Constructional Project

PIC BIG DIGIT DISPLAY

JOHN BECKER

Microcontrolling giant 7-segment displays

RECENTLY Dave Fisher of Display Electronics told *EPE* that he had acquired several thousand electro-mechanical "big digits". These had previously graced the platforms of British Rail as 6-digit 7-segment clocks. Yes, they were the familiar "*click... click...*" digits that surely any would-be passenger has watched mesmerised while waiting for that (*where IS it?*) train to arrive.

In the course of conversation, the question of *EPE* designing a suitable electronic interface for these digits came up. Would Tech Ed be interested? *Certainly*, was the author's timely response to a novel design idea.

DISPLAY RESULTS

The resulting basic design is capable of driving from one to eight digits, with expansion up to 64 digits possible, as discussed later. They can be controlled via a standard 4 × 4 data entry keypad, or via a PC-compatible computer running under MS-DOS or Win95/98/ME.

A PIC16F84 microcontroller is the controlling device between the PC or keypad and the multiplexed digits. The PC software is written in QBasic/QuickBASIC but can be run as a standalone program without the need for QB to be installed.

The digits are ideal for use in any situation that requires a large electronically controlled display where the data is to be input intermittently. Applications that come to mind are sporting scoreboards, ticket draw results, display of outdoor temperature in public arenas – well, you've seen where large digits can be used, think up your own applications!

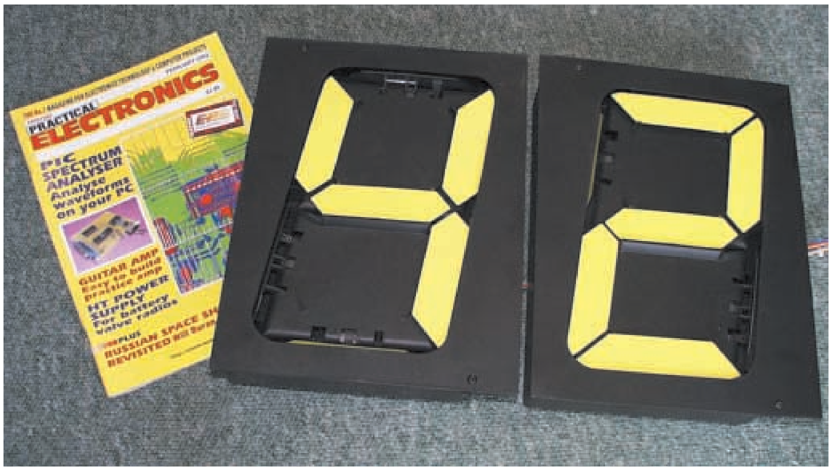
MONSTERS

Since the digits were only large versions of 7-segment displays, reasoned the author before starting the design, they could be simply driven by a PIC through a minimal bit of multiplexing. No problem – or so it seemed until two arrived!

The digits are monsters in several senses. Overall, they measure 12in high, 9in wide and 2·25in deep (30·5cm × 23cm × 5·5cm). The angled display area is effectively 10in high × 7in wide (25·5 × 18cm) and comprises seven bright-yellow hinged segments.

In the absence of fully informative data, the first task was to establish some criteria about controlling the display segments. Basically all that was known from a rudimentary data sheet was that a pulse of 12V d.c. for about 0·25secs was required to turn segments on and off, and that the pinouts of a built-in connector were shown. There was no mention of the current required, although there was a warning not to connect d.c. to the segments for long periods otherwise damage/heating will occur.

The original manufacturer's name was printed on the rear of the digits, Bodet,



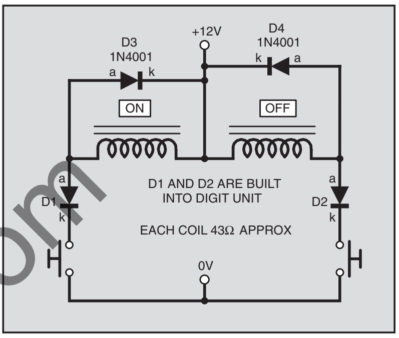


Fig.1. Basic circuit for controlling one segment.

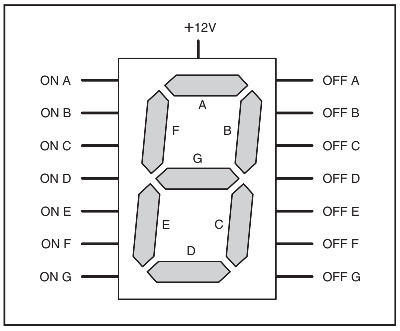


Fig.2. Each digit has 15 connections, 12V power input and two on-off controls for each segment.

along with the message Made in France. Doing a **www.google.com** search revealed the company at **www.bodet.com**, but no electronic specifications could be located, other than a schematic for one segment (see Fig.1 and Fig.2). An email to Bodet for data produced no response. Time for experiments!

Briefly connecting an ammeter between a segment and a 12V power supply revealed the current required to activate the mechanical flap – around 280mA. What?! Surely not? An ohms check across the various controlling coils showed a typical d.c. resistance of 43Ω. Wow, yes indeed, that unscientific test had shown a current figure in the right ball-park!

Furthermore, there were seven segments to be controlled – about 2A per digit, and users would probably need several digits. More used to dealing with liquid crystal displays needing only a handful of milliamps, rather than two thousand milliamps, the author recognised that the digits were more than just monsters in size.

RESEARCHING DESIGN

Having numerous data books and CD-ROMs is always to be recommended. These days, so is Internet access. Using a mixture of sources, a couple of evenings were spent researching the type of semiconductors that were available to handle such currents in a multiplexed situation. It was a foregone conclusion that they needed to be capable of being PIC-controlled.

Any idea of using any form of discrete transistor, power-FET or otherwise, was rejected. Such techniques were fine years ago, but hardly today's technology when multiplexing – even less so regarding any suggestion of relay control. No, it had to be semiconductors in integrated circuit form.

Anyone familiar with controlling 4-digit 7-segment light emitting diode displays will know that they can easily be controlled by multiplexed signals – a common 7-line "bus" feeding identically to all segments of all digits, and then separate power supply lines, each feeding to its own digit. The technique required then is to send out segment control data along the common bus, and to only turn on digit power lines individually at the appropriate moment.

However, data sheet browsing suggested that switching seven segments simultaneously at a total of 2A or so could present a significant problem. Perhaps switching segments individually at about 280mA would be more sensible?

There are many chips that can provide 1-of-8 output selection in response to a 3-bit control code. Such chips include the 74HC138, whose outputs are normally high, but go low individually when selected by the appropriate control code. The 74HC237 operates with the opposite output logic, normally low but going high when selected.

Although the outputs of these devices cannot handle the sinking or sourcing of 280mA, or a voltage of 12V, they are capable of driving intermediate high-current buffers. The question then became one of which buffers were available?

LINE DRIVER

After considerable research, it was decided to use the 7-stage line driver type ULN2004A to activate the segments. It can sink 500mA per stage, and is capable of handling voltages up to 50V.

This device also has the benefit of having built-in diodes across each output which inhibit back-e.m.f. generation when switching inductive loads, such as the segment coils (see Fig.3).

It is a bipolar-fabricated Darlington device that requires a positive voltage at each input to turn on the respective open-collector output. Conveniently, each input has its own 10·5kΩ series resistor, removing the need for external resistors (such as required in the control line feeding into the base of a "normal" discrete transistor). The inputs are also diode-protected.

Using a 74HC237 multiplexer, the seven segments can readily be controlled directly through the ULN2004A driver.

COMMON ANODE

That took care of current sinking through the segments. The problem then became that of providing multiplexed power to each digit. If you relate the concept to a common-anode 7-segment l.e.d. matrix, the individual segment cathodes had now been catered for – it was the common-anode current control that was now required. In other words, a current *source* was needed, at a minimum of 280mA.

It had been expected that as multiple high-current sinking paths were available in one i.c., as with the ULN2004A, so multiple high current sourcing devices would be equally common.

It was found that there were many options available if a source current of no more than 100mA were required, especially as the current would be pulsed intermittently. However, the requirement for at least 280mA presented a seemingly unsolvable problem, unless discrete transistors were used – which the author was determined not to resort to. Quite simply, no *ideal* i.c. devices could be found.

Briefly, power op.amps such as the L272 dual device seemed a possible solution, but that was not deemed "tidy"! Eventually, it was decided to accept a less-than-optimum option, to use an L293DN quadruple Half-H driver.

This has four devices that can each be set to sink or source a current of up to 1A at a voltage from 4·5V to 36V. It also has two enable inputs which allow pairs of drivers to have their outputs placed into a high-impedance state (see Fig.4). Additionally, it too has in-built diode protection.

The device is intended for reversible motor and solenoid control. The term Half-H refers to the bridge configuration in which the pairs of drivers can be operated. It seemed suitable for this application since no other appropriate device format could be found. Consequently, two L293DN devices are used in the main circuit, each providing power for four multiplexed digits. They are under combined control of another 74HC237 1-of-8 controller.

The L293DN, however, has the unfortunate side effect of consuming around 20mA even when the outputs are in a high impedance state. The "enable" inputs do not place the device into a quiescent state in high-impedance mode, unlike many logic devices that you may be familiar with. Regrettably, it is not cheap.

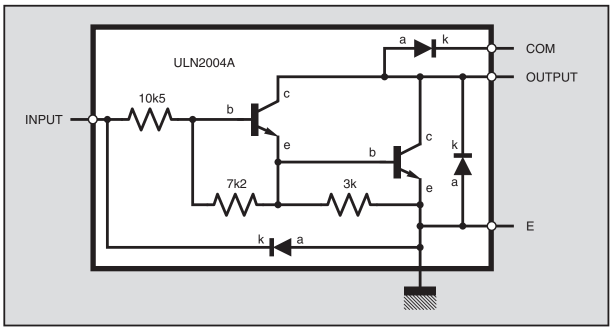


Fig.3. Schematic of one stage within a ULN 2004A 7-stage line driver.

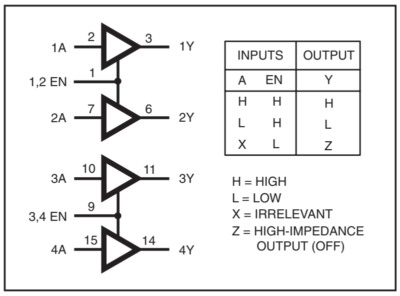


Fig.4. L293DN pinouts and logic table.

Do not use any other type of L293 device. The L293DN (note the DN suffix), is a 16-pin device with diode protection. Other L293 device types may not have the same characteristics (the L293E, for instance, has 20 pins and cannot be used).

MULTIPLEX CIRCUIT

A simplified block diagram of the control requirement is shown in Fig.5.

The circuit diagram showing the multiplexing and digit drive devices is given in Fig.6. Control data originates from a PIC16F84 microcontroller (discussed presently in relation to Fig.7). Through multiplexer IC1, 3-bit control data selects which digit is to be powered via source drivers IC2 or IC3.

As shown in Fig.6, and designed on the printed circuit board to be described later, eight digits (one "bank") can be controlled by these two drivers. Additional digit source drivers can be added separately if required (on stripboard for example, although no constructional details on this are offered).

If fewer that five digits are to be controlled, IC3 can be omitted.

The eight outputs of IC1 are common *to all* digit drivers, and IC1 does not need to be repeated if additional banks of drivers are added.

The software allows two additional banks of eight digits (a total of 24 digits) to be controlled without modification to the program. Readers who are familiar with PIC and QB programming could modify the software to cope with multiplexing up to 64 digits if an additional 74HC237 multiplexer is used (see later).

Segment selection is provided by a 3-bit code fed to multiplexers IC4 and IC5. These in turn control segment sink drivers IC6 and IC7, respectively. Only seven outputs of these multiplexers are used.

The two multiplexers are under "chip select" (CS) control by separate CS1 lines (pins 6), so that segment On or Off control is achieved not only in respect of the 3-bit code, but also in terms of current-sinking pulse duration (more later).

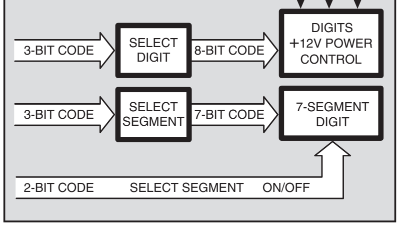


Fig.5. Multiplexed control logic.

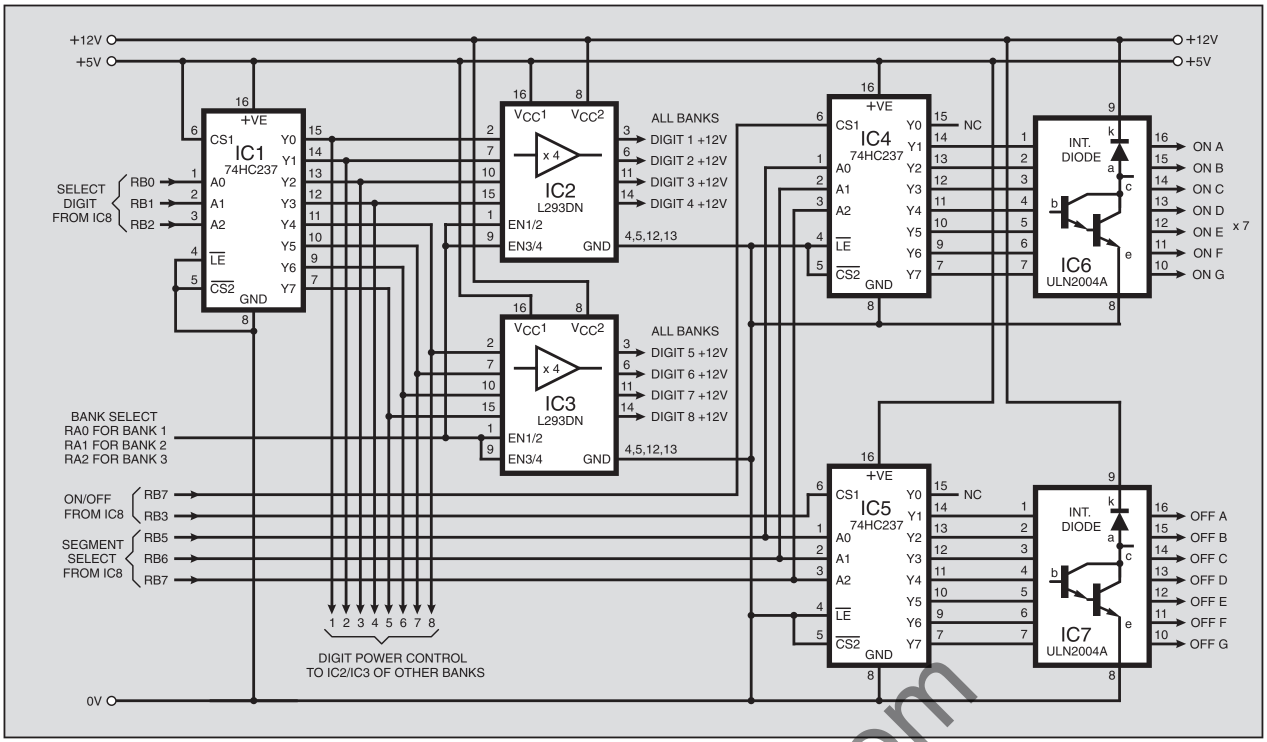


Fig.6. Circuit diagram for the multiplexed control of the digits, basically for eight, but can be modified to control 64 digits.

MICROCONTROLLER

Because of the multiplexing arrangement, a PIC16F84 microcontroller is readily suitable for this design, see Fig.7. It is capable of being user-controlled either via a 16-key (4 × 4) data entry keypad, or via a PC-compatible computer, running under MS-DOS or Win95/98/ME.

The PIC is run at 4MHz, as set by crystal X1. Port pins RB0 to RB2 control digit selection via IC1 (Fig.6), pins RB4 to RB6 control segment selection via IC4 and IC5, RB7 controls selection of IC4 (segment On control), and RB3 controls selection of IC5 (segment Off control).

Port pins RA0 to RA2 perform "bank" selection. As shown, they can control up to three banks of eight digits. If they are used to control another 74HC237 1-of-8 multiplexer, however, they could control eight banks (with suitable software modification).

Port B pins are also used for inputting data from a 16-key keypad, *or* from a PC. Note that it is unwise to connect a keypad and PC simultaneously since one might adversely affect the other. The PIC itself is protected against its Port B pins being undesirably affected by external PC/keypad control by the inclusion of buffer resistors R1 to R8.

Pins RA3 and RB7 are used by the software to achieve "handshaking" with the PC when the unit is under computer control.

Pin RA4 is used in a manner possibly not seen by readers before. It is used in oscillatory mode under software control and at a rate set by preset VR1 and capacitor C5. It allows the segment control pulse width to be varied. The controlling software routine will be discussed towards the end of this article.

As usual with the author's PIC designs, on-board programming can be performed via a 4-pin connection (TB1). Adverse effects on the +5V power line are prevented during programming control by the inclusion of resistor R9 and diode D1.

POWER SUPPLY

Power for the digits needs to be 12V d.c. This may be provided from any source capable of supplying at least 500mA (to provide "headroom" when a segment is activated). It does not need to be stabilised. A 12V car battery is suitable.

The prototype was found to operate with a supply voltage as low as 9V (with resultant reduction in current consumption).

Whilst the 13·5V (or so) of a fully charged battery seems acceptable, it would appear to be unwise to allow the supply to significantly exceed this voltage. The voltage, current and pulse duration limits for the digits are not known since Bodet did not respond to the author's request for information.

The digital control i.c.s require to be powered at +5V d.c. (which *must not* be exceeded). This is provided from the 12V line via regulator IC9, which can supply up to 100mA of sustained current. Be aware, though, that on the prototype it was

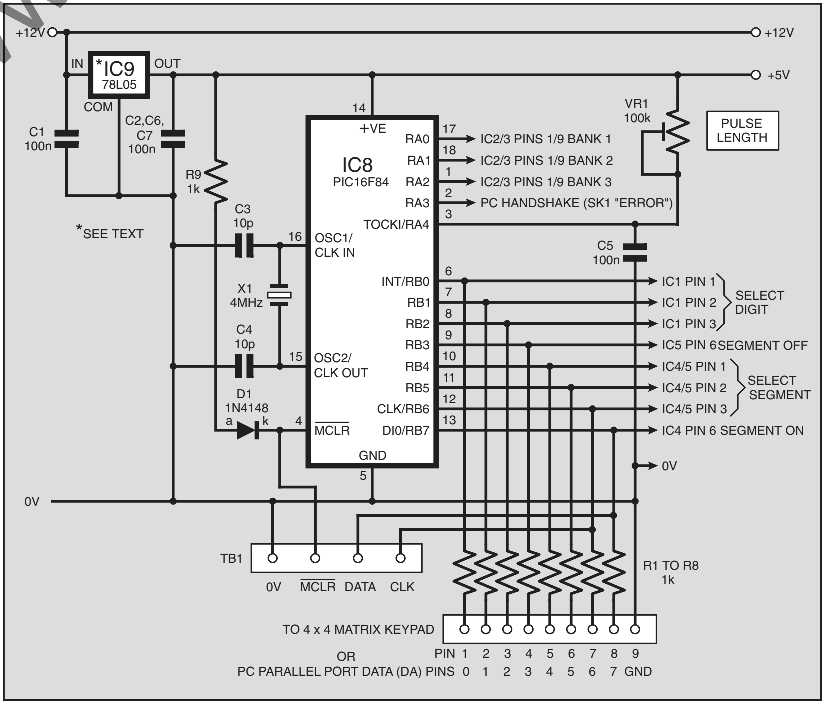


Fig.7. Circuit diagram showing the PIC16F84 control connections, plus power supply.

COMPONENTS

Resistors

R1 to R9 1k (9 off)
All 0·25W 5% carbon film

See SHOP TALK page

Potentiometer

VR1 100k min. preset, round

Capacitors

C1, C2, C5 to C7	100n ceramic, 0·2in pitch (5 off)
C3, C4	10p ceramic, 0·2in pitch (2 off)

Semiconductors

IC1, IC4, IC5	74HC237 1-to-8 multiplexer (see text) (3 off)
IC2, IC3	L293DN 16-pin Half-H driver (see text) (2 off)
IC6, IC7	ULN2004A 7-way Darlington line driver (see text) (2 off)
IC8	PIC16F84 microcontroller, preprogrammed (see text)
IC9	78L05 +5V 100mA (or 7805 +5V 1A) regulator (see text)

Miscellaneous

X1 4MHz crystal

Printed circuit board, available from the *EPE PCB Service*, code 341; RW44 10-inch 7-segment electromechanical display (big digit), quantity to suit (see text); 4 × 4 data entry keypad (optional – see text); stranded colour-coded connecting wire (individual wires or ribbon cable); 12V d.c. power source, min. 500mA output; 1mm terminal pins or pin headers; 16-pin d.i.l. socket (7 off, see text); 18-pin d.i.l. socket; printer port connectors to suit (optional – see text); solder, etc.

Approx. Cost Guidance Only **£20** *excluding hardware*

required to constantly provide around 40mA (due to the two L293DN devices). It is thus likely to get a bit warm, especially if the source power is 12V or greater.

If it is found to shut down through excessive heat (it is thermally regulated), change it to a standard 7805 +5V 1A device. It is perhaps prudent to switch off power during long periods of digit inactivity.

Note that the digits themselves only consume power during the brief pulse that changes their segment display position.

Capacitors C1, C2, C6 and C7 help to maintain powerline stability.

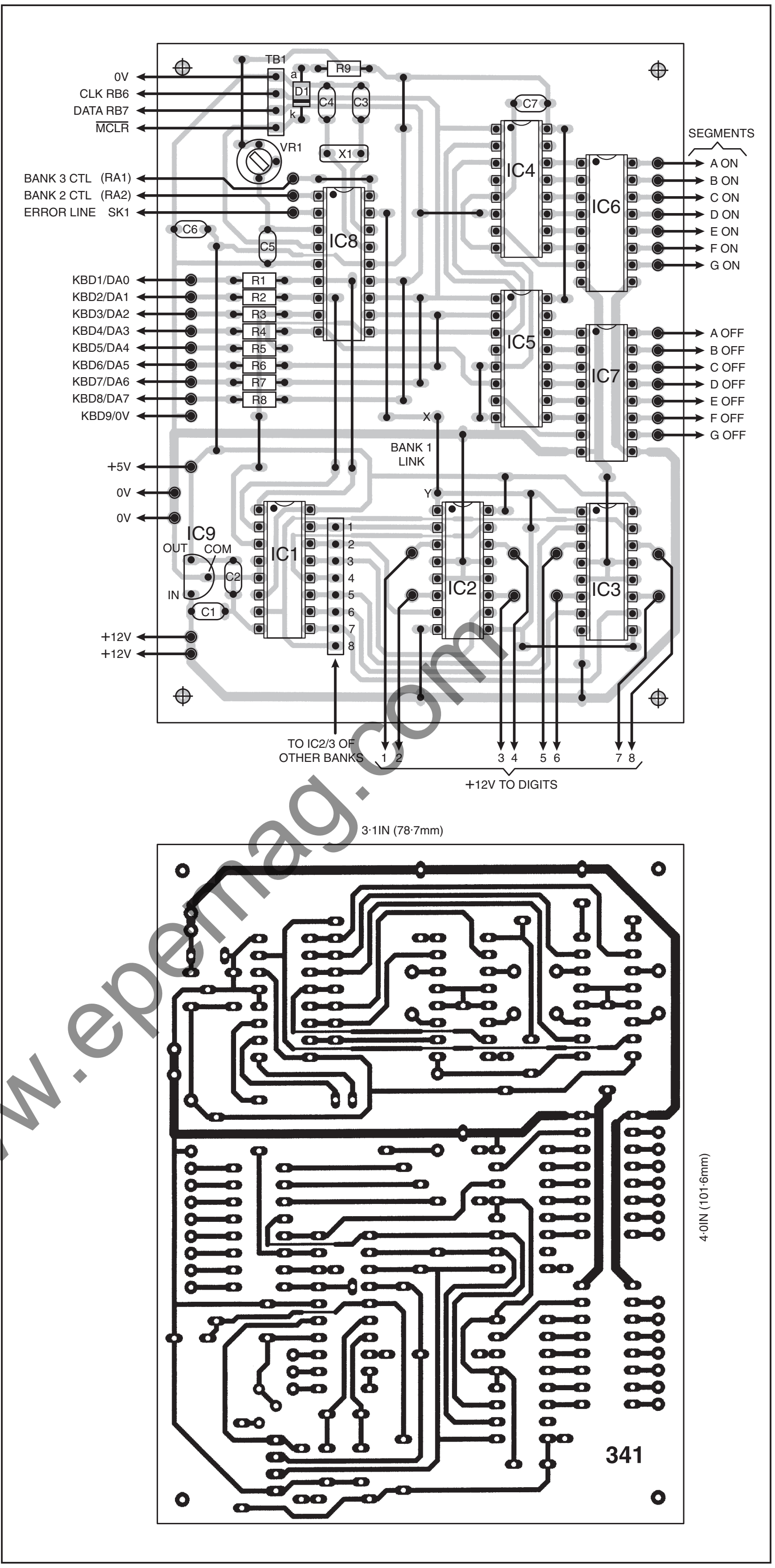


Fig.8. Component layout and full-size underside copper foil master track pattern.

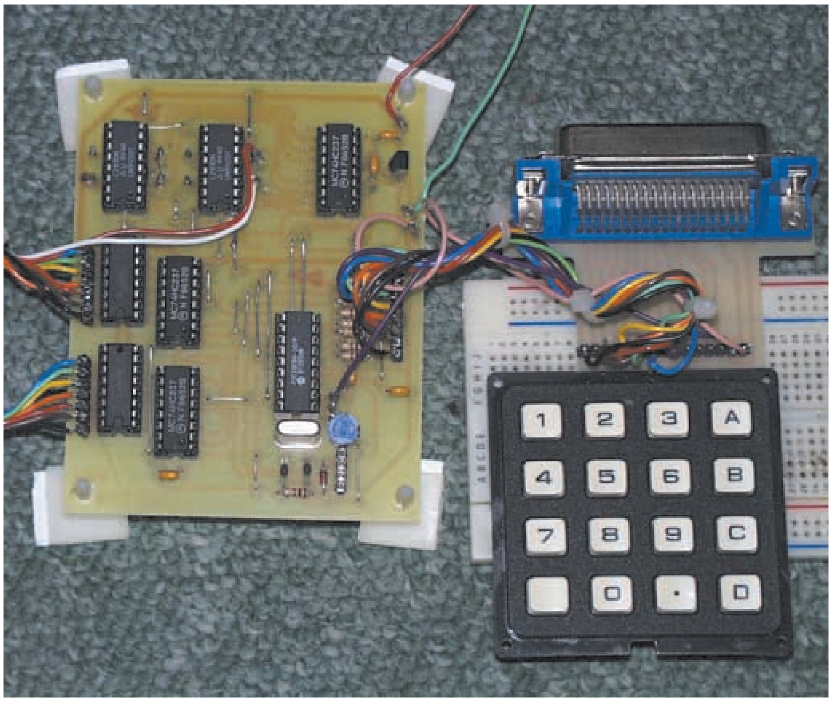
CONSTRUCTION

Printed circuit board component and track layout details are shown in Fig.8. This board is available from the *EPE PCB Service*, code 341.

Assemble in order of link wires first, including the one marked "Bank 1 Link" – this will be discussed under "Expansion". Note that some links go under the i.c. positions. Follow with the d.i.l. (dual-in-line) i.c. sockets and then continue in any convenient order. Insert 1mm terminal pins at the external connection points, but omit those alongside IC1 which are only needed if more than eight digits are to be controlled.

There are two choices of data input, as said earlier. They are connected to the board at the pins to the left of resistors R1 to R8.

If using the data entry keypad, connect its pins, as shown in Fig.9, to the similarly numbered points on the board. Keyboard


Prototype display controller board during development testing using a plug-in breadboard to temporarily connect a data keypad and a PC via a Centronics connector (mounted on the p.c.b. used with Teach-In 2000 Part 4 *– Feb '00).*

pin 9 is a ground (0V) connection for the pad's frame.

If using a PC as the data source, it needs to be connected from its parallel printer port to the board. The easiest way is to use a standard printer cable with pre-attached connectors. The "printer end" of the cable has a 36-way male D-type Centronics connector which requires a matching female type at the unit end. The latter should be hardwired to the board at the designated points using short lengths of insulated stranded wire. The pinouts for a right-angled female connector are shown in Fig.10.

Alternatively, the unit can be hardwired to a separate 25-way D-type male connector plugged into the back of the computer – 10-way ribbon cable would be ideal. The connector's pinouts are shown in Fig.11. Note that the "Error" line connects to the board pin situated near IC8. ("Error" is the name given in respect of that line's normal purpose when interfaced to a printer.)

Before inserting the d.i.l. i.c.s, do a thorough examination of the board for faulty assembly and soldering. Then only insert them after you have established that regulator IC9 is correctly supplying +5V at its output. Check this again once the i.c.s have been inserted.

Be aware that they are CMOS devices and require the normal handling precautions, discharging static electricity from your body by touching the bare metal of something earthed before handling them.

Adjust preset VR1 to a fully-clockwise setting (maximum pulse length) before testing the system.

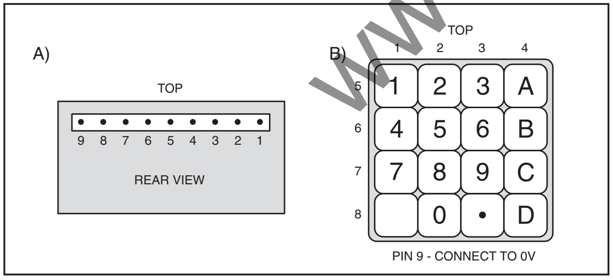


Fig.9. Keypad connection details.

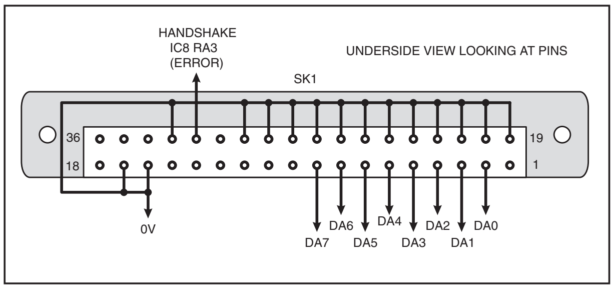


Fig.10. Connections to a 36-way D-type Centronics female connector used in conjunction with a standard PC parallel port cable.

PIN	FUNCTION
1	NC
2	NC
3	D OFF
4	D ON
5	E OFF
6	C ON
7	E ON
8	C OFF
9	F OFF
10	B ON
11	F ON
12	B OFF
13	G ON
14	A ON
15	G OFF
16	A OFF

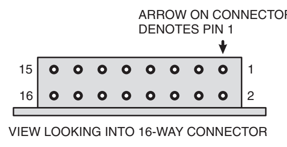


Fig.12. Pinouts for the connector mounted as part of the digit assembly.

DIGIT WIRING

Monstrous is again a term that can be used in respect of the digit connection requirements. The digits need to be wired in parallel back to the control board. However, although the manufacturers have provided a *single* connector on each digit, this only allows for one set of the 15 connection wires needed.

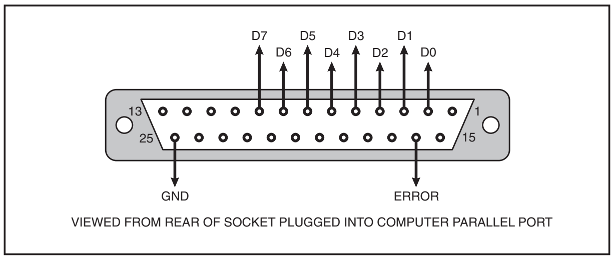
One would have expected two connectors, one for the cable harness arriving from the control board, another for the harness that then has to be connected to the next digit.

The author offers no recommendations about using the digit's own connector, although for the sake of good order, its pinouts are shown in Fig.12.

It was decided that it was easier to hardwire the connections to solder pads at various positions on the back of the digits. They are the pads to which the manufacturer's rectifier diodes (mounted inside the digit box) are soldered. The correct connection points were found experimentally and are shown in Fig.13. Ignore the unused pads.

Whereas the 14 segment wires of each harness are connected in parallel to each digit, each digit needs its own separate +12V power supply wire, originating from the control unit p.c.b. as shown in Fig.8. Make the +12V connections in numerical order in relation to the digit positions in the proposed display.

Before you fully interwire the digits, though, it is recommended that you just wire-up for the first one and check out the system.


D7
D6
D5
D4
D3
D2
D1
D0
13
1
25
15
GND
ERROR
VIEWED FROM REAR OF SOCKET PLUGGED INTO COMPUTER PARALLEL PORT
Fig.11. Alternative connections via a 25-way male D-type connector.

KEYPAD OPERATION

The software has been written so that on power being switched on the PIC sets its Port B for input with the input pull-up resistors active. A check is then made to see if the inputs are connected to any source that pulls them low.

Under keypad control (with no keys pressed) there is nothing to pull the pins low and so the PIC assumes that a keypad is the data entry source.

Having established that fact, the software goes into a perpetual loop scanning the keypad for keypresses. The software routine used is a variant of that described in the author's *Using PICs with Keypads* of Jan '01.

In response to any keypresses, look-up tables are used to relate that input value to the data to be sent to the digits. The first table (VALUE) allocates the keypress data to a numerical value between 0 and 15. Another table (TABLE) then relates that value to a binary sequence in respect of the digit segments to be turned on.

The sequence is in the right-to-left order (bit 0 to bit 7) of segment *A* to segment *G*. For example, binary 01111111 turns on all segments, resulting in the 7-segment display of numeral 8. Binary 00000110, on the other hand, only turns on segments *B* and *C*, resulting in numeral 1 being displayed. The full table is shown in Listing 1.

LISTING 1

```
TABLE:
    addwf PCL,F
    retlw %00111111      ; 0
    retlw %00000110      ; 1
    retlw %01011011      ; 2
    retlw %01001111      ; 3
    retlw %01100110      ; 4
    retlw %01101101      ; 5
    retlw %01111101      ; 6
    retlw %00000111      ; 7
    retlw %01111111      ; 8
    retlw %01100111      ; 9
    retlw %01110111      ; 10 A
    retlw %01111100      ; 11 b
    retlw %00111001      ; 12 C
    retlw %01011110      ; 13 d
    retlw %10000000      ; 14 blank
    retlw %01000000      ; 15 -
             GFEDCBA
```

Note that bit 7 in the 14th jump is set at 1. This prevents the PIC from returning a zero value from this location, which would otherwise be recognised as "no data entered from keypad".

Whilst it is suggested that decimal display values from 0 to 9 are retained, other segment arrangements could be provided for the other six positions by readers having their own PIC assembly-programming facilities, such as the author's *Toolkit Mk3/TK3* (Oct/Nov '01).

It is also worth recognising that 7-segment displays cannot in many instances be used to represent alphabet characters. For example, capital letter *A* can be represented, but lower case *a* cannot. Conversely, *b* can be, but *B* cannot (it would just look like an *8*).

Also note that any letters having diagonals cannot be represented, such as *K, M, N, Z*, nor can *T*. It is worth experimenting to see what characters can be represented, and what compromises you might have to make. You may recall that the author's *Teach-In 2000* series demonstration software illustrated the principle of 7-segment control.

Having established the segment code required, the PIC then has to send the corresponding data to the segments individually. From within a loop, the PIC reads each data bit position to see whether a segment should be On or Off. At each position it uses another look-up table (TABLE2 – see Listing 2) for the code needed to send to multiplexers IC4 and IC5 in order to control that bit.

The code also takes into account that the p.c.b. tracks are connected to the three control pins in the opposite order than might normally be expected (this was done for p.c.b. design simplicity). Only bits 6 to 4 are of importance in this table.

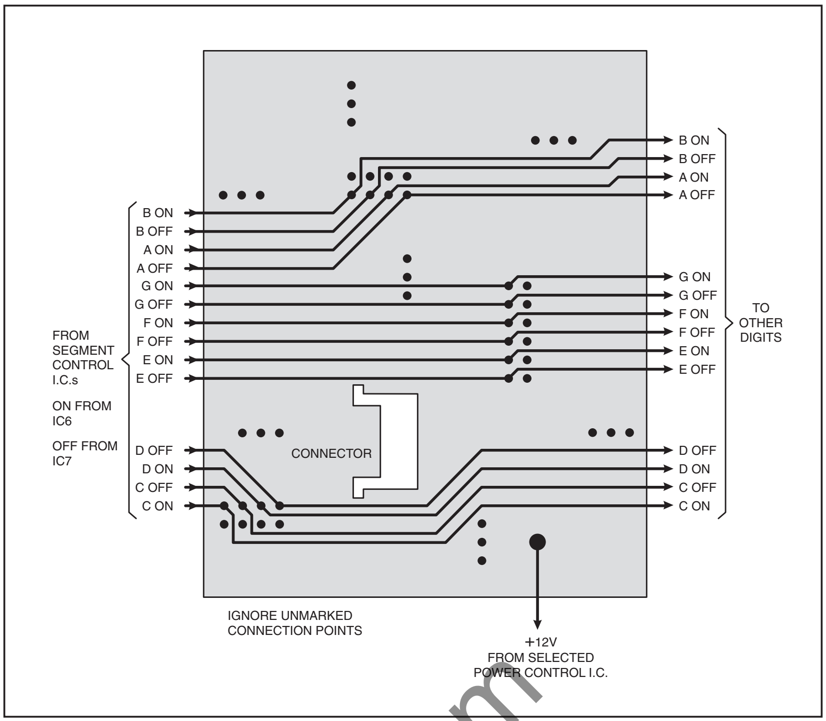
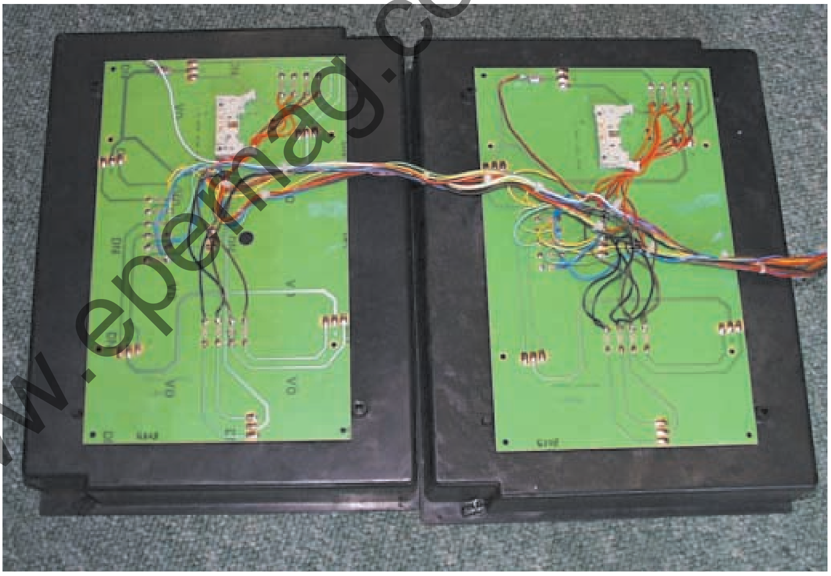


Fig.13. Wiring details for a single digit.



Interwiring between the two digits used during development.

LISTING 2

```
TABLE2:
        addwf PCL,F
        retlw %01000000    ; a
        retlw %00100000    ; b
        retlw %01100000    ; c
        retlw %00010000    ; d
        retlw %01010000    ; e
        retlw %00110000    ; f
        retlw %01110000    ; g
        retlw %00000000    ; -
```

The code is output on the 3-line common bus feeding to IC4 and IC5. Which of these i.c.s is activated depends on whether the segment needs to be turned on or turned off. To turn on IC4 (segment On), bit 7 in the code is set high. If IC5 is required (segment Off) bit 3 is set high.

It is also necessary to specify which of the digits is the target for the segment information. This data is set into the code's bits 0 to 2, representing the number (1 to 8) of the digit in the allocated bank, and destined for IC1 (see Fig.6).

DIGIT SELECTION VIA KEYPAD

Because the digits might be located away from the controlling keypad, and not be visible to the user, it was decided to allocate two keypad keys as digit stepping controls. At switch-on digit 1 is the default target, and any numeric data keyed in continues to be routed to it.

To choose Digit 2 instead, press keypad "D" (*D*igit step). This changes the control code fed to IC1, incrementing it from binary 000 to binary 001, so selecting digit 2. Display data is now repeatedly fed to this digit. Pressing "D" repeatedly steps through each digit position in turn, irrespective of whether the digit physically exists in the system.

To return to Digit 1 at any time press "C" (*C*lear back to start). It is not possible to step back individually from digit to digit. This, though, is a facility for which PIC-wise users could write a software routine. In this case it is suggested that key "B" is intercepted (*B*ackwards) in a similar way to which letters "C" and "D" are intercepted.

Each time the digit number is stepped forward, the software increments a 24-value counter (rolling over to 1 again following 24). This not only provides information on which digit is selected (from 1 to 8), but also on which Bank it is in (Bank 0 to 2), using yet another look-up table. This results in Port A pins RA0, RA1 or RA2 being selected as appropriate (in Bank order).

Referring back to Fig.6 again, it will be seen that IC2 and IC3 are shown to be under selection control by pin RA0. If additional IC2 and IC3 devices are used they would be allocated to one of the other Port A pins, RA1 or RA2, in that order of Bank.

In this way, 24 digits can be stepped through by pressing key "D" the required number of times. Yes, it tests the user's counting ability, but seemed the best solution considering the limited number of keys available.

The provision of monitoring via an alphanumeric liquid crystal display was considered, but was rejected on the grounds of adding complexity to a moderately simple design.

PIC-knowledgeable readers could probably add l.c.d. facilities if needed. There are numerous examples of l.c.d. control in many of the published *EPE* PIC projects (especially in the author's designs). Such a routine could be integrated almost as a "library" file.

It is suggested that l.c.d. control is basically via Port B with the exception of the l.c.d. E line, which is better suited to control by the otherwise unused pin RA3 (it is only used when under PC control).

Line E cannot be satisfactorily controlled by Port B as all pins are in use for other purposes, which would cause undesirable l.c.d. response. It is only Line E that is critical in this context.

COMPUTER CONTROL

When under computer control, data is fed to the PIC via the same connections as the keypad (but preferably in the keypad's absence). It is in a different coding format to that used with the keypad, however.

Because of the full range of keys on a PC keyboard, is it is possible to send a much greater variety of data to the digits. On recognition by the PIC that a PC is connected to it (see earlier), it goes into a different monitoring routine (COMPROG).

Synchronisation between the PIC and PC is maintained by using two handshake lines at the PIC end of the system, pins RA3 and RA7 as mentioned earlier. Port B pull-up resistors are turned off in this mode.

The first significant handshake action the PIC takes following switch on for PC mode, is to set pin RA3 high. This indicates to the PC that the PIC is ready to receive data. The PIC then sits in a holding loop until acknowledgement from the PC is received.

The PC software in its turn holds its printer port output DA7 low and waits for the RA3 = high signal to arrive via its printer port "Error" line. Having received this signal, however, it takes no immediate action, but waits for a keyboard key to be pressed.

Having received a keypress, the PC relates it to a lengthy look-up table that holds segment data in respect of keypresses. If segment data has been allocated to that key, it is output as seven bits (same relationship as with keypad data) plus bit 7 set high. It then remains in another holding loop until the "Error" line goes low.

The PIC, recognising that its RB7 pin has gone high, accepts the incoming 7-bits of RB0-RB6 data as valid. It immediately acknowledges this to the PC by setting line RA3 low.

The PC, having accepted this acknowledgement, is now free to wait for another keypress, but will not send it until the PIC signals that it is ready.

Between accepting bytes of data, the PIC sends the segment data serially to the selected digit in a similar fashion to that described earlier. On completion of each digit's output, the PIC again sets handshake line RA3 high, asking for more PC data.

DIGIT COUNT SELECTION

The PC program has been written so that it can be set to the exact number of digits in use, unlike the keypad software which always expects 24 digits. It also provides the facility to select which printer port register address is used.

On running the program, the screen shown in Fig.14a will be displayed. The three possible printer port registers addresses are displayed at the top. It is necessary to select the one appropriate to your PC's configuration. Most likely it will be address 378 hex, but could be hex 278 or 3BC. Select the address by pressing 0, 1 or 2 (pressing any other key, including <ENTER>, always selects 0, i.e. address 378).

If you do not know which address your PC uses, try all three. The system will show you have the correct one when it proves that it can send data to the displays. (The PIC board must be free of assembly errors of course!)

Having selected the register, the screen changes to that in Fig.14b. Underneath the main title you are asked to enter the number of digits that you wish to be controlled, with a range of 1 to 24. Values outside this range are not accepted.

At the bottom of the screen are displayed the characters which can be sent for display via the 7-segment digits. With the exception of the control keys mentioned next, this represents the full range of keys that are functional. Any others will be ignored by the program (although you can add to the range as discussed later).

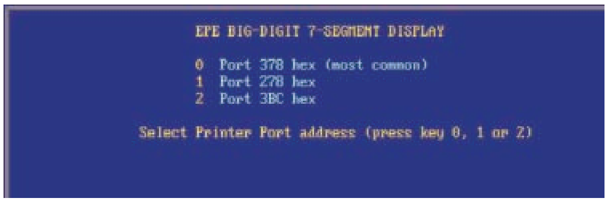
To either side of the screen are quoted the commands available when the program is in full control mode. The <ESC> (escape) key causes the program to restart from its beginning and may be used at any time. Pressing the <CTRL> and <BRK> keys simultaneously causes the program to end. This is the only way in which it can be halted and exited.

Otherwise, all keyboard characters shown in the bottom line are available for output to the digits. Acceptable keypresses are responded to immediately, and data is output to the digits in sequence, the PIC's digit count being incremented following receipt of each character. When the final digit in the sequence has been triggered, the count automatically recommences from Digit 1.

When entering data for output to the digits, pressing <ENTER> causes the PIC to reset the digit count back to Digit 1. Pressing the space bar causes the next digit to be cleared (no segments showing).

PC FIRST, PIC SECOND

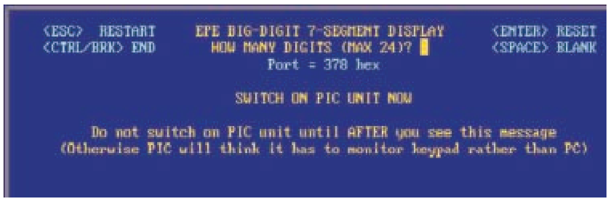
In the mid-screen area you are told that you should switch on the PIC unit "now". As said earlier, when the PIC program is first switched on, the PIC examines Port B to see whether its pins are high or low. If high, keypad control is assumed. On running the PC program, however, its first activity is to set its printer port lines low. On reading Port B being low, the PIC knows that PC control is required.



```
EPE BIG-DIGIT 7-SEGMENT DISPLAY

0  Port 378 hex (most common)
1  Port 278 hex
2  Port 3BC hex

Select Printer Port address (press key 0, 1 or 2)
```

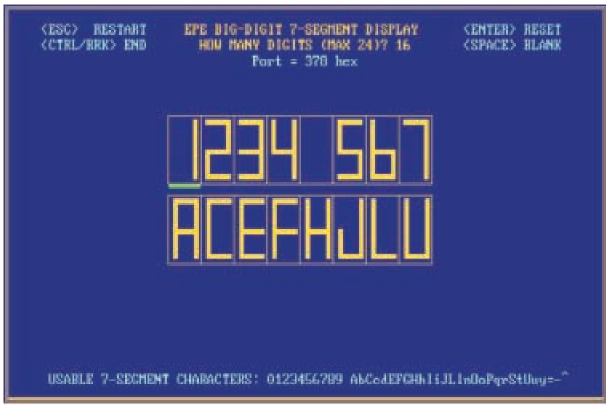


```
<ESC>  RESTART     EPE BIG-DIGIT 7-SEGMENT DISPLAY     <ENTER> RESET
<CTRL/BRK> END       HOW MANY DIGITS (MAX 24)? █       <SPACE> BLANK
                           Port = 378 hex

                        SWITCH ON PIC UNIT NOW

           Do not switch on PIC unit until AFTER you see this message
      (Otherwise PIC will think it has to monitor keypad rather than PC)
```

Fig.14. Sections of Big Digit PC program setup screens, (a) printer port selection, (b) digit quantity selections.



```
<ESC>  RESTART       EPE BIG-DIGIT 7-SEGMENT DISPLAY       <ENTER> RESET
<CTRL/BRK> END         HOW MANY DIGITS (MAX 24)? 16        <SPACE> BLANK
                            Port = 378 hex

                          1234 567
                          ACEFHJLU

USABLE 7-SEGMENT CHARACTERS: 0123456789 AbCcdEFGHhIiJLlnOoPqrStUuy=-^
```

Fig.15. Example of PC screen during digit control.

Consequently, do not switch on the PIC unit until you see the screen now being discussed. When you have switched on the PIC, then enter the number of digits to be controlled and press <ENTER>.

The program then enters its full operational mode, first drawing on screen the same number of boxes as the number of digits specified. These boxes represent the 7-segment digits and display the same characters.

Next the program sends data for numeral 8 to all digits required. It then sends a reset command to the PIC, resetting it for Digit 1, after which it sends data to clear all required digits, again followed by a reset command.

This action has three functions, to synchronise the PIC with the computer's order of digits, to prime the PIC so that it knows which segments are in which state, and thirdly to clear any existing display data.

In the latter context it is worth recognising that the segments can be set by hand without damaging them. They are only balanced on light-duty pivots, freely responding to the electromagnetic fields generated by their coils. It is quite possible that someone could have set them by hand to random positions. (In a "field" situation, it is advisable to enclose the digits to prevent this happening – and of course to protect them from the "elements".)

From this point onwards, pressing any recognised key causes the data to be displayed sequentially, with the count returning to zero (Digit 1) after the final digit (or on pressing <ENTER> as described earlier). An example PC screen display is shown in Fig.15.

PC CONTROL DATA OPTIONS

Because of the greater variety of segment codes that can be generated via the PC than with the keypad, there is the option to program the PC software with any segment combination required.

The data is held in a look-up table which can be added to by readers who have QBasic or QuickBASIC resident on their PC. The data is held as in the format extract example shown in Listing 3, in the bit order of segments ABCDEFG (the opposite order used by the PIC software's table).

LISTING 3

```
DATA 01111110
DATA 10110000
DATA 21101101
DATA C1001110
DATA c0001101
DATA D0000000
DATA d0111101
DATA K0000000
DATA k0000000
DATA " 0000000"
DATA "^1100011"
```

When the program is started, all data statements are "Read" and analysed. The first character in each data string holds the keyboard character that represents the following 7-bit segment data. Its ASCII value is taken and the remaining seven bits in the data are stored in a string array, seg$(x), at the address corresponding to the ASCII value.

For example, in the first case, "01111110", the leading "0" is the first character. Its ASCII value is 48 and so the rest of the data string ("1111110") is stored at string array position seg$(48). In the fourth case, "C" is the character, having the ASCII value 67, so its 7-bit string data is stored at seg$(67).

Note that some data statements have had to be enclosed in quotes so that the program recognises the associated character correctly (the last character in the above list cause the "degrees" symbol to be displayed when the "^" is pressed (as in 20^{o}C). The one before it is for the space bar (turns off all segments in a digit).

You will see instances where the character may be in upper or lower case, and in some cases both. If the value following the character contains one or more "1"s, the equivalent character *can* be generated on a 7-segment display. In the other cases, all zeros, the character *cannot* be formed using a 7-segment display.

If a character is not included in the table, a value of zero is returned if its key is pressed. All unacceptable keypresses are ignored.

ALLOCATING SEGMENTS

For such "unacceptable" keys, however, a segment or PIC control code can be allocated separately. For instance, the program allocates the code "00000001" when the <ENTER> key (ASCII 13) is pressed. The PIC has been programed to recognise this bit combination as the command to reset the digit number count to Digit 1, in a similar way to that in which it responds when the "D" key on the 4 × 4 data keypad is pressed.

You could, for example, allocate specific codes for the PC's forwards/backwards cursor keys. The PIC could then be told to step the digit count value backwards or forwards without causing the display data to change. Then, on pressing another key, its character would be displayed at the new digit address.

Such a facility would be of help in a display having many digits and where only one or two might need to be changed at any time. This would remove the need to key in data for all digits in the full display when only a few might need changing.

Another option open to those who are familiar with QB programming is to write a code routine that allows a string of characters to be entered via the keyboard as a sentence (using INPUT instead of INKEY$). This would not be transmitted to the PIC until the <ENTER> key had been pressed. Each character would then be sent automatically in sequence to successive digits as required.

SETTING PULSE LENGTH

So far the discussion has assumed that the length of the control pulse that activates the segment coils is correct. Setting preset VR1 earlier to a fully clockwise position sets the length to the maximum design limit. It is likely that the pulse can be shortened, so speeding segment changes.

The simple data sheet received indicated that a pulse length of about 0·25 seconds was required. Experiments with the digits showed that it could be much shorter. Although there was a slight variation in minimum operational pulse length for the various segments, the requirements were typically found to be about 70 milliseconds, but cannot be guaranteed in other assemblies (hence the need for user-adjustment rather than specifying the length as an accurate timing within the software).

A 70ms pulse length is generated with preset VR1 at a roughly midway setting. The maximum pulse length that can be set is about twice that. These figures are based on the PIC being run at 4MHz.

Once you have ascertained the correct response of the segments using a long pulse set via VR1, it is worth experimenting to find the lowest VR1 setting at which the segments will respond. This will speed the rate at which the displays can be changed.

The digits will not respond if the resistance is set too low. An intermediate stage may also be found in which some digits respond but not others. Avoid setting VR1 to a nil resistance position which will overload RA4 when it is in output-low mode (the PIC is internally protected against brief overloads – but do not sustain this condition).

It is worth noting that the software has also been written to speed segment changing. The status of each segment is recorded in the PIC's memory. When a new character is to be displayed on a particular digit, the digit's current segment status is checked against the segment requirement for the new character. If any segments match, they are ignored by the output routine, so saving one pulse duration – which can be a significant saving when many digits are in use.

SCHMITT PULSING

This now brings us to a software/hardware aspect that has not been used before in an *EPE* project – analogue control of frequency via a digital input.

You are no doubt familiar with the type of circuit in which a single Schmitt trigger inverter is used with a resistor and capacitor in order to generate a frequency (an RC oscillator). The technique used in Big Digit is similar.

The PIC16F84 has a Schmitt trigger input, pin RA4. Referring to Fig.7, the resistance is provided by preset VR1, and the capacitance by C5. Initially software sets RA4 as an output set for logic 0. This discharges C5. RA4 is then set as an input, allowing current to charge up C5 via VR1.

When the Schmitt threshold is reached, the software responds to this as an input change from logic 0 to logic 1. It immediately sets RA4 as an output at logic 0 again, discharging C5, and then resets RA4 as an input once more, and so the cycle can continue for as long as the software requires it.

In this design, 16 waveform cycles are used, which allows a lower value capacitor to be used than with a single cycle. It also increases the capacitor's discharge rate and reduces current flow when RA4 is briefly set low. Listing 1 shows the full pulse delay generation routine.

The frequency of oscillation can be changed by varying VR1 or by using a different value for C5.

EXPANSION

As said earlier, additional banks of eight digits can be controlled. In this case IC2 and IC3 need to be duplicated on a stripboard layout. Their pins should be connected identically to those in Bank 1, referring to Fig.6. The connection points on the p.c.b. are those alongside IC1, previously left unused.

The difference is that their enable pins (1 and 9) need to be controlled by a different Port A pin, RA1 for Bank 2, and RA2 for Bank 3. It is permissible to omit IC3 in the final bank if the digit count does not require it.

If more than three Banks are needed (more than 24 digits), pins RA0 to RA3 should be wired into another 74HC237, mounted on stripboard, at its pins A0 to A2. The outputs would then be used as the Bank Select lines for up to eight pairs of IC2 and IC3.

If using the extra 74HC237, remove the p.c.b. link wire marked Bank 1 Link. Connect point Y to pin Y0 of the new multiplexer. Point X then becomes the point to be regarded as the RA0 connection.

The software for the PIC and the PC will need to be modified to cope with more than three banks of digits. For this reason, only readers highly familiar with programming in both PIC and QB languages should undertake this option.

To such experienced programmers, the changes required should be obvious, but the author cannot offer advice on it. Nor can advice be offered on a breadboard layout for any additional chips added.

Note that it will be necessary to change regulator IC9 to a standard 7805 +5V 1A type if additional copies of IC2/IC3 are added (each extra chip adds about 20mA to the power drawn from the +5V line – see earlier).

QB programmers will recognise that the multiplexing circuit could be controlled directly from the PC's printer port data lines, omitting the PIC entirely. The port's other control lines could then be used in place of the RA0 to RA2 connections. The QB software would largely need to be rewritten, of course.

LISTING 4 Send pulse to segment control

```
PULSEIT:
        movlw %00010000      ; number of cycles required
        movwf PULSECNT
PULSE2:
        btfss PORTA,4        ; has bit 4 gone high (cap charged
                               up enough)?
        goto PULSE2          ; no, repeat check
        bcf PORTA,4          ; yes, set bit 4 low to discharge cap
                               again
        PAGE1
        bcf TRISA,4          ; set bit 4 as output
        PAGE0
        nop                  ; brief wait discharge capacitor
        PAGE1
        bsf TRISA,4          ; set bit 4 as input
        PAGE0
        decfsz PULSECNT,F    ; repeat for set delay loop time
        goto PULSE2
        return
```

VALEDICTUM DIGITALIS!

Two digits were sent to the author for experimentation. As described in this article, the resulting design is intended to drive up to at least 24 digits, and up to 64 with modification. Obviously this ability has not been fully proved in practice. However, extensive bench-tests and simulations have been made using the two digits and it is believed that the claims are valid. If you find any aspect that does not justify this belief, let the author know via *EPE* HQ (*NOT* via the *Chat Zone* as messages posted there may be overlooked).

The author hopes that readers will find ways in which the PIC and QB programs can be enhanced and write additional routines to suit their own needs. His intention has been to show with this design how the Big Digits can be controlled, and to provide an elementary framework within which readers can work to suit their own needs and the number of digits actually used.

Readers who do not wish to tailor the programs, though, will find that the software is perfectly usable as it stands, and that it provides a reasonable method of controlling the digits, whether just one is used, or many more. We would be interested to know how many you use and in what applications.

RESOURCES

The software for this design is available on 3·5in disk (for which a nominal handling charge applies) from *EPE* Editorial office, or free via the *EPE* ftp site (path PUB/PICS/bigdigit). The easiest route to the ftp site is via the link at the top of the main *EPE* web page at **www.epemag.wimborne.co.uk**.

The PIC software is supplied as a source code (ASM – TASM grammar), HEX code (MPASM) and OBJ code (TASM). It was developed using *EPE Toolkit Mk3/TK3*. The PC program is supplied as a standalone program (EXE) and as QBasic/Quick-BASIC source code (BAS).

The PIC configuration required is XTAL XS, POR on, WDT off. This is embedded in the ASM and HEX codes, but readers using the TASM OBJ code must configure the PIC in the usual separate manner.

Ensure that you read this month's *Shoptalk* page for details of component buying for this project.

ACKNOWLEDGEMENT

The author thanks Display Electronics (**www.distel.co.uk**) for providing the Big Digits for experimental use in the development of this project.