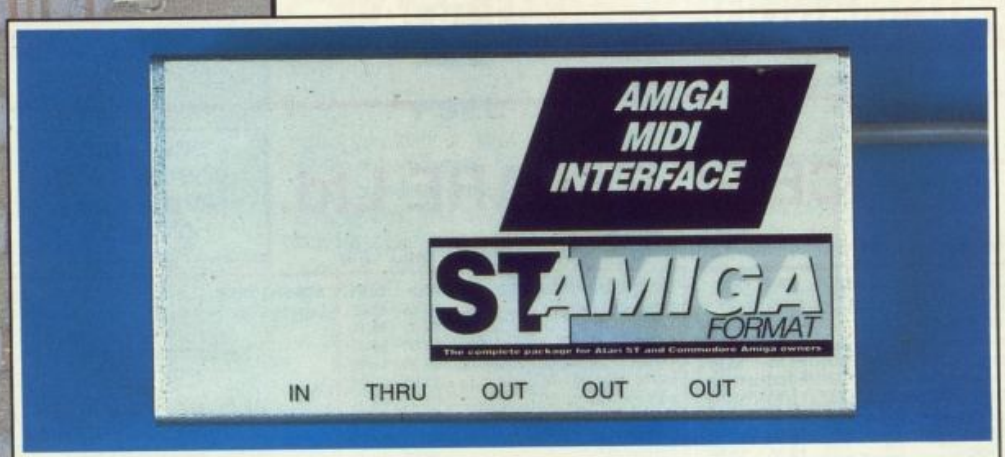
MIDI ON THE AMIGA

Judging from the replies to issue 1's questionnaire, a good number of you want to dabble in hardware projects. Never say we don't deliver. Robert Penfold explains how to build a price-busting MIDI interface for your Amiga and gives a few useful applications.

The range of MIDI software for the Amiga is not yet what could really be called vast, but there is a steady flow of new MIDI titles, and the Amiga has good potential in this area. Unlike the Atari ST, the Amiga has no built-in MIDI ports. The standard form of commercially available add-on MIDI interface for the Amiga is a simple adaptor for the serial port.

A MIDI interface for the Amiga does not need to be particularly complex, and it makes an ideal project for Amiga do-it-yourself addicts. The unit featured here plugs into the serial port of the A500, A1000, or A2000 and provides MIDI 'IN', 'THRU', and three 'OUT' ports.

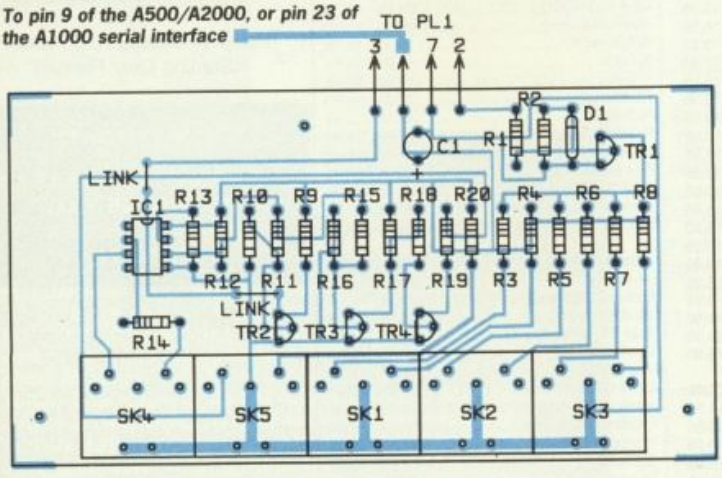


■ *The completed unit in all its glory*

Serial crops

Essentially, a MIDI interface is no more than a high-speed RS232 serial interface. However, there are differences between MIDI and standard RS232C interfaces, and these must be taken care of by software and the computer's hardware.

The standard MIDI word format is one start bit, eight data bits, one stop bit, and no parity. This is probably the most common RS232C word format, and it is one that any RS232C interface should be able to handle without any difficulty. The original MIDI specification had the baud rate at 19200 baud, which is the highest standard rate for RS232C interfaces. This was deemed too slow, and was raised to 31250 baud when MIDI was finally launched. This high and non-standard baud rate is the main obstacle when trying to use an ordinary serial port for MIDI purposes.



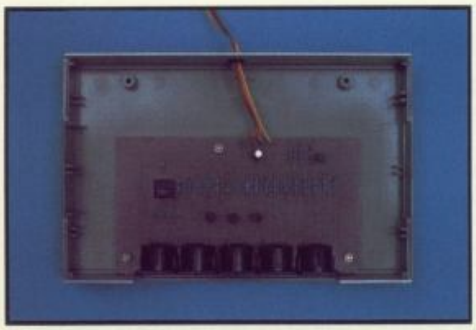
■ *Figure 1 – The component layout. In blue are the conductive copper tracks on the board. What you have to do is insert each component into its correct holes from the uncoppered side of the board, so that the wires stick out of the coppered side. Then solder the leads to the tracks. Refer to the parts list to make sure you put the right ones in the right places.*

This isn't printed at full size, so you can't etch your own board. A kit of parts is available – see later in the article.

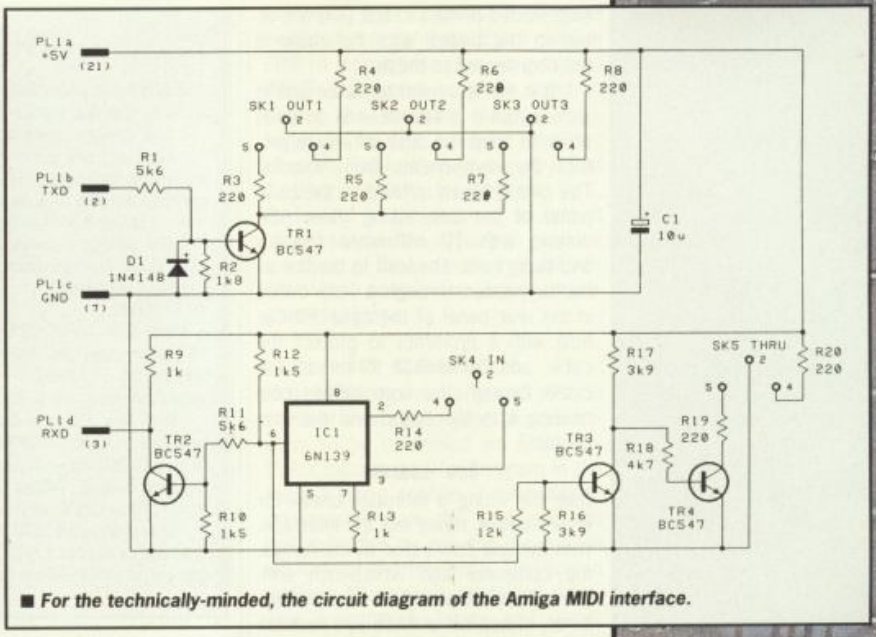
Construction details

Construction of the unit is made as easy as possible by having everything (including the five sockets) mounted on the printed circuit board. Details of the printed circuit and component positions are provided in Fig. 1.

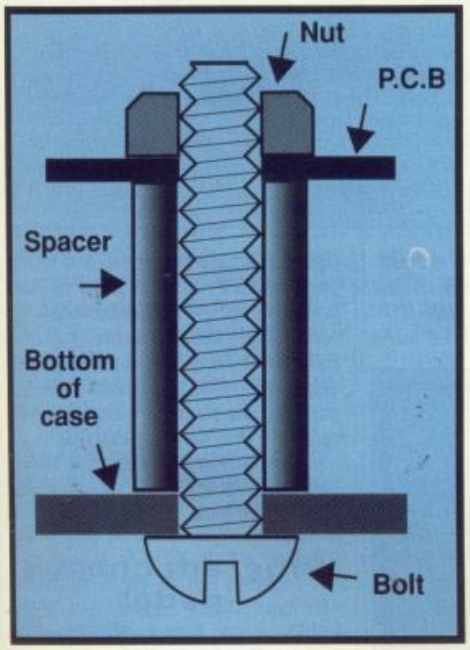
There are a few minor points to note when building the board, one of which is to make sure that IC1 is fitted round the right way. The U-shaped indentation on the body is not always present, but if not there should be a dot at the end of the component and



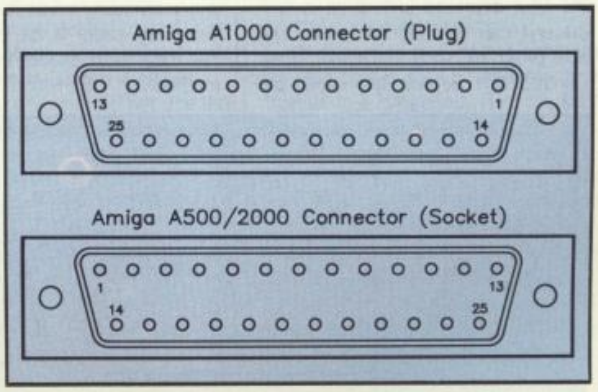
■ *A top view of the completed board.*



■ *For the technically-minded, the circuit diagram of the Amiga MIDI interface.*



■ *How to mount the board in the box using spacers. Drill out the three mounting holes marked on the board if you need to – don't try a hammer and nail or you'll split it!*



■ *Figure 2 – Connections to the serial interface 25-pin D connectors. These are viewed from the back, the side you see when soldering. The 12 volt power line is pin 9 on the A500/A2000, pin 23 on the A1000. The other pins used – 2, 3, and 7 – don't change.*

offset to the left (it indicates which pin is pin 1 of the device). You should solder the IC socket onto the board first of all (it doesn't matter which way round that goes), then slot in the chip itself the right way round.

You must also be careful to fit D1 and C1 round the right way. A '+' (and/or a '-' sign) on the body of C1 will show which lead is which, match it with the + on the board layout diagram. The four transistors have their leadout wires arranged so that they drop nicely into place on the board, but make sure there are no crossed-over wires. The resistors can be fitted either way round, but be careful not to get any of them swopped over. Two link-wires are needed, and these can be made from pieces of wire trimmed from resistor leadouts and bent to shape.

If you are new to electronic project construction, it's essential to practice soldering a few wires together before starting on the circuit board. Either that or buy two sets of components – one to practice with and one to build up properly!

You will need a soldering iron with a rating of about 15 to 25 Watts fitted with a bit of around 2 millimetres in diameter. The best solder for this type of work is a '22 s.w.g. 60% tin/40% lead multi-cored flux' type. Use plenty of solder when mounting the DIN sockets on the circuit board as they need to be well and truly fixed to it. Make sure all the components are fully pushed down onto the board before soldering them. It helps to splay the leadout wires of resitors and transistors slightly.

Good joints

Excess length on the component leadout wires should be trimmed about 3 millimetres proud of the board before they are soldered in place. When making the soldered joints place the iron on each joint first and then feed in the solder. A good joint will have a sort of mountain shape. A globular appearance usually indicates a connection of dubious quality. Where the connections are crowded together be careful to avoid bridging adjacent pads with too much solder. Carefully check the board for any accidental short circuits of this type and remove any that are found.

The board is connected to the Amiga via a lead about 1 metre long terminated in a 25 way D connector. A plug is needed for the A1000, but the serial port on the A500 and A2000 was changed to give IBM compatibility. A socket is therefore needed for these versions of the Amiga. The MIDI unit draws its power from the 12 volt pin of the RS232 port - this is pin 9 for the A500/A2000 but pin 23 for the A1000. Pins 2, 3 and 7 are the same for all Amigas, as shown in the wiring diagram for the different types of connector in Fig. 2.

Any 4-way cable should be suitable for the connecting lead. The multi-coloured "rainbow" type is best as this largely avoids the risk of misidentifying leads and getting crossed wires. Melt some solder onto the bared ends of the leads, and the appropriate four tags of the connector, before soldering the lead to the plug. There should then be no difficulty in making reliable joints. Fit the connector with a cover assembly. At the board end the lead can be soldered direct to the board, but it is easier if

DOESN'T WORK?

It isn't entirely beyond the realms of possibility that your Amiga MIDI interface won't work first time, particularly if you aren't an experienced soldering-iron wielder.

Magenta Electronics, who are supplying the kit of parts, are also offering a GET YOU GOING SERVICE. If you really can't make your unit work, send it back with a £4 cheque to cover service and post to: *MIDI Unit Repairs, Magenta Electronics, 135 Hunter Street, Burton-on-Trent, Staffs DE14 2ST.*

Make sure it is adequately padded from the post's infelicities, and please allow a couple of weeks for return.

single-sided printed circuit pins are fitted to the board, and the cable is then connected to these.

It is not essential to fit the unit in a case, but it is advisable to do so in order to keep the dust off and to protect the components from knocks. The circuit board is fixed to the base panel of the case using three 6BA screws with 10 millimetre spacers and fixing nuts. The lead to the D connector passes through a hole drilled in the rear panel of the case. Fit this hole with a grommet to protect the cable and remember to thread the cable through the hole before connecting it to the board and the connector.

In use

Give the wiring a thorough check for errors before trying out the interface. With the unit connected to the Amiga, the computer and Workbench software should boot-up and run normally. If not, switch off at once and recheck the MIDI interface unit. If all is well, the unit can be tried out with a suitable program, such as *Deluxe Music Construction Set*, or any other program which uses the serial port method of MIDI interfacing. The interface connects to the other MIDI equipment via standard (5 way DIN) MIDI Leads.

Although there are three outputs, they all transmit the same data. You do not have access to three separate and individually addressable outputs. This may seem a bit pointless, but this idea is to enable the MIDI 'star' method of connection to be used without having to buy a special THRU box (unless you want to drive more than three instruments from the unit that is).

If you like to write your own software it is not too difficult to access the MIDI interface, although it does not seem to be entirely straightforward from Amiga BASIC. Writing your own top flight sequencer or music notation programs is probably where do-it-yourself programmers can use their machine-code talents to the best effect.

When writing programs in a language that provides support for the serial interface it is probably better to take this route rather than directly controlling the serial port hardware. One point that you must bear in mind is that MIDI does not use handshaking, and so data may well be lost if the computer tries to halt the flow of input data by setting a handshake line to the hold-off state. Any processing of MIDI data must be done at high speed anyway, since the timing of data is normally important, and letting large amounts of data build up in a buffer will not give acceptable results. Although the MIDI baud rate of 31250 can be set using Workbench's preferences, note that this does not necessarily set the required baud rate when running a programming language.

Here follow three simple examples using AmigaBASIC for controlling MIDI instruments.

Program change pedal

Listing 1 is for a program that enables the Amiga to function as a program change pedal. The word 'program' is being used here in its MIDI sense of a set of sound data for a synthesiser, or a set of parameters for any MIDI controllable device. A program change pedal permits the sounds of the instruments in a system to be altered using the 'look no hands' approach. Although this is an invaluable feature for live performances, surprisingly few instruments incorporate it.

In the first part of the program the user enters the MIDI channel number (1 to 16) on which the program change messages must be transmitted. (Remember you must click the mouse on the input window before you can enter anything into an INPUT instruction.) Each program change message consists of two bytes, and the first of these is the one which carries the program change code (most significant nybble) and the channel number (least significant nybble). The program change code is 1100 binary, which is 192 in decimal. The channel number must be added to this to give the total value of the first byte. MIDI channels are conventionally numbered from 1 to 16, but the values used in MIDI messages are actually from 0 to 15. Adding 191 to the channel number entered by the user therefore

THE MIDI BAUD RATE

The MIDI unit plugs into the Amiga's serial interface, so how does the system know to use the higher-than-normal Baud rate required?

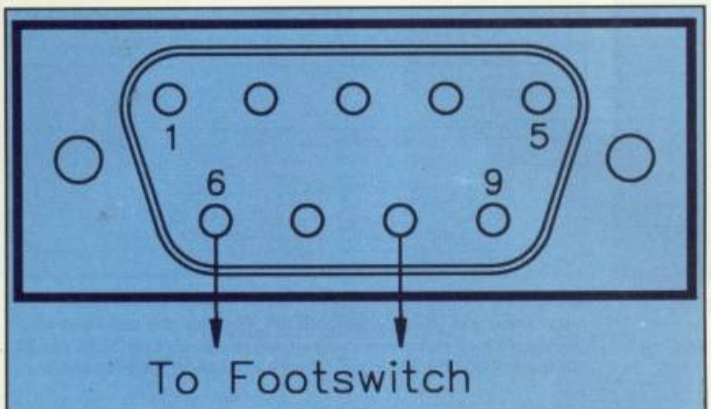
The Amiga controls its Baud rate by having a divide-by-'N' circuit between a clock generator and the device which provides the serial encoding/decoding (which is done by the the Amiga's 'Paula' custom chip). N is a number written to a hardware register, enabling any Baud rate within reason to be obtained with a fair degree of accuracy.

The Amiga's divide-by-N register is called 'SERPER', and is at address $DFF032. Not all computer languages, including AmigaBASIC, will accept six-digit hexadecimal numbers, so you need to use the decimal equivalent, 14676018.

Bits 0 to 14 of SERPER supply the number for the divide-by-N operation. To be precise, the divisor is one more than the number written to this register. Bit 15 of SERPER controls the byte length, which is either 8 or 9 bits. It would normally be 0, for eight bit operation.

With the aid of a calculator it is not too difficult to work out the correct value to write to SERPER. Each clock cycle lasts 0.2794μs, and the duration of each MIDI bit is 32μs. Dividing 32 by 0.2794 and deducting one gives the correct divisor, which is 113.53. Values of 113 or 114 give a Baud rate accurate to 0.5%, which is perfectly acceptable.

In the AmigaBASIC listings given elsewhere in the article, you will see the line `POKEW14676018&,114`. You should now be able to deduce why this sets the serial interface up for MIDI Baud rate.



■ ***Figure 3 – The pins you need to wire up if you are making a footswitch for the Pedal listing. This is shown from the back, the side you see when soldering.***

LISTING 1 – PROGRAM CHANGE

```
REM MIDI 'Program Change Pedal' Program
INPUT "Enter channel number (1 to 16) ", channel
change=channel+191
x=0
prog=0
OPEN "com1:300,n,8,1" AS #1
POKEW 14676018&,114
WHILE x=0
IF STRIG(2)=-1 THEN GOSUB midiout
WEND
midiout:
prog=prog+1
IF prog>63 THEN prog=0
PRINT# 1,CHR$(change);
PRINT# 1,CHR$(prog);
RETURN
```

LISTING 2 – MIDI CONTROLLER EDITOR

```
REM MIDI Controller Program
OPEN "coml:300,n,8,1" AS #1
POKEW 146760188,113
INPUT "Enter channel number ",chan
INPUT "Enter controller number ",conumb
header=chan+175
x=0
WHILE x=0
q=MOUSE(0)
z%=MOUSE(6)/1.92
PRINT PTAB(100);z%
PRINT# 1,CHR$(header);
PRINT# 1,CHR$(conumb);
PRINT# 1,CHR$(z%);
a$=INKEY$
IF a$=" " THEN GOSUB contnumber
WEND
contnumber:
PRINT " "
INPUT "Enter controller number ",conumb
RETURN
```

gives the correct value for the first byte in each message. The second byte in the message is simply the new program number, which is from 0 to 127. This is assumed to start at a value of 0, and the program increments it by one each time the pedal is operated.

The OPEN statement opens a channel to the serial port and sets it up with the right parameter. At least, it sets it up correctly apart from the baud rate, which is set to the right figure by the POKEW at the next line. The main body of the program is a three line WHILE...WEND loop. This checks the fire-button of joystick port 2, and branches the program MIDI-OUT subroutine when this button is operated. A joystick connected to port 2 is fine for testing purposes, but to use the system in earnest a foot pedal must be connected to port 2 via a 9 pin D socket. Connection details for the socket are shown in Fig. 3.

Some instruments do not use the MIDI program change numbers when selecting programs by way of the front panel controls. On a Casio CZ1 for example, programs are selected by two rows of push-buttons labelled A to H and 1 to 8. The instrument's manual should make it clear how these program identifiers relate to the true MIDI program number. If the maximum acceptable program number is reached, the program cycles back to zero again, and continues in this manner indefinitely.

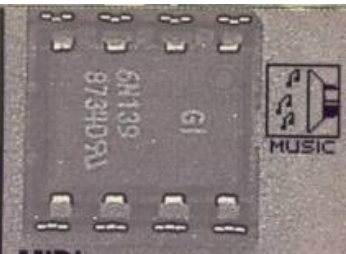
The program could easily be extended and customised to meet individual requirements. Extra pairs of PRINT# instructions plus an extended INPUT instruction could be used to permit the program change messages to be sent on more than one channel. Rather than having to adjust the instruments to set programs 0, 1, 2, 3, etc. to the required sounds, it should not be too difficult to produce a version of the program that enables the user to specify a sequence of program numbers.

In control

Most MIDI instruments permit at least some of their sound generator parameters to be controlled via MIDI controller messages. The program in the second listing enables a controller to be selected from the keyboard and then adjusted by moving the mouse backwards and forwards. The screen displays the current controller value. This is generally faster and easier than controlling instruments via their rows of push button switches.

In the initial part of the program the MIDI port is set up and the output channel to it is opened. Then the MIDI channel number and initial controller number are entered by the user. The MIDI control change message has 1011 in binary as the most significant four bits of the header byte (176 in decimal). Adding 175 to the channel number therefore gives the value of the entire header byte, which is stored in the variable HEADER. The main body of the program repeatedly outputs three byte MIDI controller messages. These consist of the header byte followed by the controller number, and then the value to be assigned to the controller. This value is obtained by reading the mouse using the MOUSE(6) function (which must be preceded by a dummy MOUSE(0) instruction). The value returned from the mouse is in the range 0 to 244, but it is converted to an integer in the desired ranger of 0-127 by some simple division, and rounded by assigning it to the integer variable Z%.

The final part of loop is used to



MUSIC

MIDI PROGRAMS ON DISK

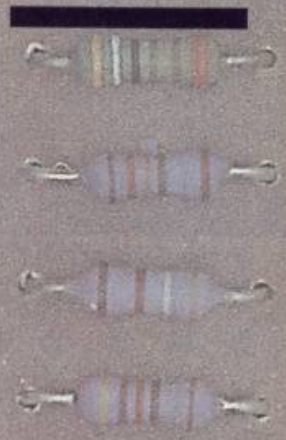
To get you going with MIDI we've included all the listings from these pages and two other superb utilities on this month's cover disk:

DX VOICE FILER – allows you to store sound voice data from any of the popular Yamaha DX series synthesizers.

MIDI UTILITY – allows you to record a MIDI sequence onto disk and play it back, and also acts as a voice filer for the Casio CZ101 synthesizer.

LISTINGS – the listings are in a drawer called MIDI LISTINGS on the disk. You need to load AmigaBASIC to be able to run them.

See the Disk Extra section on page 73 for more details.



■ *Thanks to the Bath Computer Shack for the loan of the Casio CZ101 keyboard used in the photographs.*

If you are looking for a entry-level MIDI synth, the Casio CZ101 or the Casio CZ230S are both recommended buys. They will cost around £250.

BAUD AND SIGNAL LEVELS

From the hardware point of view the main difference between MIDI and an ordinary RS232C interface is that MIDI uses a current loop system, whereas RS232C interfaces use different voltages to represent the two logic levels. The RS232C voltages are not normal 0 and 5 volt logic levels, but nominal potentials of plus and minus 12 volts. MIDI has a nominal output current of 5 milliamps which is switched on or off to represent the two logic levels. This is rather like switching a light bulb on and off, but it is actually a light emitting diode that is being controlled. This forms part of a device called an optoisolator which is standard issue at every MIDI output. The light output of the l.e.d is directed at a photo-cell, and this simple arrangement couples the signal through to the controlled equipment without having any direct electrical connection to it. This reduces the risk of problems with the "hum" loops that tend to plague electronic music systems, as well as eliminating the risk of electrical noise from the Amiga (or other micro-controller) being coupled into the audio circuits of the controlled instruments.

These differences in the input and output signals are all handled by the add-on interface.

monitor the keyboard, and it branches the program to the CONTNUMBER subroutine if the spacebar is pressed. This enables the controller number to be changed as and when desired.

It is perhaps worth pointing out that the MIDI continuous controllers have numbers from 0 to 63, but that these are used in pairs. For instance, controller 0 and controller 32 are paired, as are controllers 1 and 33, 2 and 34, etc. The lower numbered controller provides the seven most significant bits of the fourteen bit value, with the higher numbered one contributing the least significant bits. It is quite possible to roughly adjust a controller and then switch to its better half so that it can be fine tuned. In practice this is not usually necessary as few instruments use the full resolution, and the controllers from 32 to 63 are usually left unused.

Control numbers from 64 to 95 are switch types, and are either on (127) or off (0). For these controllers no other values are recognised by MIDI devices, but invalid numbers will not cause a malfunction. It is therefore quite in order to operate them by manipulating the mouse for a value of 0 or 127. Controller numbers above 95 are either unassigned, or used for things such as MIDI mode changing.

Due Process

A useful application for a computer is a MIDI processor. This is where the MIDI signal is taken in, doctored in some way, and then either fed back to the source or sent on to another device. A harmoniser is a good example of MIDI processing. This is where the MIDI signal from an instrument is altered so note values are all raised or lowered by a certain amount. If this signal is fed back to the IN socket of the source instrument, each note played on its keyboard produces a simple two-part harmony.

This function is provided by the program of listing 3. Although this is an Amiga BASIC program, it loads machine code which does all the processing. Amiga BASIC is really too slow for this type of thing, and there are difficulties in reading the serial port on a byte by byte basis anyway. Attempts to overload the machine code routine proved fruitless.

When asked to enter the note offset it is alright to use a negative number if the additional notes must be lower in pitch than the originals. A value of 5 or 12 will give a pleasant effect. Don't worry about the mouse 'dying' when the machine code program is running – it will come back to life when you exit the program. When using this program, remember that the MIDI output of the instrument must connect to the input of the MIDI interface, and the output of the interface must be connected back to the MIDI input of the instrument.

LISTING 3 – MIDI HARMONISER

```
REM MIDI Harmoniser Program
POKEW 146760188,114
INPUT "Enter offset in semitones ",offset
offset=offset+256
POKEW 146761228,8228
ba=2000008
FOR loop=1 TO 57
READ v(I)
POKEW ba,v(I)
ba=ba+2
NEXT loop
ba=2000008
POKEW 2000988,offset
PRINT "Press left mouse button and play
   note to exit"
CALL ba
POKEW 146761228, 409968
DATA &H267C,&H00BF,&HE001
DATA &H343C,&H0800,&H363C,&H0080,&H227C
DATA &H00DF,&HF030,&H247C,&H00DF,&HF09C
DATA &H207C,&H00DF,&HF018,&H3010,&H0800
DATA &H000E,&H6700,&HFFF8,&H3482,&H3200
DATA &H0241,&H00E0,&HB641,&H6700,&H0010
DATA &H0240,&H00FF,&H0640,&H0100,&H3280
DATA &H6000,&HFFDC,&H0240,&H00FF,&H0640
DATA &H0100,&H3280,&H3010,&H0800,&H000E
DATA &H6700,&HFFF8,&H3482,&H0240,&H00FF
DATA &H0640,&H0105,&H3280,&H1413,&H0802
DATA &H0006,&H6600,&HFFB2,&H4E75
```

ALL FOR UNDER £20!

To save you trouble, ST Amiga Format has arranged a special kit offer with Magenta Electronics:

Kit 1 – all components in this list except the case: £14.95 inclusive of VAT and p&p.

Kit 2 – as kit 1 but complete with a drilled case and tasteful ST Amiga Format labels, as in the photograph below: £19.95 inclusive. (You don't need the case for the kit to work, but it's obviously a good protection for the circuitry inside).

Send your orders direct to: ST Amiga Format MIDI Offer, Magenta Electronics, 135 Hunter Street, Burton-on-Trent, Staffs DE14 2ST, or phone 0283 65435 with your credit card details.

IMPORTANT: unless you specifically ask for an A1000 version you will get plugs suitable for the A500/A2000.

Please allow 21 days for delivery.

If you do want to shop around for yourself, one source of electronic components is Maplin Electronics in Essex. You'll find their catalogue for sale in most branches of WH Smiths.

PARTS LIST

RESISTORS (all 0.25 Watt, 5% or better)

R1, R11	5k6 (2 off)	green, blue, red, gold
R2	1k8	brown, grey, red, gold
R14	220Ω	red, red, brown, gold
R3 to R9, R13, R19, R20	1k (10 off)	brown, black, red, gold
R10, R12	1k5 (2 off)	brown, red, orange, gold
R15	12k	brown, red, orange, gold
R16, R17	3k9 (2 off)	orange, white, red, gold
R18	4k7	yellow, violet, red, gold

The gold band at the end of all the resistor colour codes means 5% tolerance. A red band means 2%, and brown 1% – either of these is quite satisfactory to use.

SEMICONDUCTORS

IC1	6N139	Opto-isolator (Integrated circuit)
TR1, TR2, TR3, TR4	BC547 (4 off)	Transistors
D1	1N4148	Diode

CAPACITOR

C1	10uF 25V radial electrolytic

MISCELLANEOUS

SK1 to SK5	5 way 180° DIN sockets, printed circuit mounting
PL1	25-way D connector (see test; varies with model of Amiga).
	Printed circuit board
	Case about 180x120x40mm
	8-pin d.i.l. integrated circuit holder
	6BA 12.5mm bolts, nuts and 6.5mm spacers (3 of each)
	1mm diameter printed circuit pins
	1 metre of 4-way cable
	1 suitable grommet
	solder