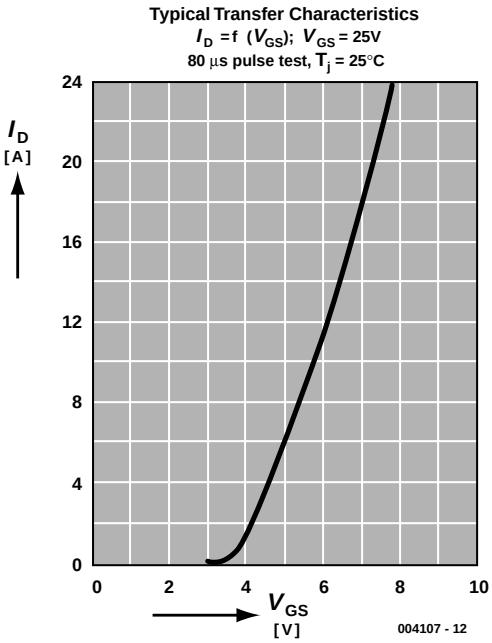


Figure 2. The BUZ10 Field Effect Transistor Transfer characteristics.

The one-minute light circuit shown in **Figure 1** consists of six components, whose functions are easy to point out. The power source is a 6-volt battery, paralleled to the branch point with a lamp and transistor, acting an on/off switch.

Likewise, parallel to the battery is a node linking a pushbutton switch coupled to a very high ohmic valued resistor. The BUZ10 is a Field Effect Transistor (FET), which differentiates itself from a normal transistor, in so much that almost no base current is needed for it to switch on — instead, a voltage at the Gate pin is required. Thus switching without performance loss is possible, which is extremely practical for battery switching circuits.

The voltage potential between the pushbutton switch and resistor determines whether current passes to the lamp through the switching transistor or not.

Standby mode

In standby mode the pushbutton is open. The capacitor is discharged through the resistor. The transistor then blocks, allowing no trickle current to flow to the lamp.

Switch mode

One push on the button instantaneously brings a potential of +6 V to the node. This voltage is also seen at the Gate of the Field Effect Transistor BUZ10. As the Transfer characteristics of this transistor in **Figure 2** shows, when the Gate/Source-Voltage V_{GS} is +6 V, a maximum current of 12 A flows through the Drain-Source junction. Strictly speaking, the diagram is only valid for a Drain-Source-Voltage of 25 V. Actually, such a high current does not flow, due to on the one hand the battery having an relatively high internal resistance and on the other hand the sudden resistance change of the lamp filament from a low ohmic value in its cold state to about 60 Ω in a few milliseconds as the filament glows hot. The current flowing through this branch is easy determined using Ohm's law.

$I = U/R$ = 6 V / 60 Ω = 100 mA

By the way, the potential loss across the conducting FET is marginal thanks to the low drain-source resistance of less than 0.1 Ω.

Discharging

Pushing on the button, something else also happens, the capacitor gets a kick-start charge of +6 V. When releasing the pushbutton, this potential remains at the Gate of the BUZ10, due to the fact that the capacitor can only discharge itself very slowly through the high ohmic value resistor. The voltage across the capacitor u_e decreases not linearly, but rather in an exponential fashion:

$u_c = U \cdot e^{-t/RC}$

Here, U is the initial capacitor potential and e the base of natural logarithms, or 2.718. The corresponding curve (Voltage against Time) is shown in **Figure 3**.

Due to the fact that the capacitor is slowly discharged, and the BUZ10 conducting sufficiently until the Gate voltage drops to about 3.75 V, the lamp glows a short spell after the pushbutton is released.

The afterglow effect can even be determined exactly, that is, mathematically. This involves solving the above formula in the time domain and using an instantaneous voltage of u_c = 3.75 V:

$$
\begin{aligned}
t &= RC \cdot \ln (u_c/U) \\
&= -(4{,}7 \cdot 10^6 \Omega \cdot 100 \cdot 10^{-6}\ \mathrm{F}) \cdot \ln (3.75\ \mathrm{V}/6\ \mathrm{V}) \\
&= -470\ \mathrm{s} \cdot \ln 0.625 \\
&= -470\ \mathrm{s} \cdot -0.47 \\
&= 221\ \mathrm{s}
\end{aligned}
$$

This theoretical value will however not be attained due to the fact that capacitors (especially electrolytic ones) also discharge themselves.

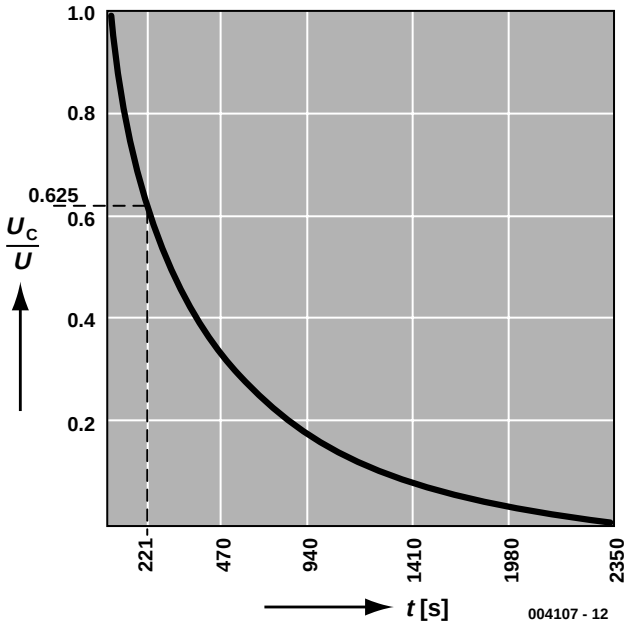


Figure 3. The discharge curve of an ideal capacitor.

(004107-1)

Märklin Digital Model Train Control (2)

Final part: software, construction and operation

From an idea by J. Schröder

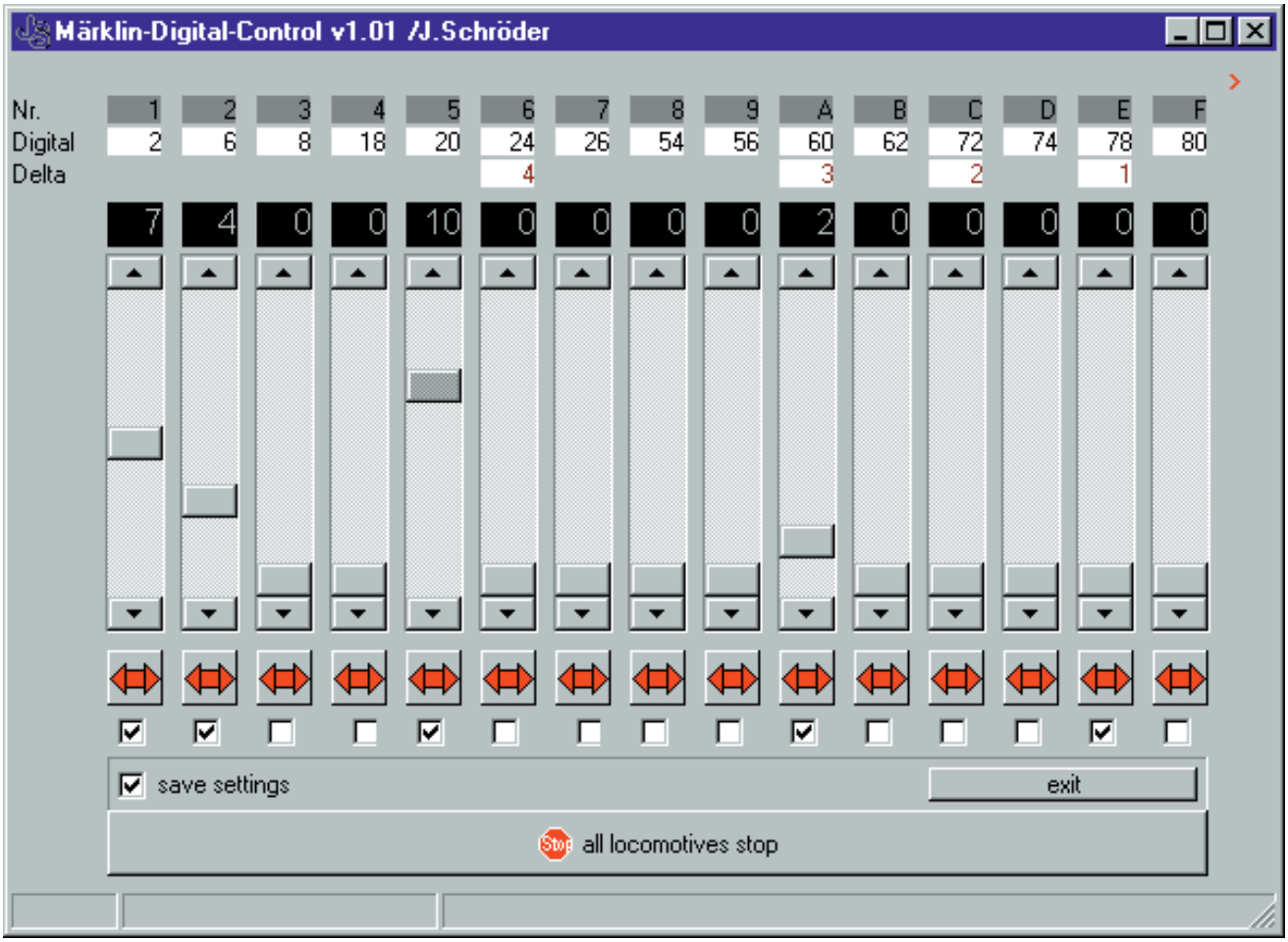


Figure 3. Screenshot of the Windows program written for the train control system.

The Software

In addition to this circuit, a PC is necessary onto which the requisite Windows program needs to be installed. Continuing in the spirit of simplicity, manual controls are not provided. The system requirements have been kept to a minimum. Practically any PC should be capable of performing the task. This applies also to the minimum required version of Windows. The time-honoured Windows 3.1x meets the needs admirably. In addition, provision has been made to make keyboard operation possible. Experience shows that, once familiar with the keystrokes, this method of operation is extraordinarily convenient and fast.

After starting the software, a screen will appear as depicted in **Figure 3**. For language purists, there

is the built in facility to customise the labels of the buttons to your heart's content.

At the top are the (fixed) loco address configurations: first the Märklin loco address, below that, the Delta loco address of the four applicable controls.

The slider (can also be operated with the arrow keys) speaks for itself, it is used to adjust the speed. The small window above the slider indicates the selected speed.

The purpose of the button below that is to select the direction of travel (toggle function). A marginal note is in order here. At certain loco speeds, this button will cause instant reversal of speed. At higher speeds (approximately speed level 7 and up) this abrupt reversal is disabled, perhaps for the wellbeing of potential model railway passengers. In this case, the speed must first be reduced to a lower value, or zero, before carrying out the reversal of direction.

The tick box at the bottom is used to enable or disable the control. The response time of the system is faster when fewer controls are active.

The button 'Save Settings' saves the configuration of the active controls in the file *mrkln01.ini*, which is in the same directory as the one from which the program was started. Note that the current position of the loco control is not stored. The stop button (also operable by hitting the space bar) immediately forces all controls to zero. In contrast with the stop button on the PCB, this stop function will maintain power to the rails.

A number of functions, including which parallel port to use (LPT1: or LPT2:) are fixed in the file *mrkln01.ini*. The details are explained in the box titled 'Software Operation'.

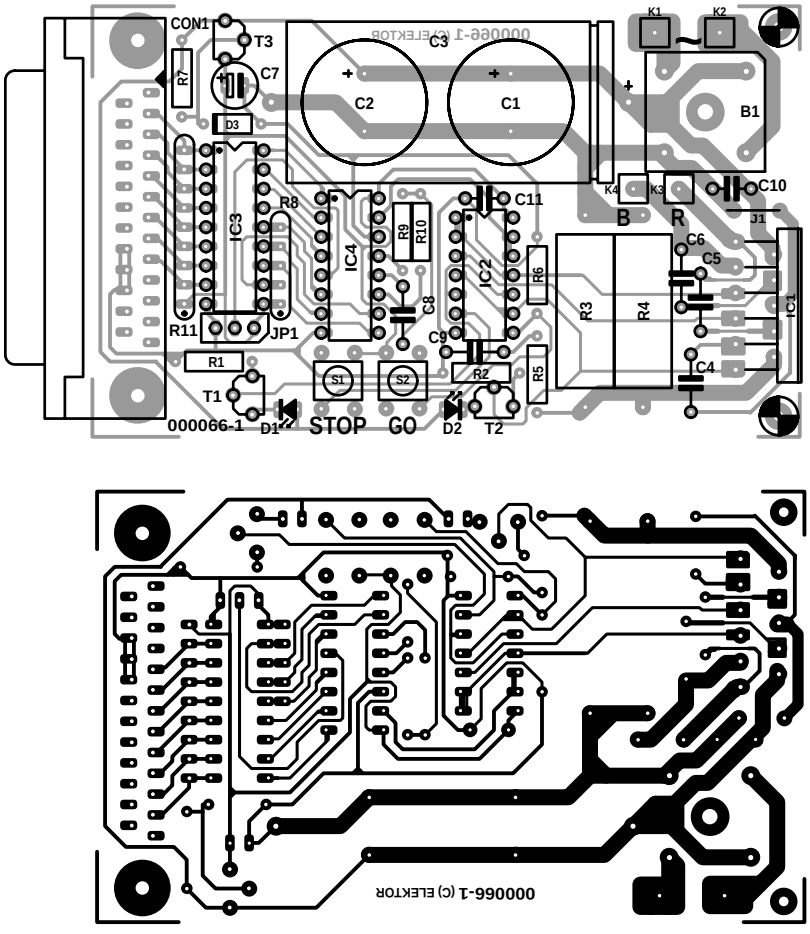


Figure 4. PCB layout and component overlay.

Construction

With the aid of the component overlay and the layout of the single sided PCB (**Figure 4**), you will not need to resort to witchcraft in order to build a functional circuit. A few construction hints, though.

There is a single wire link just above IC4. The trimmed-off part of a lead from C1, C2 or C3 may be used for this. Use this in preference to a lead from one of the resistors, since

COMPONENTS LIST

Resistors:
R1,R2 = 560Ω
R3,R4 = 1Ω5 5 watts
R5,R6 = 10kΩ
R7 = 2kΩ2
R8 = 4x10kΩ SIL9 array, 1 common
R9 = 22kΩ
R10 = 47kΩ
R11 = 8x10kΩ SIL9 array, 1 common

Capacitors:
C1,C2 = 2200µF 40V radial
or C3 = 4700µF 35V axial
C4 = 220nF MKT
C5,C6 = 15nF MKT
C7 = 10µF 16V
C8 = 1nF MKT
C9 = 10nF
C10,C11 = 100nF ceramic

Semiconductors:
B1 = KBPC601 (6A bridge, International Rectifier)
D1 = LED, red, 3 mm
D2 = LED, green, 3 mm
D3 = zener diode 5V6 400mW
T1,T2,T3 = BC547B
IC1 = L6203 (ST Microelectronics)
IC2 = 4001
IC3 = ULN2803A
IC4 = MC145026 (Motorola)

Miscellaneous:
S1,S2 = pushbutton with make contact (e.g., Diptronics DTS-6XX)
CON1 = PCB mount 25-way sub-D plug (male)
JP1 = 3-way SIL pinheader with jumper, or changeover switch
Heatsinking material for IC1 (e.g. aluminium bracket, min. thickness 2 mm)
PCB, order code **000066-1** (see Readers Services
3,5"-inch floppy disk, Windows control software, order code **996016-1**

Optional:
Power supply transformer, 15 V / 5 A, as an alternative for the Märklin transformer

this one is slightly thicker and the link carries the entire output current.

The PCB offers the option to use two radial (upright) electrolytic filter capacitors (C1 and C2) or a single axial one. C1 and C2 may both be 2200 μF. This is sufficient, but larger ones (2 $\times$ 3300 μF or 2 $\times$ 4700 μF) are permissible and will also fit. A single axial electrolytic capacitor (C3), as used in our prototype, is also a possibility. The operating voltage was deliberately selected to be on the high side (35 V minimum) to ensure that the circuit will survive if the Märklin transformer is inadvertently switched to direction-reversal.

CON1 is the connector with which the circuit is connected to the parallel port of a PC. This may be mounted directly to the PCB. Take care that the force of plugging or unplugging the connector is not transferred to the solder connections. To ensure mechanical strength, there exist connectors with holes for mounting screws or a kind of barb for soldering to the PCB. Either type fits. If you prefer not to connect the circuit directly to the PC, but position it closer to the railway, you can use a DB25 extension cable (available for a small outlay in every computer store). Alternatively, you can solder the wires of such a cable directly into the circuit board.

S1 and S2 are switches with momentary make contacts. The switches specified in the parts list fit directly in the PCB. However, other types may also be used if they are connected with a short length of wire.

A similar story applies to JP1, which switches the function on or off. Three pins in a row and a jumper are satisfactory, because the function is usually switched on. Miniature PCB mount switches with changeover contacts also exist. Of course, an external switch attached with hook-up wire is also possible.

The tolerance of C8 is critical, because it defines the timing. Use an MKT-type (Siemens or similar) capacitor.

The rectifier bridge (a very common type from International Rectifier) will, in the event that maximum output current is demanded, appreciate a small amount of heatsinking. Prior to fitting onto the board, insert an M3 bolt in the mounting hole of the bridge (the screw head is at the bottom of B1). B1 may now be soldered onto the PCB. Finally, make a thermal link, using a right-angle section of aluminium extrusion, to the right angle heatsink onto which IC4 is mounted (refer to **Figure 5**). The hole in the bottom of the circuit board allows access so that the screw can be tightened. Don't put the screw through both the rectifier bridge and the PCB or you will run the risk that, when tightening the screw, the circuit board traces get peeled off the board. Alternatively, you can mount B1

Tabel I. Loco addresses and train speed/reversing codes

control	loco address	Delta address	A1-A4
1	2		X 0 0 0
2	6		0 X 0 0
3	8		X X 0 0
4	18		0 0 X 0
5	20		X 0 X 0
6	24	4	0 X X 0
7	26		X X X 0
8	54		0 0 0 X
9	56		X 0 0 X
A	60	3	0 X 0 X
B	62		X X 0 X
C	72	2	0 0 X X
D	74		X 0 X X
E	78	1	0 X X X
F	80		X X X X

Data bit	Coding	Result
A5	0	function (lights) off
	1	function (lights) on
A6..A9	0 0 0 0	stop
	1 0 0 0	reversing command
	0 1 0 0	speed level 1
	1 1 0 0	speed level 2 (etc.) …
	1 1 1 1	… speed level 14

0 = logic zerol
X = logic open

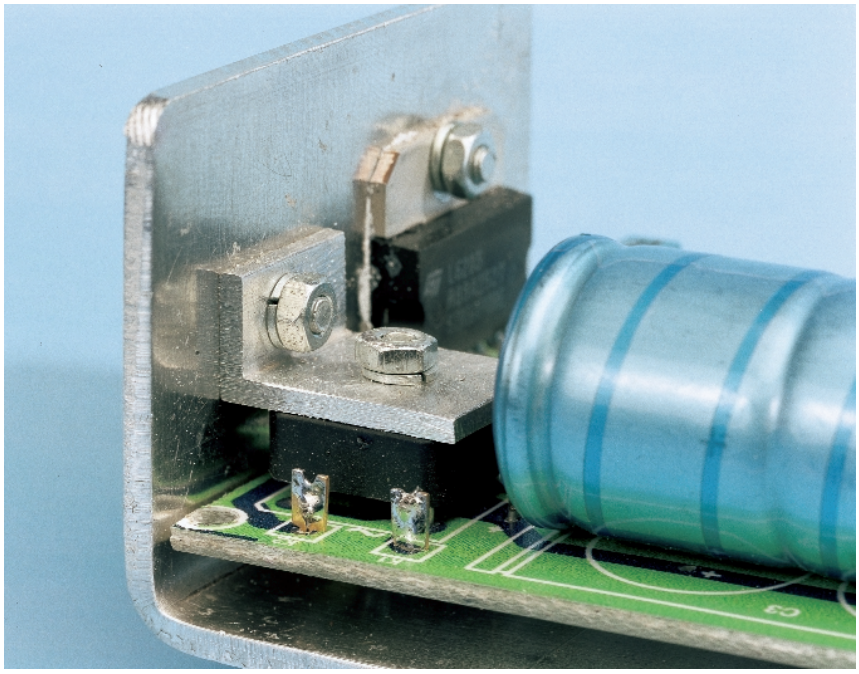


Figure 5. Construction of the thermal link for the bridge rectifier.

Software Operation

Key combinations
The program may be operated without using the mouse (applicable to mrkln01.exe V1.01)

Key (or combination)	result
<TAB>	next loco control
<SHIFT><TAB>	previous loco control
<1>-<9>,<A>-<F>	direct selection of relevant loco control
=	repeat same key, train reverse
<CTRL><1>-<9>, <A>-<F>	enable / disable relevant loco control
cursor up/down keys	increase / reduce speed
<0>	speed 0, immediately on selected control
<space>	speed 0, immediately on all controls (software emergency stop)
<ALT><F4>	quit program

Settings in file mrkln01.ini
The file mrkln01.ini (residing in the same directory as mrkln01.exe) contains information on currently used loco controllers (loco addresses), the printer port used, and the text strings inside the control buttons. Comment not included in the .ini file itself is shown to the right.

```
[PARAMETER]
CH1=0                          0 = control not in use
CH2=0
CH3=0
CH4=0
CH5=0
CH6=1                          1 = control in use
CH7=0
CH8=0
CH9=0
CH10=0
CH11=0
CH12=1
CH13=0
CH14=0
CH15=0
ADDRESS=888                    printer port; 888 = LPT1:, 632 = LPT2
USE_AD=0                       not used
SAVE_SETTING_TEXT=save settings text may be edited
EXIT_TEXT=close                to requirement
STOP_TEXT=stop all trains
```

directly to the heatsink and bend the connecting leads in a right angle; they should be long enough.

IC4 is also mounted on the heatsink. Electrical isolation is not required (the metal part is connected to the GND pin), but heat conducting paste is required. Remember that none of the AC inputs (K1 and K2) are connected to ground.

Powering up

It is always exciting the see whether or not the result of your industrious activity transforms into smoke when powering up, especially when 'power' is involved.

To be safe, it is always good practice to check the power supply voltage first. Connect the circuit terminals K1 and K2 to a Märklin transformer. By initially using the brown and red terminals (instead of the yellow) you can make a cautious start with a lower voltage.

The standard Märklin transformer is rated 30 VA. This is too small to deliver the maximum output current of 3.5 A. However, actual usage will indicate if it is sufficient for your normal use. The 50 VA 'lighting transformer', with its fixed 16 V output voltage, may be more appropriate. Naturally, any other 15 V transformer can also be utilised. From a safety point of view, we strongly discourage connecting the secondary windings of different transformers in parallel.

First check the voltage across C1 and C2 (or C3); this may be 20-25 V at the most. Then check the logic power supply, for example, between pin 8 and pin 16 of IC4. This must be between 4.8 V and 5.2 V; the exact value is not critical. If everything is in order, then the entire circuit may be connected to the PC and the tracks, using the connections B (brown, outside rails) and R (red, centre rail).

After switching on, the circuit will normally be in stop-mode. Push the appropriate button to activate the run-mode. Voltage is now applied to the tracks. In case of an overload, the circuit will automatically switch itself to stop-mode. Install the software by copying the two required files (*mrkln01.exe* and *mrkln01.ini*) to a directory of your choice (they must both be in the same directory). The file *mrkln01.ini* contains the definition specifying which printer port the circuit is connected to. This file also contains the text labels for the operating controls. After starting up the software, the screen will be as shown in **Figure 3**. Operation with a mouse is self-explanatory, for operation from the keyboard we refer you to the appropriate box. If it doesn't work, check first that the correct printer port (888 = LPT1:, 632 = LPT2:) is selected in the *mrkln01.ini* file. Another possibility is incorrect timing of the encoder. In this case you will need to tweak the values of C8 or R9. Those who hunger for more output power may connect terminals R and B directly to the input of the EEDTs-booster. If the tracks are fed only via the booster, then additional cooling for IC4 and B1 is not required. A few ground issues to keep an eye on, particularly if the EEDTs-booster is connected. Mains earth is connected through the PC to ground of the circuit (negative terminals of C1, C2, C3). Because the output is a full bridge, output B is **NOT** connected to mains earth. The R and B terminals of the EEDTs-booster must be allowed to float with respect to mains earth and this applies to the entire railway as well. When connecting (grounded) measuring equipment this has to be taken into account.

(000066-2)

Thinkquest

students make Internet projects

By Harry Baggen

Thinkquest is an organisation that aims at promoting the use of computers and network technology in education. One of the means applied by Thinkquest to achieve its goals is an organized competition in which students can create their own website, in different categories.

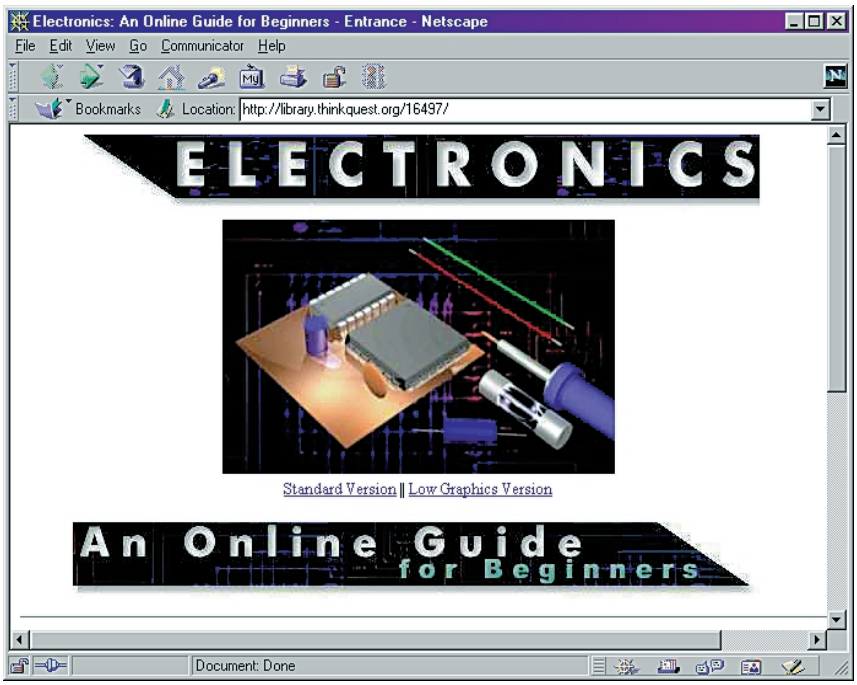
Thinkquest [1] was founded in 1996. Since then, their annual competitions have attracted over 60,000 participants. There are separate programmes for students (Internet Challenge) and youngsters (Thinkquest Junior). Especially the student entries are often beautifully styled and structured and certainly worth having a look at. All competition entries are available and freely accessible on Thinkquest's website. We picked out a few that we feel should be of interest to our electronics-minded readership.

In 'Electronics, an online guide for beginners' [2] we find a good general-purpose introduction into the world of electronics. A little theory, the structure of a circuit diagram, the workings of components, soldering together a circuit and a variety of circuits are all discussed.

The Amateur Radio Web Site [3] is aimed at budding radio amateurs. Although the descriptions of the radio licences are applicable to the US situation only, these pages contain lots of nice information.

The Soundry [4] teaches us a lot about sound. The workings of the human ear, a physics-class explanation of the phenomenon 'sound', applications of sound and a history of sound-related discoveries and developments are the ingredients of this site. You'll a beautifully designed interactive sound lab in which you turn the controls with your mouse to make all kinds of sound effects audible.

The last Thinkquest site we'll discuss here was elected 'best site of the year 1999'. Electricity Online [5] is a kind of course on 'All Things Electrical'. The course covers electrostatics, magnetism and basic electronics.



Besides these subjects, many applications of electricity are discussed, including basic building blocks like a relay, but also telecommunication equipment and computers. E also found a section on electrochemistry (how does a battery work?). In the electricity generation department, the subjects include solar cells, turbines and transformers. Finally, a small section is reserved for history and many other things related to electricity in one way or another. There's even an Electricity Game!

(005117-1)

Internet addresses

[1] Thinkquest:
http://www.thinkquest.org
[2] Electronics:
http://library.thinkquest.org/16497/
[3] The Amateur Radio Web Site:
http://library.thinkquest.org/17234/
[4] The Soundry:
http://library.thinkquest.org/19537/
[5] Electricity Online:
http://library.thinkquest.org/28032/

urls available as hyperlinks on the Hyperlinks page of the Elektor Electronics website.

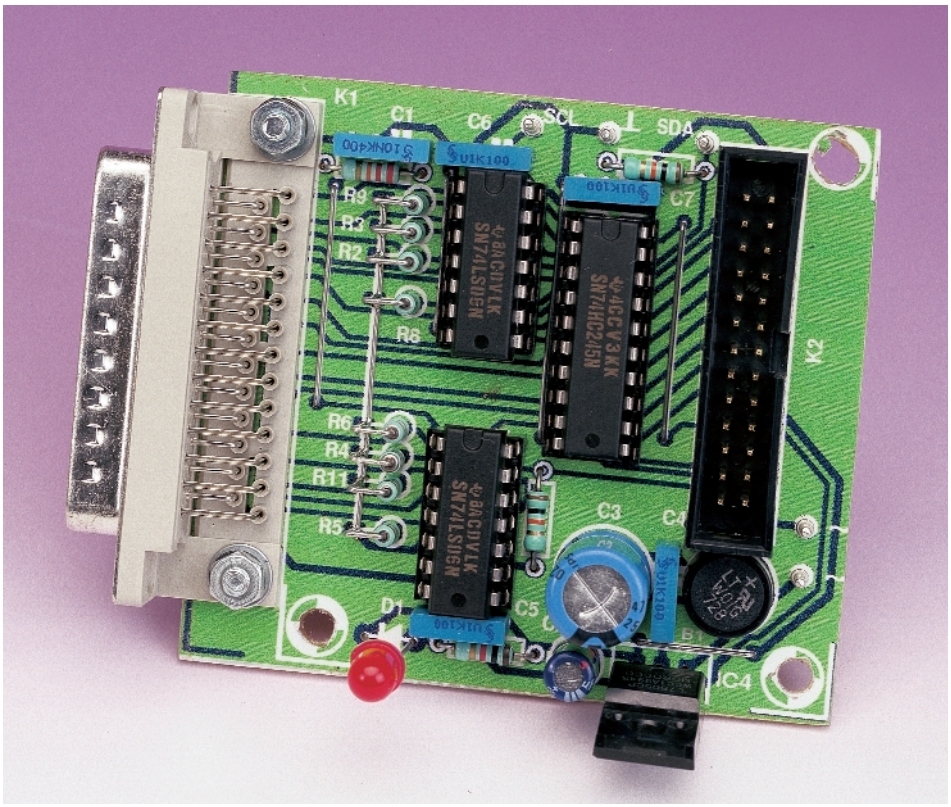
Versatile Centronics/I²C Interface

with logic analyser

by Ingo Gerlach DH1AAD

e-mail: IngoGerlach@welfen-netz.com

When this interface is connected to an extended parallel printer port (EPP), your PC can communicate using the I²C protocol, and it can also make direct read and write accesses to eight data bits.



Most of the I²C interfaces that have been described in *Elektor Electronics* up to now are based on a Philips application note. This circuit is no exception. However, its special feature is the combination of an I²C interface with an 8-bit parallel interface. The parallel port of the PC can be used for control, since 8-bit output was defined for this port by IBM from the very beginning. However, since the data lines were only intended to be used for output, it used to be necessary to resort to a trick in order to read in data. This consisted of using the status port of the interface to read data. Since the status port is only five bits wide, data had to be read in 4-bit nibbles.

With the development of new modes such as EPP (enhanced parallel port) and ECP (extended capabilities port), it became possible to directly read in 8-bit data via the data bus. Following the introduction of these modes, a steadily increasing number of peripheral devices (such as scanners) utilise the parallel interface for communications with the PC, since a suitably high data transfer rate can only be achieved if the data are read directly.

Hardware

The hardware of the I²C/8-bit interface, as shown in **Figure 1**, consists only of a 25-pin connector and a few driver and inverter gates. I²C communications take place via gates IC3b and IC3d, IC3e and IC3f. SDA and SCL are driven via the two data lines D0 and D1, and they are read via UserB2 (pin 15) and Intr (pin 10), respectively, on the EPP interface connector K1. A few pull-up resistors provide the proper voltage levels. The clock and data lines of the I²C bus are located on pin 3 (SCL) and pin 5 (SDA) of the combination pin connector K2, and they are also brought out to solder posts, along with ground.

The PC can determine whether an adapter is connected (*init iic*) via the loopback connection formed by gate IC2b between pin 17 (nAStrb) and pin 13 (XFlagUser3). The bi-directional 8-bit bus driver IC1 is activated via pin 16 of the control port

EPP interface pin assignments

Pin	Function
1	nWrite
2	Data D0
3	Data D1
4	Data D2
5	Data D3
6	Data D4
7	Data D5
8	DataD6
9	Data D7
10	Intr
11	nWait
12	UserB1
13	XFlagUser3
14	nDStrb
15	UserB2
16	nInit
17	nAStrb
18...25	Masse

Output connector K2 pin assignments

Pin	Function
1	Strobe
3	SCL
5	SDA
7	Ground
9	+5 V (Output)
11	Data D7
13	Data D6
15	DataD5
17	Data D4
19	Data D3
21	Data D2
23	Data D1
25	Data D0
2	PE (Input)

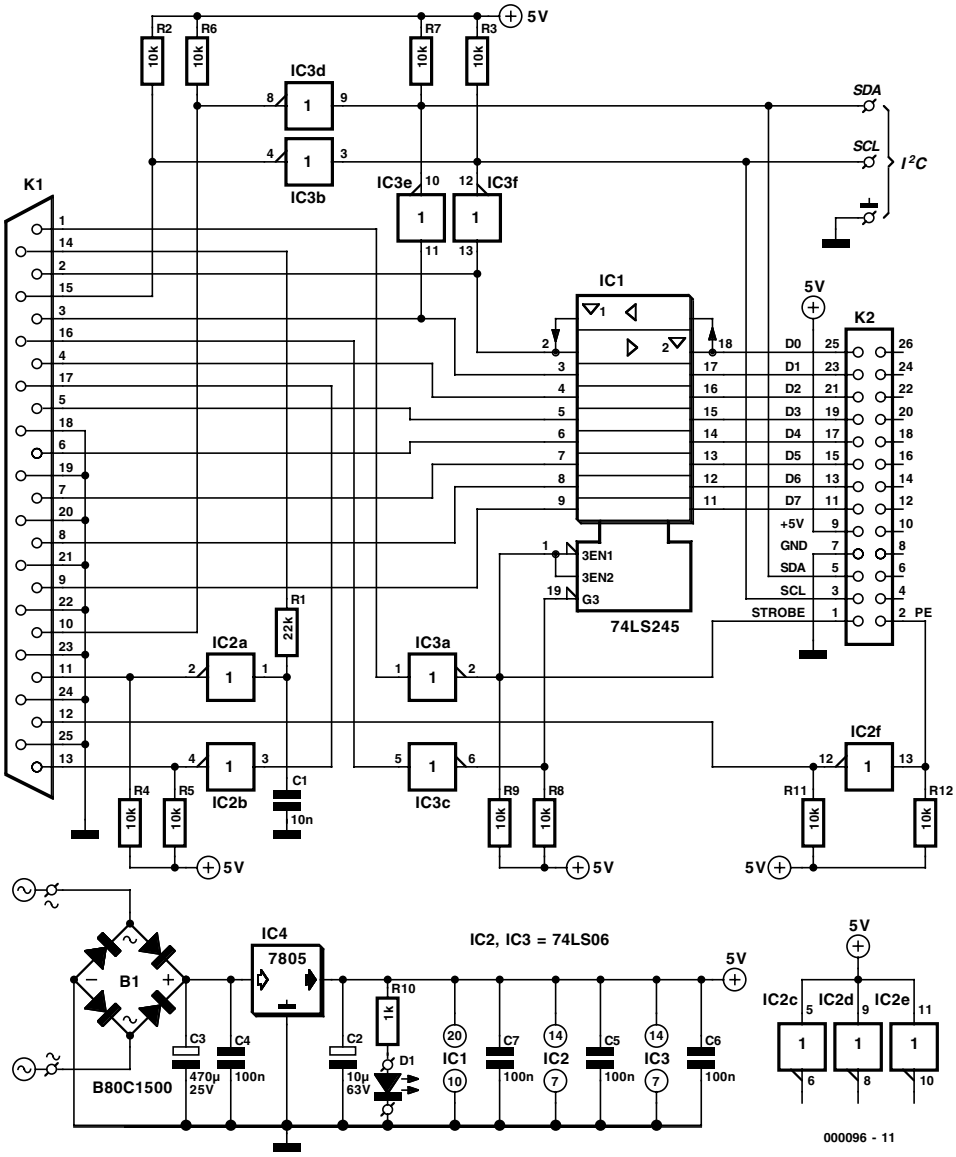


Figure 1. The hardware of the interface consists of a few gates and an 8-bit bi-directional bus driver.

(nInit), which determines whether the interface circuit works in the I²C mode or the parallel mode (DLL *io enable* and *io disable* functions). The data transfer direction (read or write) is selected via nWrite and gate IC3a (Low = read, High = write). Finally, the RC network R1/C1 and gate IC2a provide the required data input acceptance pulse on pin 11 of K1 (nWait). The correct timing for a read sequence is shown in **Figure 2**.

All these components, along with a simple voltage regulator formed by B1, C2-C7 and a 5-V regulator (IC4), plus a function lamp (D1), are mounted on the printed circuit board shown in **Figure 3**. The ICs must be mounted in suitable sockets, since some of the eleven wire bridges run underneath these components. Most of the 10-kΩ pull-up resistors are mounted vertically, and their free ends are attached in common to a single wire that connects to +5 V at the end of the row of resistors. A small 9-V transformer, or better yet a simple 9-V mains adapter, provides an adequate source of power.

DLL and BAS

All functions are located in the file *I2CDLL.DLL*, which is available along with the rest of the software on a diskette with order code **000096-11**. Since these are based on the C functions *outp* and *inp*, which directly access the hardware, they work only under Windows 95/98. They cannot be used under Windows NT, since NT prohibits direct hardware access. The file *I2CEPP.BAS* provides a sort of interface between the DLL and an application program, such as Visual BASIC. The functions and calling parameters are declared and described in this file.

Test program

The diskette includes a simply structured 8-channel logic analyser, which can be used without any additional hardware for testing the interface and displaying eight TTL-level signals on the screen. A maximum of 64,000

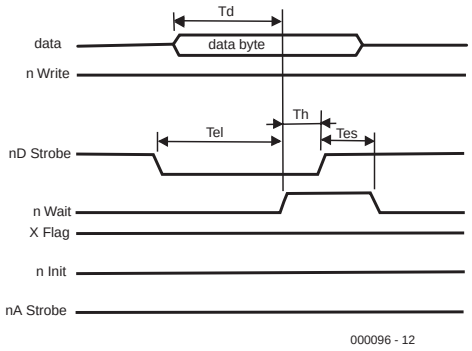


Figure 2. A read sequence using the EPP interface.

measurements can be made (controlled via the Counts slider), and the region to be displayed can be selected using the StartShow and EndShow controls. The measured values can be stored and read back in. The Strobe signal (pin 1 of K2) is pulled low during a measurement.

It is also possible to initiate measurements externally. The check box PE Trigger must be set for this. In this case, the measurement starts only after pin 2 of K2 goes Low.

(000096-1)

Literature:
Parallel Port Complete, Jan Axelson; Lakeview Research, ISBN 0-9650819-1-5

COMPONENTS LIST

Resistors:
R1 = 22kΩ
R2-R9,R11,R12 = 10kΩ
R10 = 1kΩ

Capacitors:
C1 = 10nF
C2 = 10µF 63V radial
C3 = 470F 25V radial
C4-C7 = 100nF ceramic

Semiconductors:
D1 = LED, high efficiency
IC1 = 74LS245
IC2,IC3 = 74LS06
IC4 = 7805

Miscellaneous:
K1 = 25-way sub-D plug (male), PCB mount
K2 = 26-way boxheader or pinheader
B1 = B80C1500 in round case (80V piv, 1.5A peak)
PCB, order code **000096-1**
Project software on disk, order code **000096-11**

On disk set #000096-1/4

SETUP	LST	8.474	04-03-00
ST4UNST	EX_	33.149	04-03-00
SETUP	EXE	59.392	04-03-00
SETUP132	EX_	68.950	04-03-00
STKIT432	DL_	12.439	04-03-00
VB40032	DL_	471.576	04-03-00
VEN2232	OL_	23.683	04-03-00
OLEPRO32	DL_	89.431	04-03-00
MSVCRT20	DL_	154.560	04-03-00
MSVCRT40	DL_	23.256	04-03-00
CTL3D32	DL_	15.385	04-03-00
VB4DE32	DL_	11.179	04-03-00
COMDLG32	OC_	74.496	04-03-00
MFC40	DL1	408.187	04-03-00
MFC40	DL2	134.144	04-03-00
MFC40LOC	DL_	14.265	04-03-00
TABCTL32	OC_	128.754	04-03-00
RICHTX32	OC_	117.028	04-03-00
COMCTL32	OC_	318.415	04-03-00
CCTLDE32	DL_	12.091	04-03-00
DBLIST32	OC_	117.805	04-03-00
DBGRID32	OC_	185.161	04-03-00
GRDKRN32	DL_	124.298	04-03-00
MSRDC32	OC_	68.918	04-03-00
MSRDO32	DL_	147.902	04-03-00
MSRDDE32	DL_	5.906	04-03-00
MSOUTL32	OC_	43.502	04-03-00
LA	EX_	15.469	04-03-00
CD4017	L_	8.567	04-03-00
DREHGEB1	L_	9.367	04-03-00
DREHGEB2	L1	3.069	04-03-00
DREHGEB2	L2	6.144	04-03-00
I2C_DLL	DL_	12.243	04-03-00
GRAPH01	IC_	505	04-03-00
COPYRI~1	TXT	188	17-08-00
CONTEN~1	TXT	822	17-08-00
CONTEN~2	TXT	1.035	17-08-00
CONTEN~3	TXT	411	17-08-00
CONTEN~1	TXT	0	17-08-00
COPYRI~1	TXT	182	17-08-00
cd4017	la	64.002	19-02-00
diagramm	Bas	11.170	18-02-00
drehgeb1	la	64.002	21-02-00
drehgeb2	la	64.002	28-02-00
epp_la	bas	2.901	22-02-00
FILEDI~1	BAS	11.566	19-02-00
GRAPH01	ICO	1.078	15-08-95
I2CEPP	bas	6.023	04-03-00
I2C_DLL	dll	22.528	21-02-00
la	exe	36.352	04-03-00
la	LA	64.002	19-02-00
la	vbp	1.378	26-06-00
la	vbw	116	26-06-00
la_main	frm	19.205	26-06-00
la_main	frx	1.090	26-06-00
MSSCCPRJ	SCC	189	26-06-00

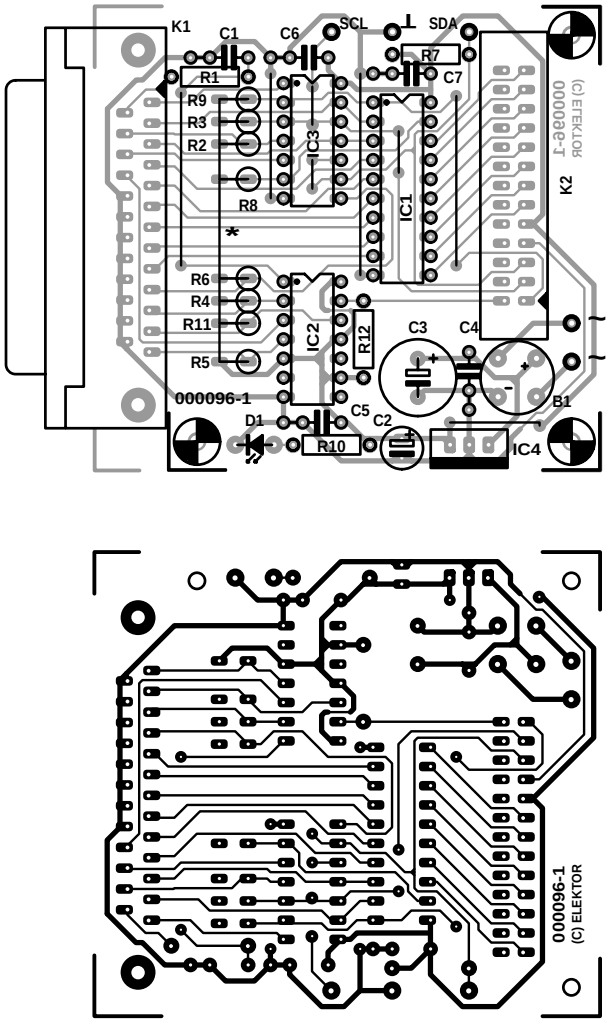


Figure 3. The hardware is mounted on this small printed circuit board.

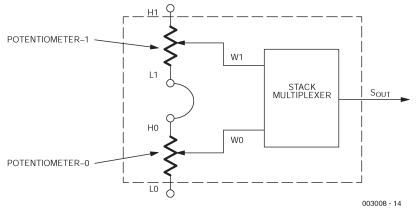


Figure 3. Stacked Configuration.

the potentiometer-1 wiper position value. Bit 1 will contain the MSB of the wiper set-ting for potentiometer-1 and bit 8 the LSB for the wiper setting. Bits 9 through 16 of the I/O shift register contain the value of the potentiometer-0 wiper position with the MSB for the wiper position occupying bit 9 and the LSB bit 16.

Transmission of data always begins with the stack select bit followed by the potentiometer-1 wiper position value and lastly the potentiometer–0 wiper position value.

When wiper position data is to be written to the DS1267, 17 bits (or some integer multiple) of data should always be transmitted. Transactions which do not send a complete 17-bits (or multiple) will leave the register incomplete and possibly an error in the desired wiper positions.

After a communication transaction has been completed, the $\overline{RST}$ signal input should be taken to a low state to prevent any inadvertent changes to the device shift register.

Once RST has reached a low state, the contents of the I/O shift register are loaded into the respective multiplexers for setting wiper position. A new wiper position will only engage after a RST transition to the inactive state.

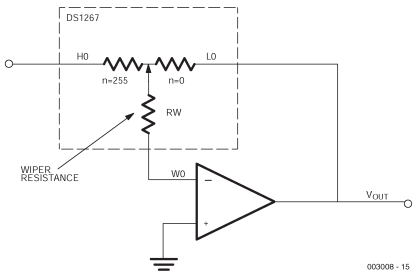
On device power-up the DS1267 wiper positions will be set at 50% of the total resistance or binary value 1000 0000.

Stacked Configuration

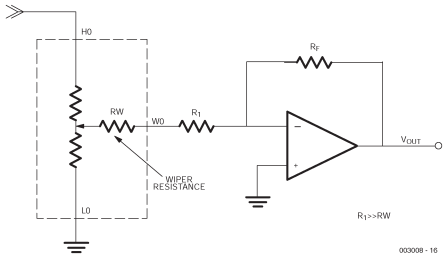
The potentiometers of the DS1267 can be connected in series as shown in Figure 3. This is referred to as the stacked configuration. The stacked configuration allows the user to double the total end-to-end resistance of the part and the number of steps to 512 (or 9 bits of resolution).

The wiper output for the combined stacked potentiometer will be taken at the S_{OUT} pin, which is the multiplexed output of the wiper of potentiometer-0 (W0) or potentiometer-1 (W1). The potentiometer wiper selected at the

S_{OUT} output is governed by the setting of the stack select bit (bit 0) of the 17-bit I/O shift register. If the stack select bit has value 0, the multiplexed output, S_{OUT}, will be that of the potentiometer-0 wiper. If the stack select bit has value 1, the multiplexed output, S_{OUT}, will be that of the potentiometer-1 wiper.



Typical application configuration: inverting variable gain amplifier.



Typical application configuration: fixed gain attenuator.

DS1267
Integrated Circuits
Special Function
ELEKTOR ELECTRONICS
DATASHEET 10/2000

DS1267
Dual Digital Potentiometer Chip

Manufacturer

Dallas Semiconductor.
Internet: http://www.dalsemi.com

Application Example

Gameboy Digital Sampling Oscilloscope (GBDSO), *Elektor Electronics* October & November 2000

Features

- Ultra low power consumption, quiet, pumpless design
- Two digitally controlled, 256-position potentiometers
- Serial port provides means for setting and reading both potentiometers
- Resistors can be connected in series to provide increased total resistance
- 14-pin DIP, 16-pin SOIC, 20-pin TSSOP packages
- Resistive elements are temperature compensated to ±0.3 LSB relative linearity
- Standard resistance values:
  - DS1267-10 ~10KΩ
  - DS1267-50 ~50KΩ
  - DS1267-100 ~100KΩ
- Operating Temperature Range −40°C to +85°C

Description

The DS1267 consists of two digitally controlled solid–state potentiometers. Each potentiometer is composed of 256 resistive sections. Between each resistive section and both ends of the potentiometer are tap points which are accessible to the wiper. The position of the wiper on the resistive array is set by an 8-bit value that controls which tap point is connected to the wiper output.

Communication and control of the device are accomplished via a 3-wire serial port interface. This interface allows the device wiper position to be read or written.

Both potentiometers can be connected in series (or stacked) for an increased total resistance with the same resolution. For multiple device single processor environments, the DS1267 can be cascaded or daisy chained. This feature provides for control of multiple devices over a single 3-wire bus.

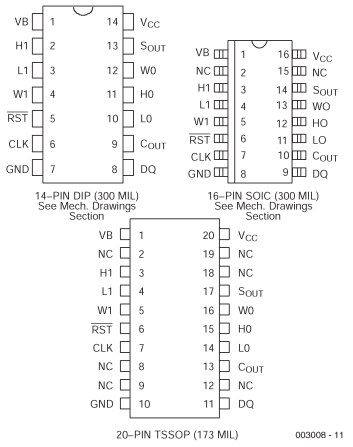
Pin description

L0, L1	Low End of Resistor
H0, H1	High End of Resistor
W0, W1	Wiper Terminal of Resistor
VB	Substrate Bias Voltage
SOUT	Stacked Configuration Output
RST	Serial Port Reset Input
DQ	Serial Port Data Input
CLK	Serial Port Clock Input
COUT	Cascade Port Output
VCC	+5 Volt Supply
GND	Ground
NC	No Internal Connection

The DS1267 is offered in three standard resistance values which include 10K, 50K, and 100K ohm versions.

Available packages for the device include a 14-pin DIP, 16-pin SOIC, and 20-pin TSSOP.

PIN ASSIGNMENT



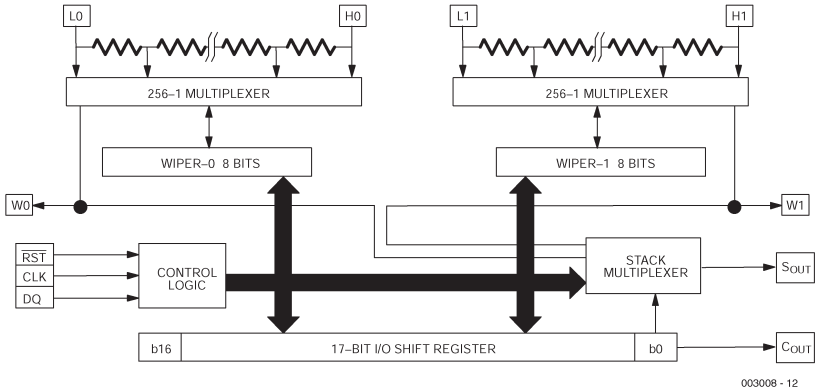


Figure 1. DS1267 Block Diagram

Operation

The DS1267 contains two 256-position potentiometers whose wiper positions are set by an 8-bit value. These two 8-bit values are written to a 17–bit I/O shift register which is used to store the two wiper positions and the stack select bit when the device is powered. A block diagram of the DS1267 is presented in Figure 1.

Communication and control of the DS1267 is accomplished through a 3-wire serial port interface that drives an internal control logic unit. The 3-wire serial interface consists of the three input signals: $\overline{RST}$, CLK, and DQ.

The RST control signal is used to enable the 3-wire serial port operation of the device. The Chip is selected when RST is high and RST must be high to begin any communication to the DS1267. The CLK signal input is used to provide timing synchronization for data input and output.

The DQ signal line is used to transmit potentiometer wiper settings and the stack select bit configuration to the 17-bit I/O shift register of the DS1267.

Communication with the DS1267 requires the transition of the $\overline{RST}$ input from a low state to a high state. Once the 3–wire port has been activated, data is entered into the part on the low to high transition of the CLK signal inputs.

Data written to the DS1267 over the 3-wire serial inter-face is stored in the 17-bit I/O shift register (see Figure 2). The 17-bit I/O shift register contains both 8–bit potentiometer wiper position values and the stack select bit. The composition of the I/O shift register is presented in Figure 2. Bit 0 of the I/O shift register contains the stack select bit. This bit will be discussed in the section entitled Stacked Configuration. Bits 1 through 8 of the I/O shift register contain

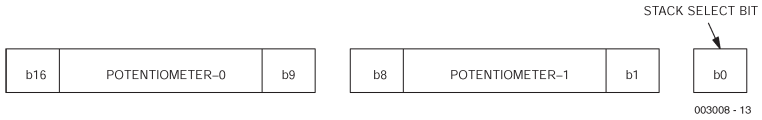


Figure 2. I/O Shift Register.

DS1267
Integrated Circuits
Special Function
ELEKTOR ELECTRONICS
DATASHEET 10/2000

Recommended DC Operating Conditions (–40°C to +85°C; V_{CC}=5.0V ±10%)						
PARAMETER	**SYMBOL**	**MIN**	**TYP**	**MAX**	**UNITS**	**NOTES**
Supply Voltage	V_{CC}	4.5		5.5	V	1
Input Logic 1	V_{IH}	2.0		V_{CC}+0.5	V	1
Input Logic 0	V_{IL}	–0.5		+0.8	V	1
Substrate Bias	V_B	–5.5		GND	V	1
Resistor Inputs	L, H, W	VB–0.5		V_{CC}+0.5	V	2
DC Electrical Characteristics (–40°C to +85°C; V_{CC}=5.0V ±10%)						
PARAMETER	**SYMBOL**	**MIN**	**TYP**	**MAX**	**UNITS**	**NOTES**
Supply Current	I_{CC}		22	650	µA	9
Input Leakage	I_{LI}	–1		+1	µA	
Wiper Resistance	R_W		400	1000	Ω	5
Wiper Current	I_W			1	mA	
Output Leakage	I_{LO}	–1		+1	µA	
Logic 1 Output @ 2.4 Volts	I_{OH}	–1			mA	7
Logic 0 Output @ 0.4 Volts	I_{OL}			4	mA	7
Standby Current	I_{STBY}		22		µA	5
Analog Resistor Characteristics (–40°C to +85°C; V_{CC}=5.0V ±10%)						
PARAMETER	**SYMBOL**	**MIN**	**TYP**	**MAX**	**UNITS**	**NOTES**
End-to-End Resistor Tolerance		–20		+20 %		
Absolute Linearity			±0.75		LSB	3
Relative Linearity			±0.3		LSB	4
–3 dB Cutoff Frequency	F_{CUTOFF}				Hz	6
Temperature Coefficient			±800		ppm/°C	

NOTES:

1. All voltages are referenced to ground.

2. Resistor inputs cannot exceed the substrate bias voltage, Vb, in the negative direction.

3. Absolute linearity is used to determine wiper voltage versus expected voltage as determined by wiper position. Device test limits ±1.6 LSB.

4. Relative linearity is used to determined the change in voltage between successive tap positions. Device test limits ±0.5 LSB.

5. Typical values are for TA = 25°C and nominal supply voltage.

6. –3 dB cutoff frequency characteristics for the DS1267 depend on potentiometer total resistance: DS1267-010; 1 MHz, DS1267-050; 200 KHz, DS1267-100; 100 KHz.

7. C_{OUT} is active regardless of the state of RST.

9. See Figure 11 in complete datasheet.