DISPLAY COUNTER

In this instalment of Workshop we add two seven-segment displays to our user port system that will enable us to display the contents of the user port data continuously in hexadecimal.

In order to display hexadecimal digits, four bits are required (four bits give us 16 permutations of zeros and ones). Thus, any eight-bit number can be represented using two hexadecimal digits: one for the lower four bits and another for the upper four. Although each display is made up of seven LED segments, the various combinations of segments can be 'driven' by four input lines if decoder logic is incorporated into the circuit.

Decoders are circuits that translate instructions from the computer to its peripherals into electrical signals, and vice versa. In our series on logic we built our own decoder circuit (see page 146), but for this exercise we can buy an off- the-shelf logic circuit. This is chip 7447 in the parts list.

The decoder for each display accepts four input lines from the user port and, via a sequence of logic gates, provides seven outputs. The logic circuit has been designed in such a way that if, say, the four input lines were 0111, then the appropriate bars would be lit to display the figure 7 (0111 in binary is equivalent to 7 in hexademical). The truth table for this is shown in the margin.

Hexadecimal digits greater than nine are usually represented by the first six letters of the alphabet — A to F. You will notice that the decoder chip that we are using has rather strange patterns to represent these digits. It is probable that these patterns can be generated using the logic circuits required for the digits zero to nine. More decoding logic would be needed to display the last six hex digits in the more usual alphabetic way, so by leaving the extra logic out and using different symbols for these digits, the number of logic gates in the decoder is reduced, thus bringing down the cost of manufacturing the chip.

Once the display circuit has been built we can display continuously the contents of the user port data register in hex using the eight data lines provided. There are sufficient lines available to drive the two displays simultaneously, but in many display applications this is not the case and several seven-segment displays have to share the same data lines. So that each display can show different information at the same time, a technique called 'multiplexing' is used. Essentially, the data lines from the display decoder are flipped from one display to the next, the data present on the lines also being changed appropriately. If this is done fast enough all the displays multiplexed in this way will appear to glow continuously, each displaying the data that was present at the time when it is momentarily connected to the data lines.

Parts List

No	Item	Maplin No
14	330 ohm 0.4 watt resistors	M330R
2	7447 BCD to 7-segment decoder	QX55K
1	Common anode double digit display	BY66W
2	16-pin DIL chip socket	BL19V
1	12-way minicon right-angle socket	YW30H
1	10-way minicon right-angle plug	YW19V
	8-way ribbon cable*	
	7-way ribbon cable*	
	Tinned bare wire*	
1	50 hole x 36 strip veroboard	FL09K
1	116 x 61 x 36mm plastic box	LH60Q

*You may have these parts left over from previous projects. The 12-way socket is necessary only if you want to extend the system bus. This, and several wire links on the circuit board may be omitted

We can demonstrate the principle of multiplexing using the two seven-segment displays that we are building. Because the display decoder represents decimal 15 by a blank, we can use this number to blank out one display while lighting the other. The following program, when run, asks for a digit to be displayed and then appears to show the digit on both displays simultaneously. However, a routine is included that inserts a delay to slow down the oscillation between the two displays. The delay is inserted while the Space bar is depressed. We can see, on running the program and depressing the Space bar, that the digit does in fact flip backwards and forwards between the two displays. When the Space bar is again released, the delay is removed and the flipping action is faster, making the digit seem to appear simultaneously on each display.

```
 10 REM BBC MULTIPLEXING
 20 DDR=&FE62:DATREG=&FE60
 30 ?DDR=255
 40 left_blank=15*16
 50 right_blank=15
 55 :
 60 REPEAT
 70 INPUT"DATA TO BE MULTIPLEXED";data
 80 ?DATREG=data+left_blank
 90 PROCslower
100 ?DATREG=data*16+right_blank
110 PROCslower
120 GOTO80
130 END
140 :
150 DEF PROCslower
155 REM IS SPACE BAR PRESSED ?
```