MIDI Analyser

When putting together complex MIDI systems and trying to get everything set up correctly it can sometimes be difficult to track down malfunctions. Most of these faults are not actually faults at all – it is just that something in the system is not set to the right operating mode, or sub-mode of an operating mode. With modern MIDI equipment and software there are usually a large number of options to choose from, and it is very easy to overlook something when setting-up a system. It can be quite time consuming (and frustrating) to track down these errors. Is it the transmitting device or the receiving one which is set to the wrong mode, or is there a genuine fault in the system?

This unit helps with the tracking down of problems in MIDI systems by indicating what type of MIDI message or messages a MIDI source is producing. With channel messages it shows the type of message (note on, pitch wheel, etc.) plus its MIDI channel number. For system messages it shows that the message type is indeed of the system variety, and exactly what kind of system message it is (start, continue, etc.). If the MIDI source is sending data on the wrong channel or something of this nature, this analyser should quickly identify the problem. The message type and channel number are indicated on a twenty-four LED display.

System Operation

The block diagram of Figure 3.27 shows the general arrangement used in this project. It is based on a UART which has a clock oscillator circuit to set the correct baud rate, and an opto-isolator circuit to convert incoming signals into a form that the UART can read. In this application only the receiver section of the UART is utilized.

The UART gives a series of eight bit codes on its parallel output, and the rest of the circuit must filter out the header bytes from the data bytes, and provide information on the header byte. This processing is done by two decoder circuits. The first of these detects message bytes, and indicates the type of message by setting one of eight outputs high. It is easy for the unit to differentiate between header bytes and data types as the most significant bit is always high on the former and

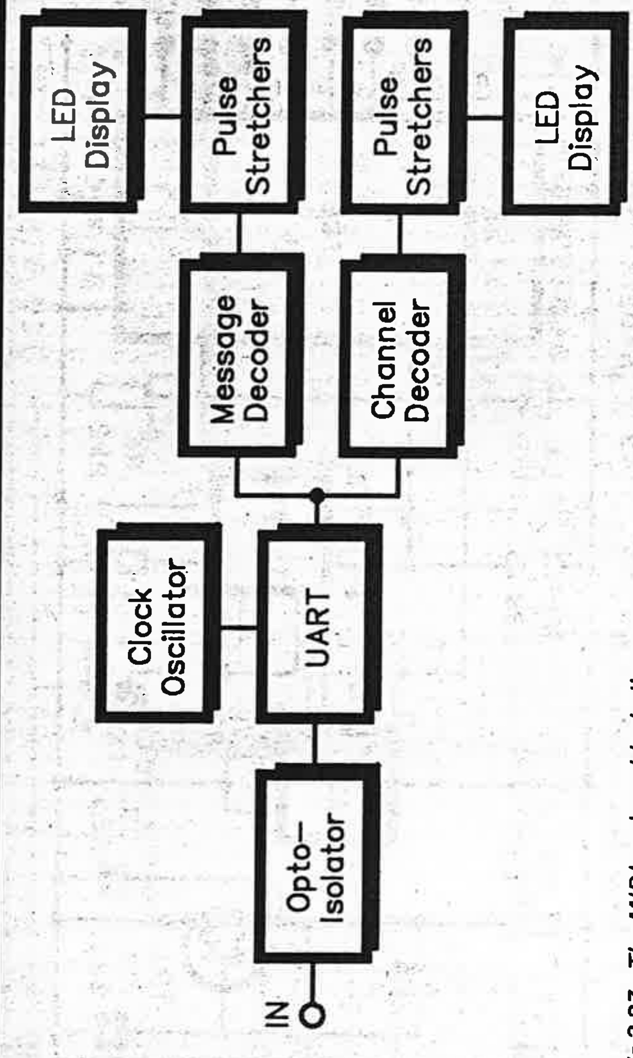

Fig.3.27 The MIDI analyser block diagram

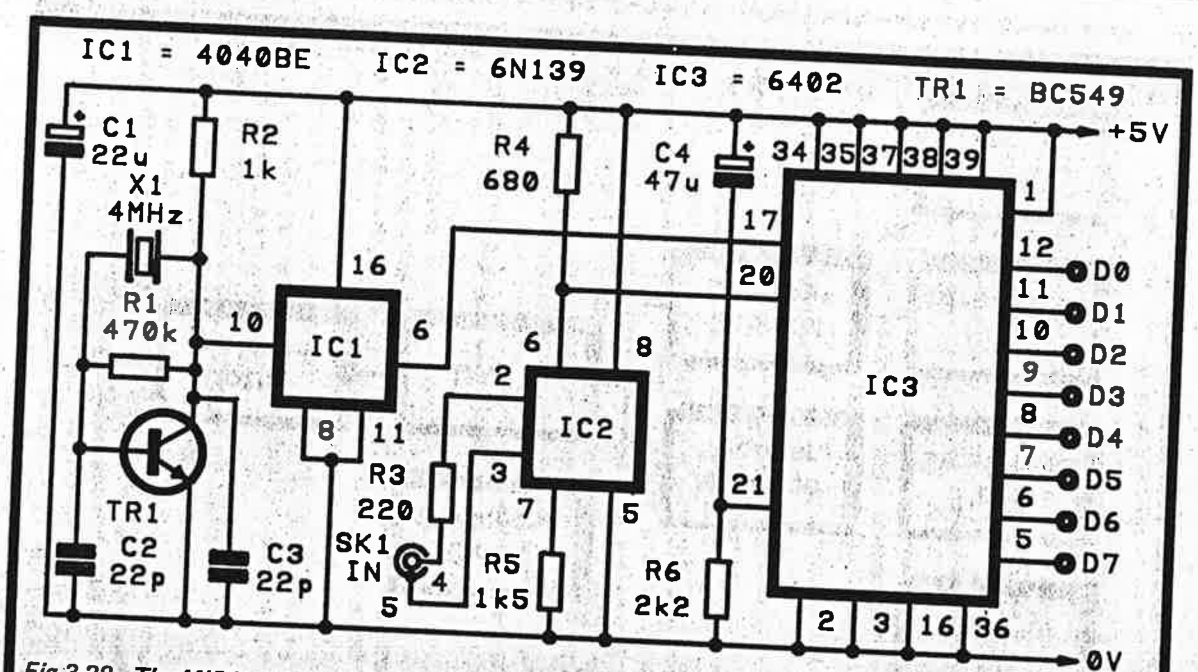

Fig.3.28 The MIDI analyser input circuit

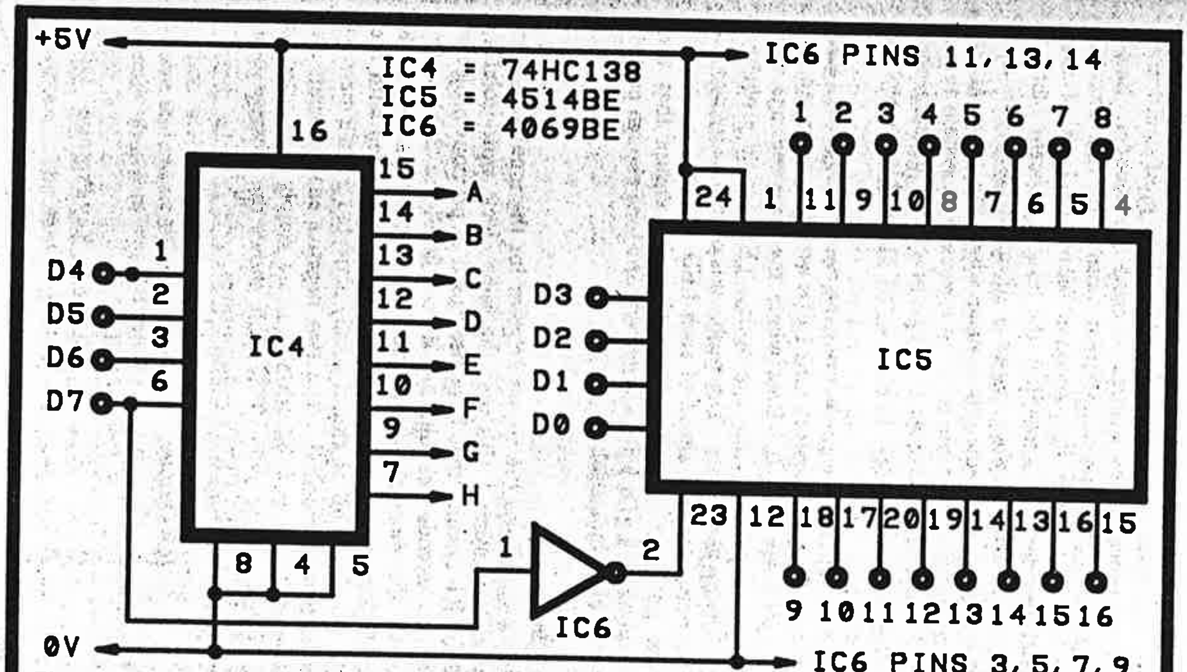

Fig.3.29 The two decoder circuits

low on the latter. The other three bits of the most significant nibble are processed by a three to eight line decoder. Each message type activates a different output of this circuit.

A four to sixteen line decoder forms the basis of the channel decoder. This monitors the least significant nibble, and each channel number causes a different output of the device to be activated. Of course, when the other MIDI decoder indicates that the received message is a system type, then this decoder indicates the type of system message and not the channel number (which is not applicable to system messages).

Directly driving the LEDs from the decoder circuits is not likely to be very successful as the LEDs would be activated for only very brief periods. Even where the same message was being sent repeatedly, the appropriate LEDs would probably only light up very dimly. This problem is overcome by using a simple pulse stretcher ahead of each LED. Even if a message occurs just once, the pulse stretchers will ensure that the corresponding LEDs will be activated for a long enough period to give a clear indication.

The Circuit

Figure 3.28 shows the circuit diagram for the input stages of the MIDI analyser. This uses the same clock, opto-isolator, and UART circuits that have been used in previous projects. The two decoder circuits are shown in Figure 3.29. IC4 decodes the most significant nibble, and this is a 74HC138 3 to 8 line decoder. Its positive enable input is fed with the most significant bit, so that it is only activated when a message header byte is received. The two negative enable inputs are not needed in this application, and are simply connected to the 0 volt supply rail. This table shows the message type indicated by each of IC4's eight outputs.

IC4 Output	*Message Type*
A	Note Off
B	Note On
C	Polyphonic Key Pressure
D	Control Change
E	Program Change

IC4 Output	*Message Type*
F	Channel Pressure
G	Pitch Wheel Change
H	System Message

The second decoder is based on IC5, which is a 4514BE 4 to 16 line decoder. This decodes the least significant nibble. It has an "inhibit" input at pin 23, and this is driven from the most significant bit via an inverter (IC6). Consequently, IC5 is deactivated when data bytes are present on the output of IC3. The numbers marked on the outputs of IC5 show the MIDI channel* number that activates each one. When the message is a system type, the outputs indicate the kind of system message that has been received, as detailed in this table. Note that some of the available codes are not yet assigned, and have therefore been omitted from this table.

IC5 Output	*Message Type*
1	System Exclusive
3	Song Position Pointer
4	Song Select
7	Tune Request
8	End Of System Exclusive
9	Clock
11	Start
12	Continue
13	Stop
15	Active Sensing
16	System Reset

There is an important difference between IC4 and IC5 in that IC4's outputs are normally high and go low when activated, whereas IC5's outputs are normally low and go high when activated. Consequently they require slightly different pulse stretcher/LED driver circuits. IC4 requires the stretcher circuit shown in Figure 3.30. There are four pulse stretchers here, with one based on each of IC7's gates. These are NAND gates, but in this circuit they are wired to operate as simple inverters. The diode, resistor, and capacitor at the input of each circuit, aided by the very high input impedance of

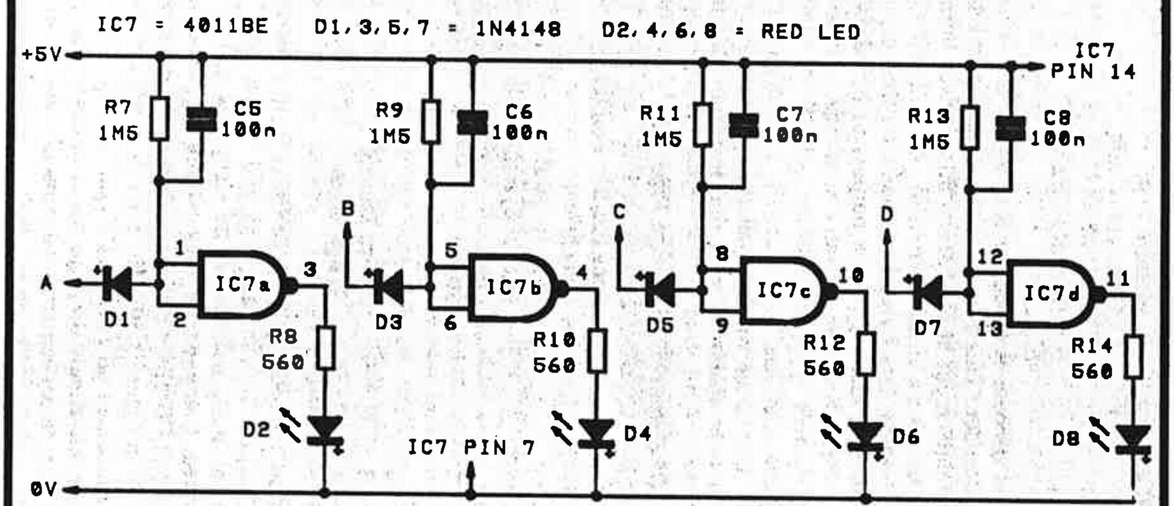

Fig.3.30 The pulse stretchers for IC4 (two sets required)

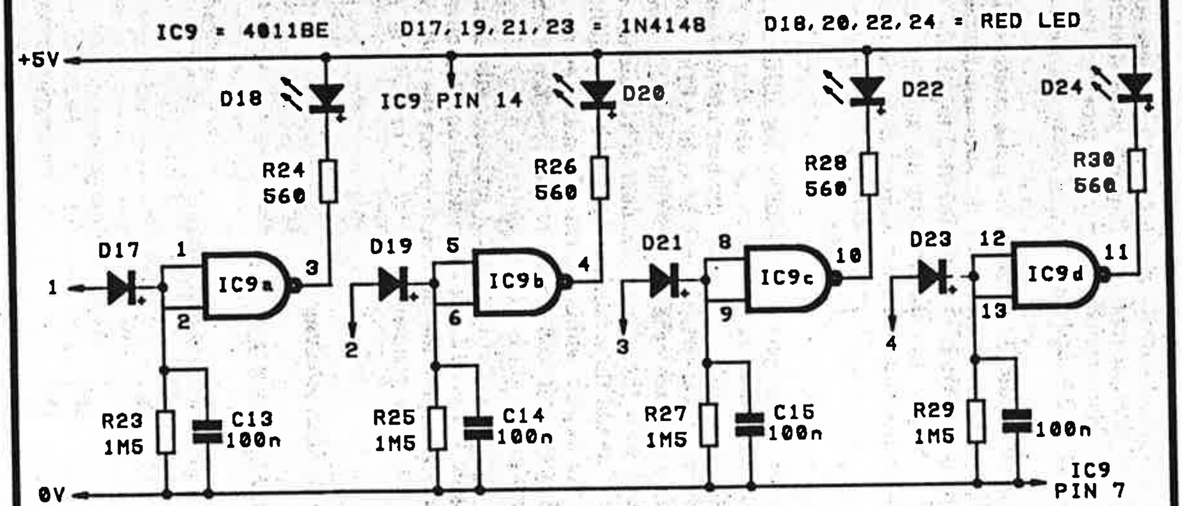

Fig.3.31 The pulse stretchers for IC5 (four required)

CMOS logic integrated circuits, give these circuits a fast attack and slow decay, thus providing the required pulse stretching. Note that only four stretcher/drivers are provided by the circuit of Figure 3.29, and that two of these are therefore needed.

The circuit of Figure 3.31 shows the pulse stretcher/LED drivers for IC5. These are essentially the same as the ones for IC4, but the configuration has been inverted to suit the outputs of IC5. As IC5 has sixteen outputs, and this circuit provides only four stretcher/drivers, four of these circuits are required.

Construction

Construction of this project should not be too difficult, but obviously the large number of LEDs in the display does complicate things slightly. Probably the best arrangement is to have them in three vertical rows of eight LEDs. Leave plenty of space between each row so that each LED can be clearly marked with the channel number it represents, etc. It is important to add these labels as it would be very difficult to interpret the display without them. Even if they are not very neat they should still prove to be very worthwhile.

The current consumption of this project is under 20 milliamps under standby conditions, but it increases substantially above this figure if several LEDs are switched on. The current consumption is still quite low enough to permit the unit to be powered in the same manner as the other projects in this chapter.

Components for MIDI Analyser

(Main Circuit, Figs 3.28 & 3.29)

Resistors (all 0.25 watt 5% carbon film)

R1	470k
R2	1k
R3	220
R4	680
R5	1k5
R6	2k2

Capacitors

C1	22µ 16V elect
C2	22p ceramic
C3	22p ceramic
C4	47µ 10V elect

Semiconductors

IC1	4040BE or 74HC4040
IC2	6N139
IC3	6402
IC4	74HC138
IC5	4514BE
IC6	4069BE
TR1	BC549

Miscellaneous

SK1	5 way (180 degree) DIN socket
X1	4 MHz crystal

Case, circuit board, wire etc.

(Display, Fig.3.30, also component values apply to Fig.3.31)

Resistors (all 0.25 watt 5% carbon film)

R7	1M5
R8	560
R9	1M5
R10	560
R11	1M5
R12	560
R13	1M5
R14	560

Capacitors

C5	100n polyester
C6	100n polyester
C7	100n polyester
C8	100n polyester

Semiconductors

IC7	4011BE

Semiconductors (continued)

D1	1N4148
D2	Red LED
D3	1N4148
D4	Red LED
D5	1N4148
D6	Red LED
D7	1N4148
D8	Red LED

Miscellaneous
LED panel holder (4 off)
14 pin d.i.l. holder
Wire, solder, etc.

Note that six sets of display components are needed in order to provide a full twenty-four LED display.

Finally

The projects featured in this book are all tried and tested designs, but they also provide a useful selection of basic building blocks which experienced readers can use as the basis of their own designs. Something that is well worth pursuing is MIDI processing. Apart from channelising, there are other possible applications for simple processors, including MIDI filters and harmonisers. There is plenty of scope for experimentation, and MIDI enables the imaginative user to do practically anything he or she wishes to.

Semiconductor pinout details are shown in Figure 3.32.

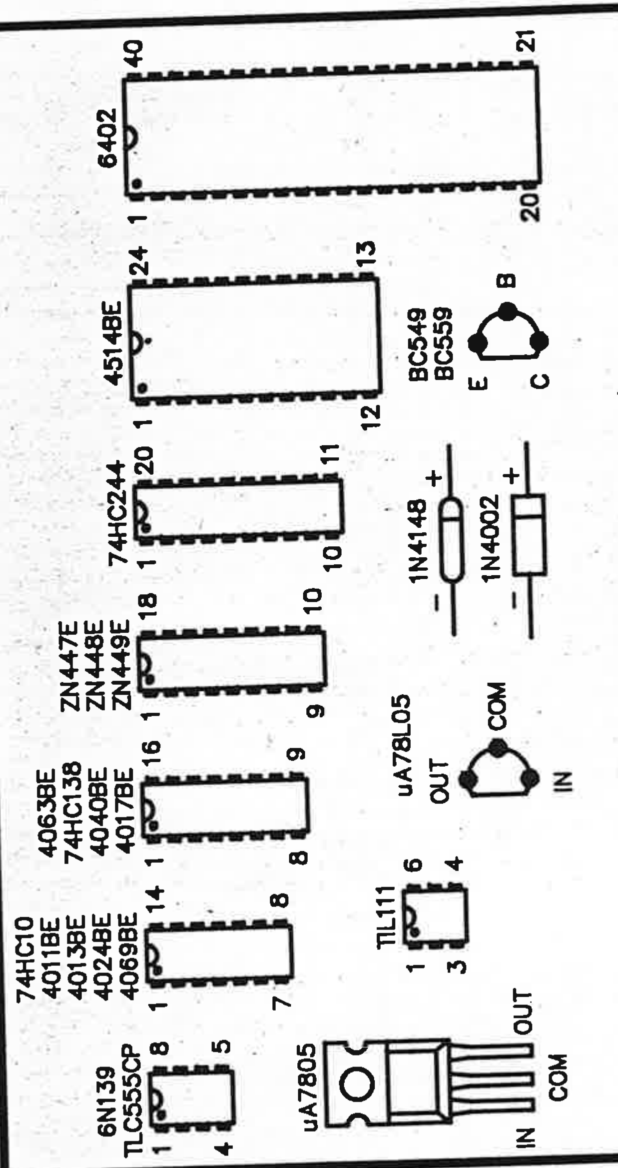

Fig.3.32 Semiconductor details (I.C. top views, transistor base views)