Interfacing COMPUKIT

Part 2 D.E.Graham

THIS MONTH we discuss the use of the MC 6821 PIA on the Decoding Module, and look at the way in which digital data may be input to Compukit using both this, and sets of tristate buffers.

The 6821 PIA

Motorola's 6821 PIA provides two 8-bit input-output ports together with interrupt and peripheral control facilities. It achieves this through the operation of six internal registers; three for each port. One of these carries the actual data passing through the port, a second determines the direction in which data is carried, while the third is a control register largely associated with the generation of interrupts, and the operation of control lines to peripheral devices. Somewhat unusually, although the chip uses six registers, it occupies only four addresses, and as a consequence accessing its registers is not as straightforward as it might be.

It circumvents the problem by using bit 2 of each control register to determine whether the other two addresses refer to the data direction register or to the port data register. Table 2.1 gives the disposition of the six registers, and the four corresponding addresses used in the Decoding Module. Accessing the two control registers is quite straightforward. The command POKE 61341, W will place the value W into the control register of port A, while POKE 61343, W will do the same for port B. In order to access the data direction register, bit 2 of the control register must first be set to zero. POKE 61341, O will achieve this for port A. The command POKE 61340, Y will then configure port A for input if Y=O, or for output if Y=255. Bit 2 of the control register must then be set to 1 (eg. by POKE 61341, 255) to achieve access to the port itself, and this completes the initialisation procedure. PEEKing or POKEing to 61340 can now be used for input or output of data.

This is not really as difficult as it sounds, and Table 2.2 gives the four complete sets of commands for configuring either port for 8-bit input or output. Note that on Reset, or at switch-on, all registers are automatically zeroed, so that a number of commands in this table may be omitted in certain situations; and of course it is only usually necessary to configure the ports once at the beginning of a program, and all further Reads or Writes can be accomplished with a single PEEK or POKE.

In the examples discussed so far, the data direction register has been loaded with either a zero (for input) or 255 (for output). In practice each of the 8 bits of the two ports is individually controllable in this respect, with one binary digit of the data direction register controlling one bit of the corresponding port; a zero signifying input, and a 1 an output. Thus whilst placing 255 (11111111 binary) in the data direction register will configure all 8 bits for output, the number 15 (00001111 binary) would cause the top 4 bits to be set for input, and the lowest 4 for output, and so on.

Although the registers of ports A and B are addressed in an identical fashion, the two ports differ electrically in certain respects. Both have a drive capability of two TTL gates. But on input, the output circuitry of port B adopts a tristate condition, whereas that of port A does not. Port A's inputs are accordingly taken high by internal pull-up resistors, and require an effective resistance of 1k or less to earth in order to render them low. The voltage sensitivities of the two ports is also somewhat different. Port A takes voltage lower than about 1·4 as a logical zero, and higher than about 1·6 as a logical 1, whereas the two corresponding voltages for port B are about 0·7 and 3·0.

In a later article in this series we will look at the use of the peripheral control lines of the 6821. We now turn to the construction and testing of the Decoding Module.

Table 2.1. 6821 Registers

Address	Control Bit Reg A	Control Bit Reg B	Function	
61340	1	X	Port Data Register	Port A
61340	0	X	Data Direction Register	Port A
61341	X	X	Control Flag Register	Port A
61342	X	1	Port Data Register	Port B
61342	X	0	Data Direction Register	Port B
61343	X	X	Control Flag Register	Port B

Table 2.2

OUTPUT

For output on Port A P=61340	For output on Port B P=61342	Function
POKE P+1, 0	POKE P+1, 0	codes for data direction register
POKE P, 255	POKE P, 255	sets data direction to output
POKE P+1, 255	POKE P+1, 255	code for peripheral register

Further commands of *POKE P, W* will now place W on Port A or B

INPUT

For input on Port A P=61340	For input on Port B P=61342	Function
POKE P+1, 0	POKE P+1, 0	code for data direction register
POKE P, 0	POKE P, 0	sets data direction for input
POKE P+1, 255	POKE P+1, 255	code for peripheral register

calls of PEEK (P) will now return the data at port A or B

CIRCUIT OPERATION

A full circuit of the Decoding Module is given in Fig 1.5. Device IC1, a 13-input NAND gate, provides the complete decoding of the base address of the system. As the circuit stands, with a single inverter in address line A12, this is EF80 hex (61312 dec). It will be seen however, that a number of pads are provided to engage two further inverters in address lines A11 and A13. This allows the user, by cutting tracks and wiring between the pads on any of the three lines A11-A13 to set the base address at 7 other possible locations. The various permutations are given in Table 1.10. It should be noted that the top two of the 8 possible base addresses fall within the UK101's 2K monitor, and should therefore be avoided! The remaining 6 base addresses and accompanying blocks are unused by both the Compukit and the Superboard II. Since however, all software for the series will assume a base address of EF80, it may be simplest to leave all three sets of pads unaltered.

The output of IC1 is taken to IC2, a 74LS138 3-to-8 line decoder, which decodes address lines A4–A6 to produce eight 16-byte blocks (BL0–BL7). The decoding of these blocks has not been made conditional of Ø2 since they are generally intended for use with multi-register devices which often possess a separate master Enable pin, which when connected to Ø2 satisfies all timing requirements of the chip. Some devices, such as the 6522 VIA, require the chip select lines to settle before Ø2 goes high; so that if the output of IC2 had been made conditional on Ø2, no settling time would be given, and the device would not respond to the CPU's attempts to select it.

Device IC3, a 74LS154 4-to-16 line decoder, provides the sixteen address-decoded Write lines in response to BL0, R/W and Ø2. The outputs of both the 74138 and the 74154 are active-low, and a number of IC3's outputs have been inverted for convenience of subsequent use. Lines W5 and W6 have, in conjunction with R6, been further decoded to provide a pair of signals BDIR and BC1 for use with an AY-3-8910 or 8912 programmable sound generator. This decoding is all that the PSG requires to both write to, and read from its full complement of registers, and details of the use of these lines will be given later in the series.

IC4 (74LS138) decodes 8 Read lines in response to address lines A0–A3, $\overline{\text{BL0}}$, R/W and Ø2. The reason why 16 Write lines have been made available, and only 8 Read lines, is largely the additional Write requirement imposed by the use of 7-segment l.e.d. readouts, to be featured later. The addresses of these 24 lines are given in Table 1.9. As will be seen, a number of these have been ear-marked for particular projects in the series.

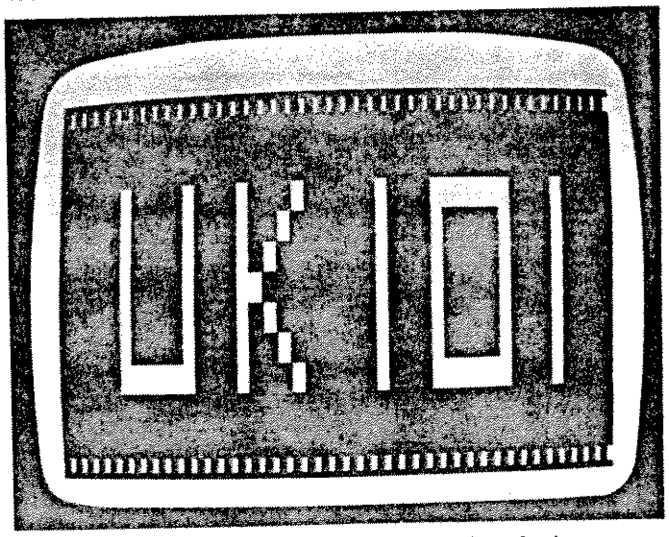
The NAND gate IC8A is used to produce the DD control signal. This is brought low when a Read is carried out at any of the addresses within the 128-byte block used by the module. The reason why it is not simply connected directly to the inverted R/W line is that this would cause the data bus of the CPU to receive extraneous noise from all interfaces during every memory Read cycle, whereas with the present circuit, interfaces are only given access to the CPU when the 128-byte block is called up.

As may be seen from Fig. 1.5, the 6821 PIA (IC9) is selected by the $\overline{\text{BL1}}$ line. Address lines A2 and A3 are wired to high-Enable Chip-Select pins 22 and 24, so placing the four addresses used by the PIA at the top of Block 1 (61340–61343 dec). Lines A0 and A1 are used internally by the PIA for decoding within the 4-byte block. Since block 1 is 16 bytes wide, there is clearly room for a further 3 PIAs to be located within it. This could be accomplished by wiring additional PIAs exactly as for IC9, but with inverters in either A2 or A3 or both; although it would probably be easier to locate them in one of the 6 unused blocks (BL2–BL7) whose signals appear on SK6.

The PIA's pin 25 is a master Enable against which all of its operations are timed. It is connected directly to the Ø2 clock. Pin 34 is a 6502-compatible $\overline{\text{RESET}}$ line. This has not been taken to the UK 101's Reset circuitry; one of the reasons for this being that the Compukit $\overline{\text{RESET}}$ signal does not actually appear at the expansion socket. Pin 34 is taken instead to a pair of pads intended for the connection of a pushbutton which may be used to simultaneously reset all devices wired to the Decoding Module. Capacitor C4 is used to give a power-on Reset. It does this by simply holding the $\overline{\text{RESET}}$ line low for a fraction of a second after power-up. A similar technique can be used on the Compukit itself: connecting 100 μF from pin 40 of the 6502 to earth will ensure that the D/C/W/M? message appears instantly at switch-on.

The Decoding Module power supply circuitry is quite straightforward, and employs a 7805 regulator to produce +5 volts at about 500 mA, and a zener stabiliser to give −5 volts at about 30 mA. The Module draws about 100 mA from the positive supply, leaving ample in reserve for driving external devices. The negative supply is generally intended for use with dual polarity analogue i.c.s that will be encountered in D/A and A/D conversion later in the series, and is not used by the Module itself.

The power supply requires a 9-0-9V a.c. transformer rated at 1A. It should be possible to tap Compukit's 3A transformer for this purpose providing it is not already heavily loaded.


Large characters produced by the joystic drawing routine

SK1 — TO COMPUKIT EXPANSION SOCKET — 1 off 2 × 22 pin edge connector
SK2 — 40 PIN DIL — ALLOWS FOR FURTHER EXPANSION OF COMPUKIT
SK3 & SK4 — 16 PIN DIL SOCKET — PORTS A & B OF PIA
SK5 — 24 PIN DIL SOCKET — DECODING MODULE OUTPUT
SK6 — 1 OFF 2 × 25 PIN 0·1″ EDGE CONNECTOR — DECODING MODULE OUTPUTS

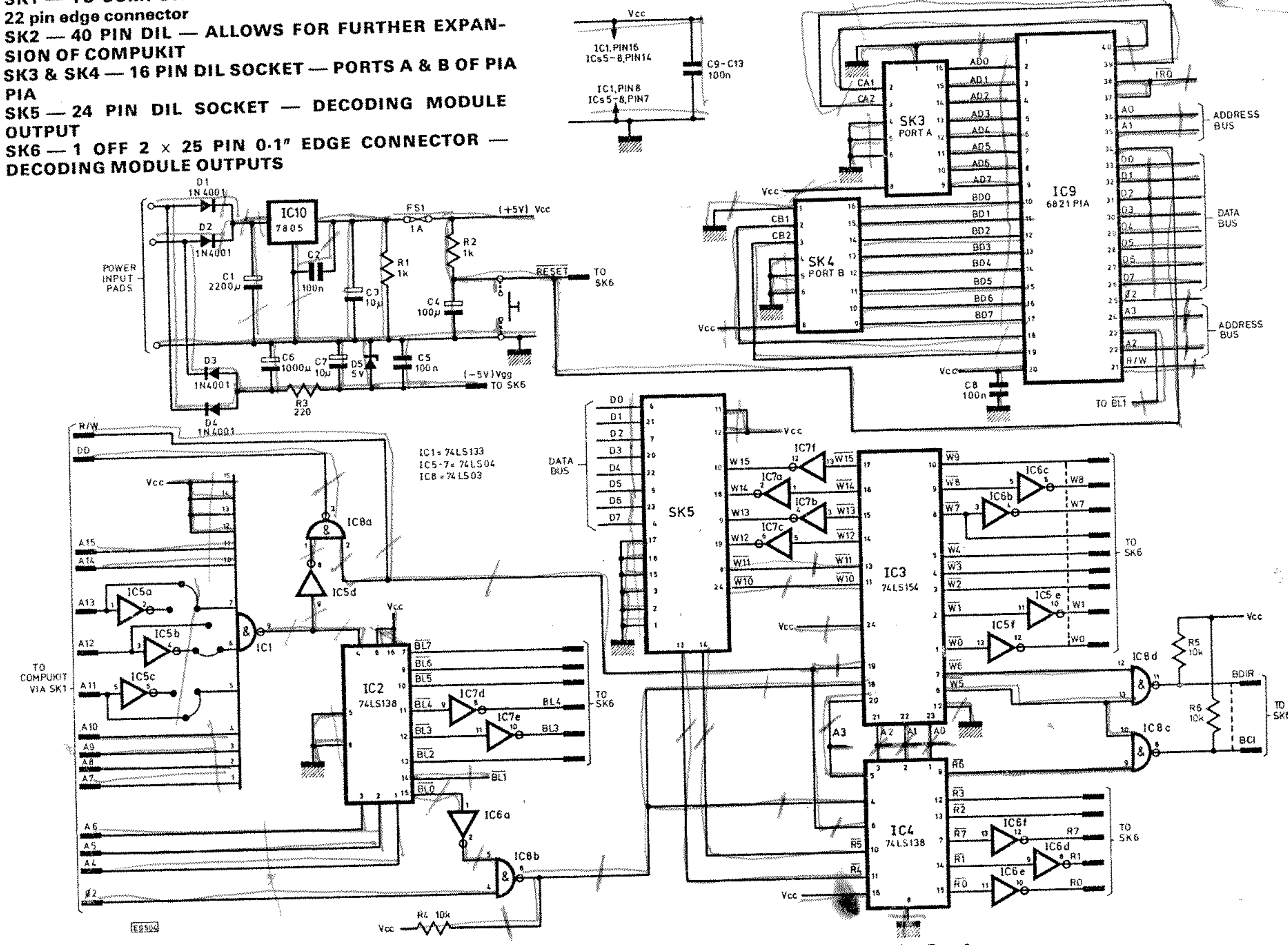


Fig. 2.1. Full circuit diagram. See Points Arising in this issue, concerning Part 1

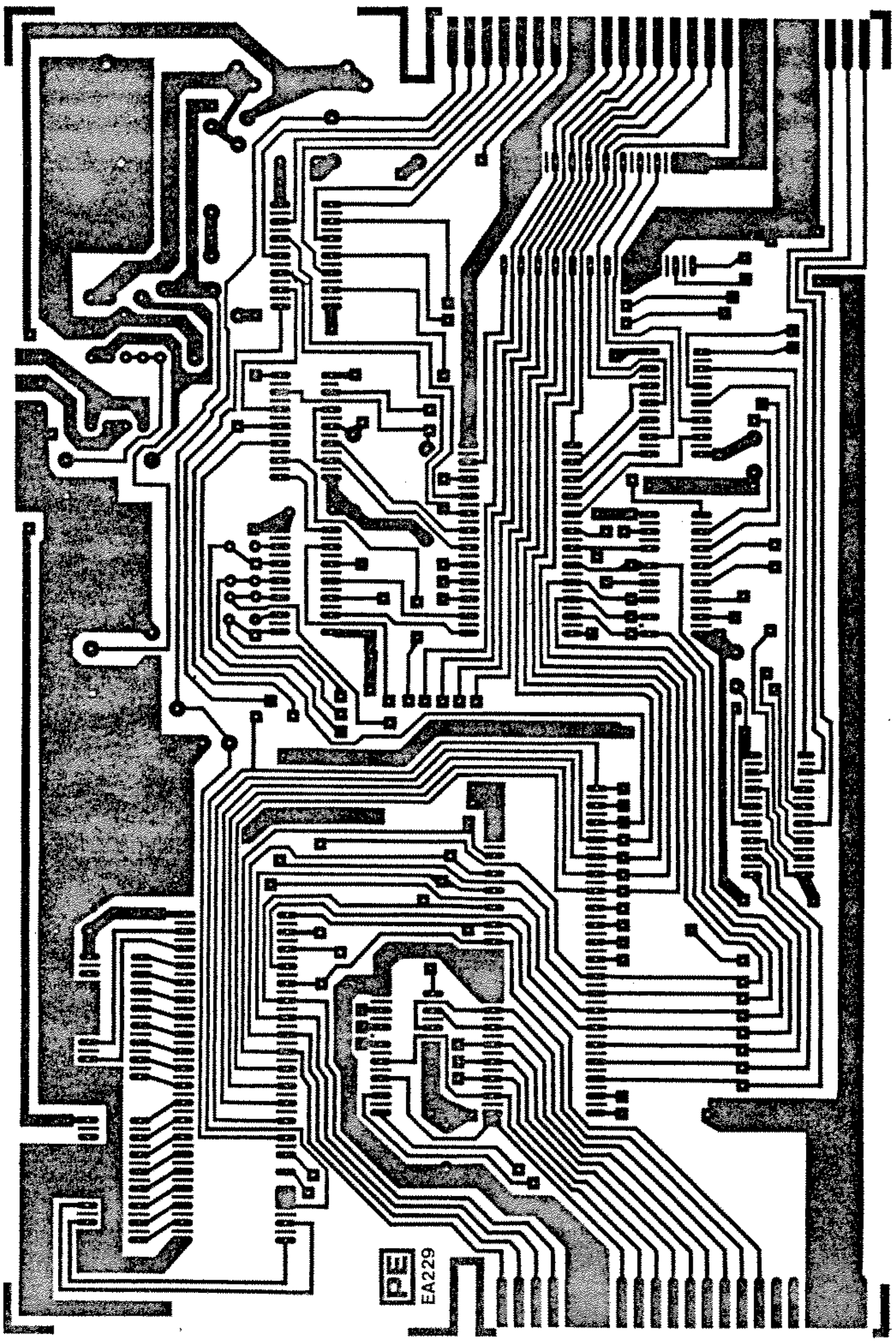


Fig. 2.2. Copper-side p.c.b. layout (actual size)

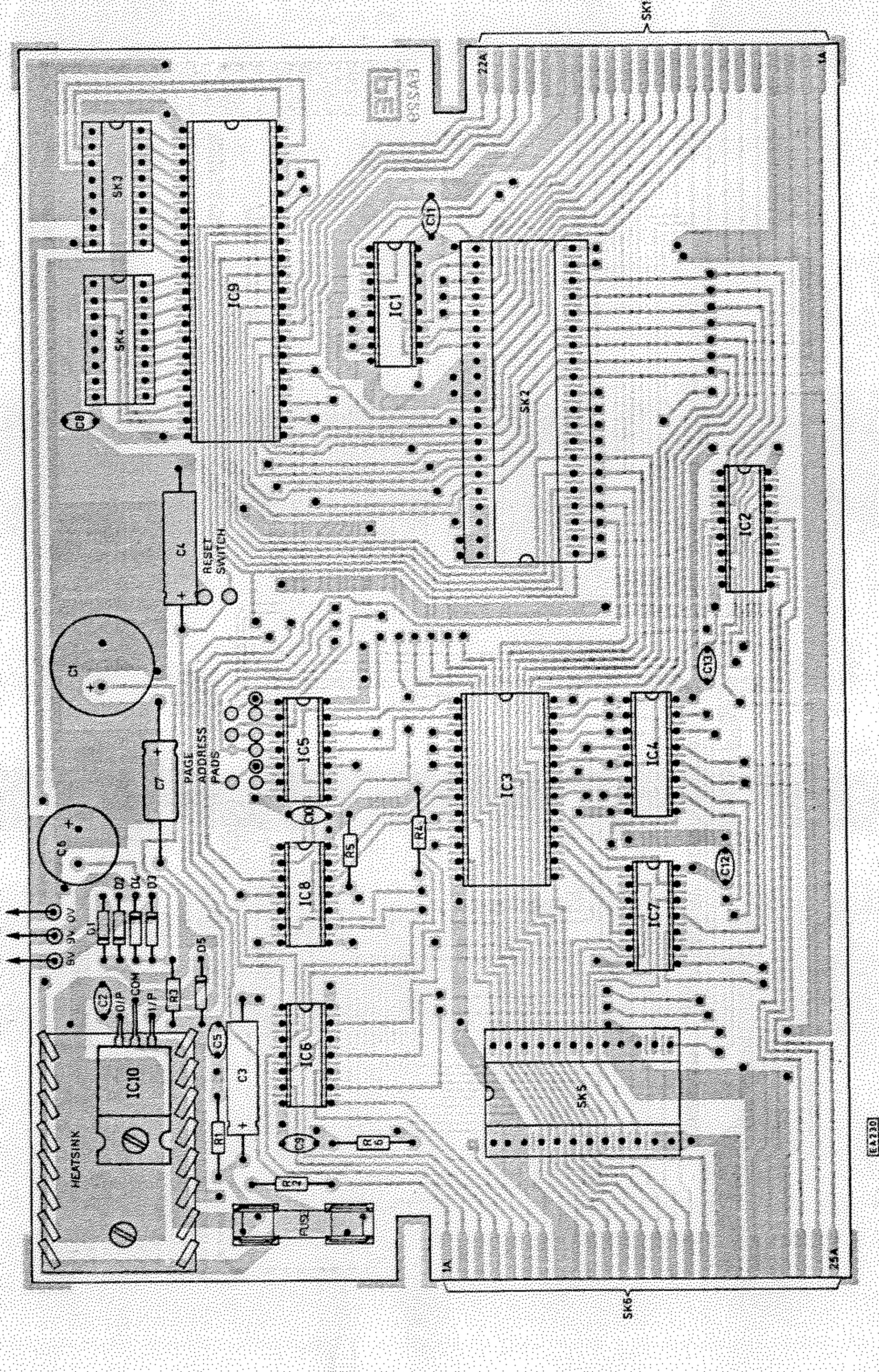


Fig. 2.4. Component layout. A complete kit of parts for the Decoding Module, including i.c.s, p.c.b., headers, edge connectors, ribbon cable etc., (but excluding transformer & 8T28s) is available from: Technomatic Ltd., 17 Burnley Road, London NW10. Price: £27.50 exc. VAT. Post free

J_1 EXPANSION SOCKET.
EDGE CONNECTOR

1	$\overline{IRQ}$	11
2	$\overline{NMI}$	12
3	DD	13
4	D_0	1
5	D_1	2
6	D_2	3
7	D_3	4
8	GND	31 & 32
9	GND	
10	GND	
11	SPARE PIN	
12	A_2	17
13	A_1	16
14	A_0	15
15	A_3	18
16	A_4	19
17	A_5	20
18	A_6	21
19	A_7	22
20	A_8	23
21	A_9	24
22	A_{10}	25
23	A_{11}	26
24	A_{12}	27
25	A_{13}	28
26	A_{14}	29
27	A_{15}	30
28	GND	31 & 32
29	GND	
30	GND	
31	$\emptyset 2$	10
32	R/W	9
33	D_7	8
34	D_6	7
35	D_5	6
36	D_4	5
37	GND	31 & 32
38	GND	
39	GND	
40	GND	

"SK6"

2	$\overline{BL7}$
3	$\overline{BL6}$
4	$\overline{BL5}$
5	BL4
6	$\overline{BL3}$
7	$\overline{BL2}$
8	$\overline{BL1}$
9	W0
10	W1
11	$\overline{W2}$
12	$\overline{W3}$
13	$\overline{W4}$
14	$\overline{W7}$
15	W7
17	W8
18	$\overline{W9}$
19	BDIR
20	BC1
21	R_0
22	R_1
23	$\overline{R_2}$
24	$\overline{R_3}$
25	R_7
26	

16—

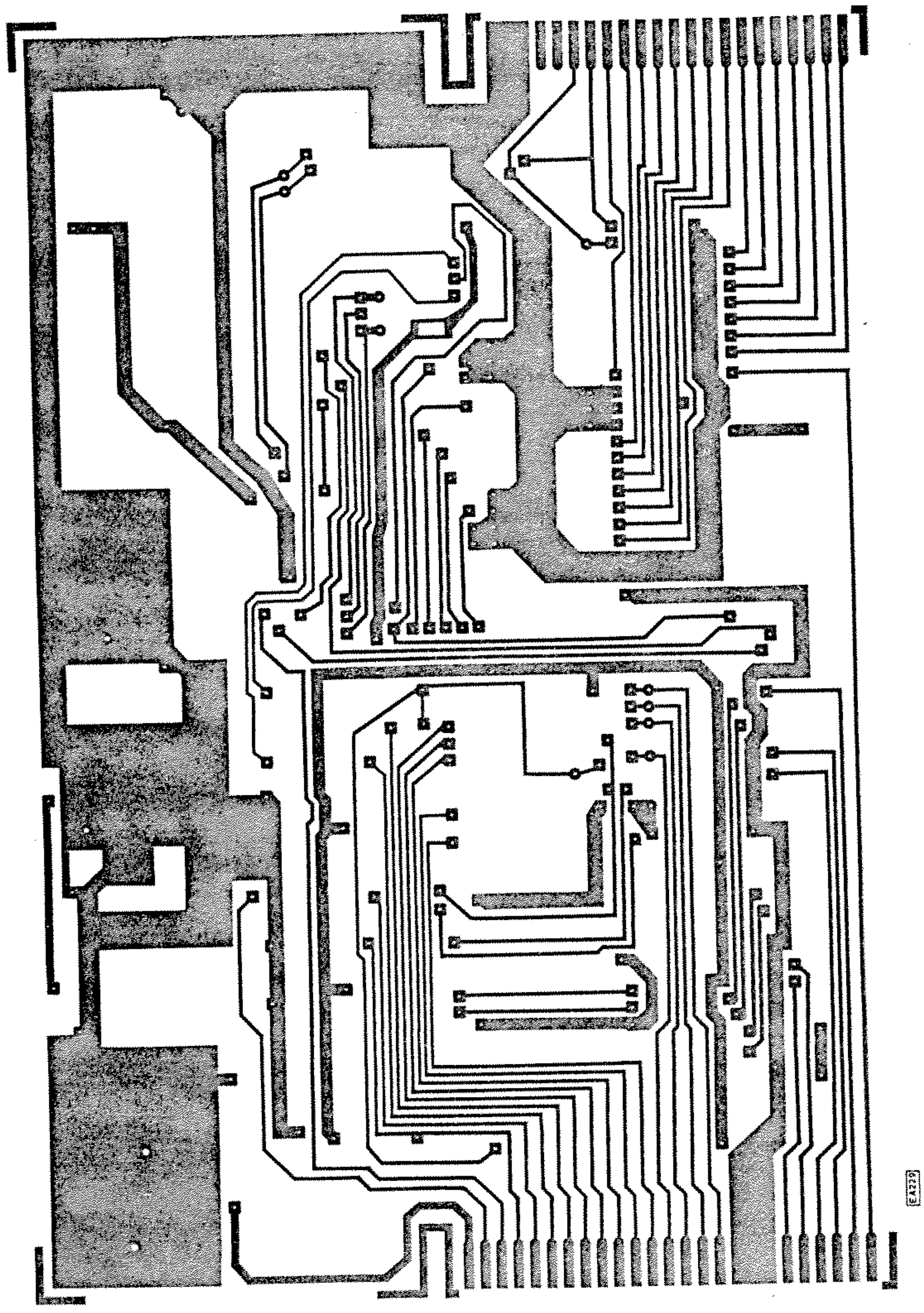


Fig. 2.3. Component-side p.c.b. layout (actual size)

CONSTRUCTION

Construction of the Decoding Module should pose no real difficulties to those experienced with soldering. It might be useful to put in all i.c. holders and other components before tackling the through-holes. The positions of these are indicated in the component overlay drawing, and may be identified in almost all cases as holes surrounded by a square rather than a round track on both sides of the board. All through-pins should be separately soldered on both sides of the board, and a check made that *both* joints are in tact at the end of the operation. The voltage regulator, IC10, should be mounted on the heat-sink with a 6BA bolt or similar.

Note that if it is intended to tap Compukit's PSU transformer to supply the Module, the connection should be made *directly* to the transformer tags, and *not* to the three pads on Compukit's board. This latter precaution will reduce mains hum due to the earth loop, keeping it within reasonably acceptable limits.

TESTING

After checking with an ohm-meter that the PSU lines are not shorted to earth, connect the Module to the 9–0–9 V supply with the fuse and all i.c.s except IC10 removed. If this produces 5 volts, the remaining i.c.s may be inserted, taking particular care with IC9, since this device may be damaged by high voltages. With all i.c.s. inserted, the Module should draw about 100mA (measured in series with the 1A fuse).

Next, connect the Module to Compukit's expansion socket. This should not cause any significant change in current consumption. If Compukit "locks up" at this point and refuses to Reset, then there is probably a short in the Ø2, R/W, data or address lines, or the DD line has been brought permanently low. Check for these possibilities with an ohm-meter (with the Module disconnected from Compukit and from its power supply), or by disconnecting the leads to SK1 one at a time until the fault clears.

When all appears to be well, connect a high impedance voltmeter to pin 16 of SK3. This should read about 4·5 volts. Ensure that IC6 has been reset, and then execute POKE 61340, 255. This sets port A to output, and should cause the voltage on pin 16 to drop to about 0·2 volts. Now execute POKE 61341, 255 (to call up the peripheral register) followed by POKE 61340, 255 (to put 1s in the output register). This should cause the voltage on pin 16 to rise to about 4·5 volts, and should indicate that ICs 1, 2, and 9 are operational. If pin 16 refuses to go high on resetting the 6821, this indicates a fault not associated with the decoding. If it is high, but failed to go low on POKEing 61340, then the fault could either be in the connections to the 6821 or a failure in the decoding.

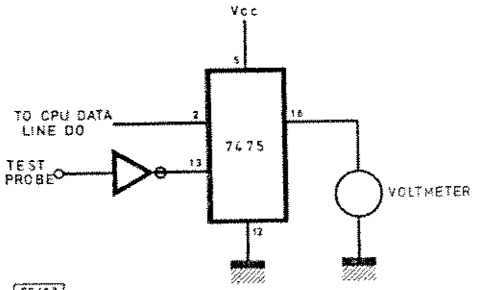


Fig. 2.5. 7475 Latch for testing the module

To check the decoding you will either need an extremely good oscilloscope, or a bistable latch. Fig. 2.5 shows a 74LS75 latch and inverter wired up for this purpose. The first point to check is the output of IC1. Connect this to the test probe, and POKE any address within the 128 byte block used (ie 61312–61439) with a 1 and then with a zero alternately. The voltmeter on the latch output should follow this, so verifying that IC1 is decoding the base address. The sixteen outputs of IC3 may also be tested with the probe. To test for the full operation of the Read circuitry a tristate buffer should be used (see below). Some indication as to the functioning of IC4 may be obtained by using the data latch of Fig. 2.5. This should read a '1' when connected to the outputs of IC4 when PEEK commands are executed at the appropriate address.

DIGITAL INPUT TO THE COMPUKIT TRISTATE BUFFERS

Digital data may be input to Compukit either directly using the Decoding Module's PIA, or indirectly through the use of tristate buffers. The function of the latter is simply to keep peripheral circuits isolated from the CPU data bus until the exact moment that data is required from it. Fig. 2.6 shows a tristate buffer buffering a single data line. With X high data will pass freely to the CPU, but when it goes low the buffer adopts a high impedance state. Eight such devices would be required to buffer a full 8–bit data bus, in which case the Enable lines X would all be connected to the same address-decoded line from the Decoding Module. A PEEK to the given address would then allow the CPU to read the data at the buffer inputs, and as soon as the Read was complete, the CPU would return the buffers to a high impedance state.

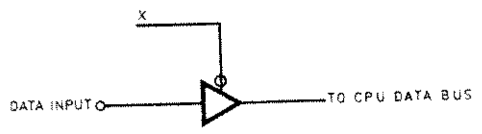


Fig. 2.6. Tristate buffer on single data line

The chief advantage of using this method over the PIA are its lower cost, and the fact that it does not require initialisation before use. This must of course be balanced against the extra complexity of wiring involved. There are also a number of more complex buffers on the market (eg the 74241) which are easier from the wiring point of view, but this is offset by their greater cost per bit.

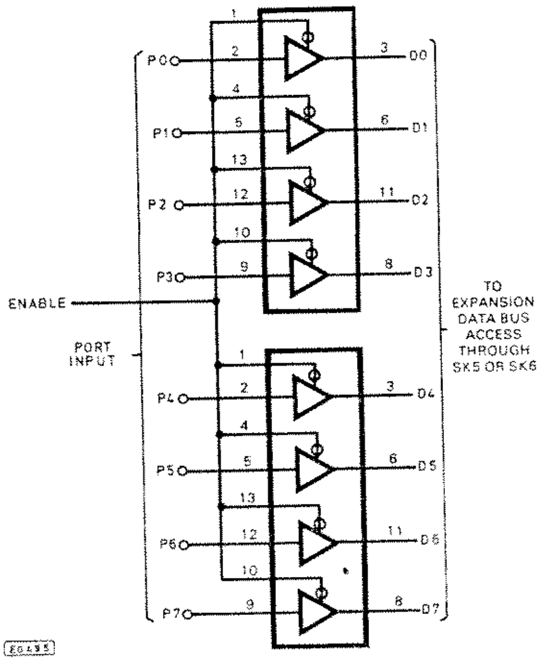


Fig. 2.7. 74LS125 8 bit port

0000
0001
0100

In the event that more input ports are required than the Decoding Module PIA provides, Fig. 2.7 gives the circuit of a pair of 74LS125s wired as an 8-bit input port that may be used directly with the Module. The Enable, being active-low, is directly compatible with the Decoding Module's non-inverted Read lines.

SWITCHES AND JOYSTICKS

The 6821 and PIA or a 74125 port can be easily used for the input of information from switches or push-buttons. Fig. 2.8 shows one way in which this may be achieved. The resistor value of 1k used here is appropriate for either port of the PIA or for a 74125 port.

Interfacing a games joystick for digital operation is also easily accomplished (the more complex interfacing of joysticks for analogue control will be dealt with in a later article). The two-axis joystick consists of two linear potentiometers of about 100k mounted at right angles, and can be conveniently interfaced in a similar manner to the switches. Fig. 2.9 gives the circuit for a control box containing 4 push-buttons and a joystick. The 1k resistors may be mounted inside the box, and a single 10-strand ribbon lead used to connect this to the Decoding Module.

A truth table for the four least significant bits of this circuit is shown in Table 2.3. As may be seen, there are nine possible configurations, including the four diagonal directions. The great advantage which the use of switches and joysticks confers over the polled keyboard for games and other uses is that it is possible to activate any number of switches etc. simultaneously without blocking the input. In the case of the joystick alone this simply means that "diagonal" instructions can automatically be accepted as well as "vertical" or "horizontal" ones. But it also implies that any combination of the four push-buttons may be simultaneously acted upon.

It is also a simple matter to extract the relevant information from the binary data at the port. The easiest way to do this is to use the extremely useful AND operator in Compukit's BASIC. If this operator is used on decimal numbers rather than on expressions, it converts those numbers to binary, and behaves like a set of eight 2-input AND gates. Thus the instruction PRINT 13 AND 25 will give the result 9.

Converting 13 and 25 to binary and ANDing them we can see why.

```
13  00001101
25  00011001
 9  00001001
```

The only bits which are 1 in both numbers are the first and the fourth, which gives the binary representation of 9. This function makes decoding of the joysticks and other digital data an extremely easy matter.

To test whether the joystick is in the "up" position one can simply PEEK the corresponding port and AND the result with the binary number 00000010. Clearly, the result will only be non-zero if the second bit of the joystick data is also non-zero; ie if it is in the "up" position. Similar operations can be performed for the other three bits, and since each operates completely independently of the others, the diagonal positions will automatically be catered for. Thus the following four program lines could be used to move a cursor in eight different directions across Compukit's screen. X and Y are the horizontal and vertical screen positions, and A the address of the port.

```
100  IF(PEEK(A) AND 1) >0 THEN Y=Y+1
110  IF(PEEK(A) AND 2) >0 THEN Y=Y−1
120  IF(PEEK(A) AND 4) >0 THEN X=X−1
130  IF(PEEK(A) AND 8) >0 THEN X=X+1
```

In order to demonstrate these principles in action, a simple program is given for screen writing, using the joystick and push-button circuit of Fig. 2.9. The joystick is used to move a flashing cursor to any point on the screen, and the four buttons have the following functions: 1 (at bit 4) Draw, 2 Erase, 3 Change character, 4 Clear screen. The program, which is listed in Table 2.4 is extremely short, but allows intricate graphics work to be executed on the screen.

The techniques used here could obviously be implemented in a number of different ways for games purposes, and one could easily add a second joystick to allow for two-person games. In next month's issue, when we will be discussing digital output techniques, we will give circuitry and

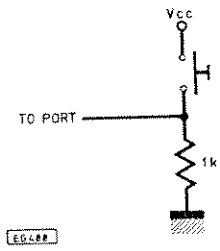


Fig. 2.8. (Left) push button inputs

Fig. 2.9. (Below) Joystick control box

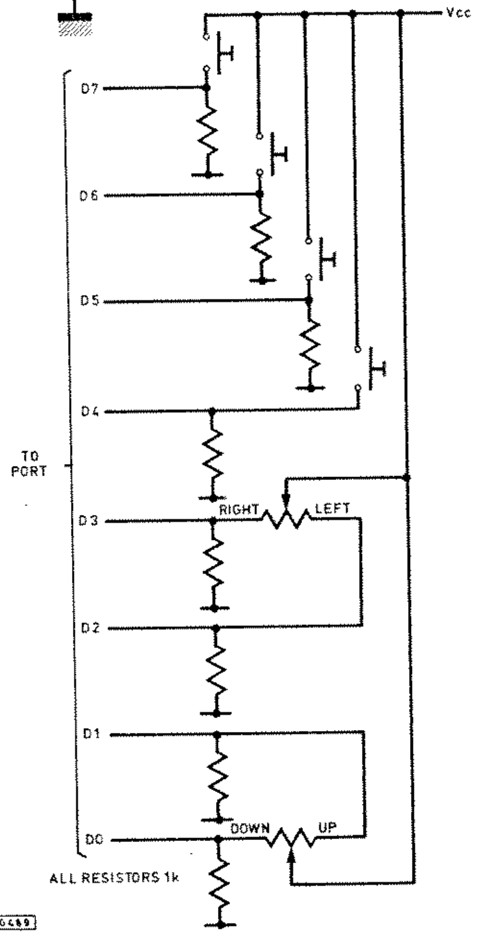


Table 2.3. Truth table for Joystick

Position of Joystick	Four Least Significant Bits P3	P2	P1	P0
Centre	0	0	0	0
Down	0	0	0	1
Up	0	0	1	0
Left	0	1	0	0
Right	1	0	0	0
Down and Left	0	1	0	1
Down and Right	1	0	1	0
Up and Left	0	1	1	0
Up and Right	1	0	1	0

soft-ware for adding a 7-segment l.e.d. display to indicate the position of the cursor in the above program. This is a useful complement and enables the program to be used to set up graphics work for transferral of both BASIC and 6502 code programs.

```
OK
LIST
 80 REM P.E. INTERFACING UK101 PROG 2
 90 REM JOYSTICK DRAWING ROUTINE
 95 REM (WITHOUT LED DISPLAY)
 100 FORI=OTO15:PRINT:NEXT
 110 V=53260
 120 X=23:Y=8
 125 VP=V+X+64*Y
 130 P=61340
 140 C=161
 150 POKEP+1,0:POKEP,0
 160 POKEP+1,255
 170 Q=PEEK(P)
 180 IF(QAND64)=OTHEN200
 184 C=C+1:IFC=191THENC=128
 186 POKEVP,C
 188 FORI=OTO300:NEXT
 200 IF(QAND1)>OANDY<15THENY=Y+1
 210 IF(QAND2)>OANDY>OTHENY=Y-1
 220 IF(QAND4)>OANDX>OTHENX=X-1
 230 IF(QAND8)>OANDX<47THENX=X+1
 240 IF(QAND128)>OTHEN100
 250 VP=V+X+64*Y
 260 C1=PEEK(VP)
 265 POKEVP,35
 267 FORT=OTO100:NEXT
 270 POKEVP,C
 280 IF(QAND16)>OTHEN170
 290 IF(QAND32)>OTHENPOKEVP,32:GOTO170
 300 POKEVP,C1
 310 GOTO170
OK
LOAD
```

```
OK
LIST
 30 REM P.E. INTERFACING UK101 PROG 3
 40 REM UK101 LOGIC TESTER
 50 REM NOTE - RETURN KEY GIVES SCREEN
 55 REM RECORD OF LOGIC STATES
 60 V=54125
 70 P=61340
 80 POKEP+1,0:POKEP,0
 90 POKEP+1,255
 100 FORI=OTO15:PRINT:NEXT
 110 PRINTTAB(13);"UK101 LOGIC TESTER"
 115 PRINT:PRINT
 120 PRINTTAB(11);"D7 D6 D5 D4 D3 D2 D1 D0"
 130 PRINT:PRINT
 200 POKE530,0
 210 FORI=OTO7
 220 Q=PEEK(P)
 230 POKEV-3*I,((QAND(2↑I))/(2↑I))+48.1
 250 NEXT
 270 POKE530,1
 280 POKE57088,223
 290 IFPEEK(57088)=247THEN120
 300 GOTO200
OK
LOAD
```

Table 2.4. Left
Table 2.5. Above

LOGIC TESTER

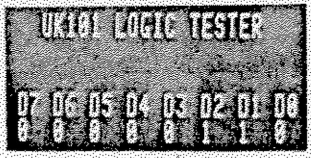
It is a very simple matter to use one or more of the ports developed with the Decoding Module for the purpose of testing logic state in digital circuitry. This simply involves connecting test clips to the eight lines of a particular port, and using these in conjunction with the appropriate software. Table 2.5 gives the listing of a program which displays the logic states of the eight lines of port A of the PIA. If 74125s are to be used in place of the PIA, then line 70 should be changed so as to set P to the appropriate address, and line 80 and 90 deleted. Line 230 uses the AND operator to determine the value of a given input line, and it performs this once for each of the eight data lines, using the FOR loop at program line 210. The 48·1 at the end of line 230 should really be 48·0. Its function is to convert a zero or 1 into the ASCII code for those characters, and the number 48 will normally achieve this; but since Compukit thinks that 2↑16 is 65536·2 (rather than 65536·0), 48·1 must be used so as to convert correctly.

The photographs show Compukit's screen during the running of the program. Pressing Return at any time causes the current logic states to be printed as a record higher up the screen as may be seen. Note that since the program is written in BASIC, the response time is relatively slow.

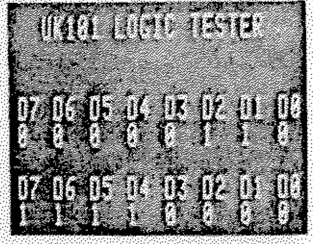
OTHER INPUTS

There is of course no limit to the variety of devices that can be interfaced using the 6821 PIA or a 74125 port. In fact any device capable of producing a transition between about 0·5 and 3·5 volts (less for port A of the PIA) can be directly connected to them. We shall briefly look at the implementation of light and sound detectors, although here, as in many other cases, far greater information can be derived from such sources using analogue to digital conversion techniques, to be treated later in the series.

Light dependent resistors such as the ORP 12 are amongst the easiest light detectors to interface because their parameters change over many decades for relatively small changes in illumination. It is even possible to wire these directly to the port using a 1k resistor to ground as shown in Fig. 2.10. If it is required to detect lower levels of light than this circuit permits, a simple transistor amplifier may be used as in Fig. 2.11. With the high gain 2N2926G transistor this is capable of detecting very low levels of light. If desired, a potentiometer could be inserted in the base circuitry of the transistor so as to provide some degree of level control. The circuit of Fig. 2.11 differs from that of Fig. 2.10 in that it takes the port low when light is present. This effect can easily be reversed in software by subtracting the data read at the port from 255. This will cause each of the eight inputs of the port to be effectively active-low rather than active-high.


```
UK101 LOGIC TESTER

D7 D6 D5 D4 D3 D2 D1 D0
0  0  0  0  0  1  1  0
```


```
UK101 LOGIC TESTER

D7 D6 D5 D4 D3 D2 D1 D0
0  0  0  0  0  1  1  0

D7 D6 D5 D4 D3 D2 D1 D0
1  1  1  1  0  0  0  0
```

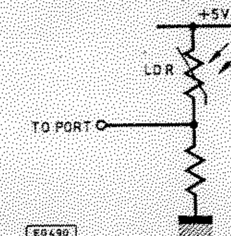


Fig. 2.10. (Above) LDR directly connected

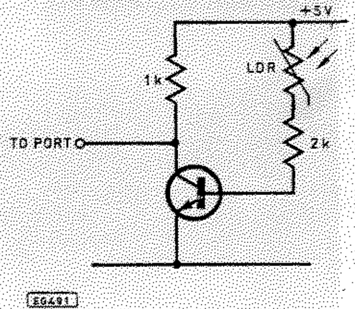
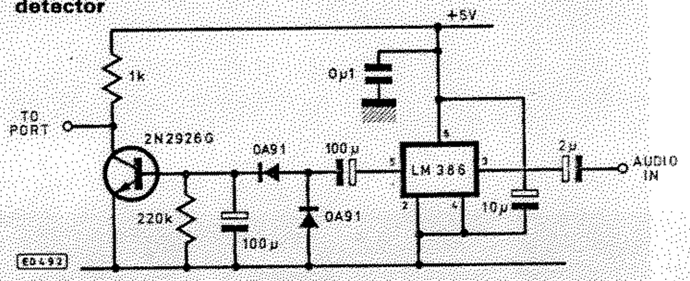
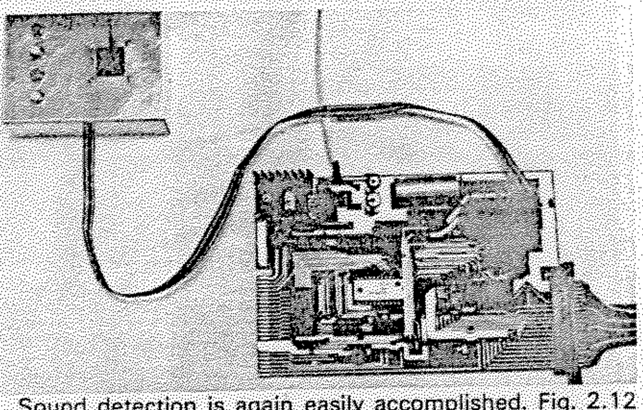


Fig. 2.11. (Right) High Sensitivity light detector

Fig. 2.12. (Below) Audio detector





Sound detection is again easily accomplished. Fig. 2.12 gives the circuit of a simple audio detector using an LM386 audio amplifer, which will run happily at 5 volts. The amplifier is sufficiently sensitive for the circuit to operate with a high output dynamic microphone, but if greater sensitivity is required, a preamplifier should be used. In setting up this and other digital interfaces it will be found that the Logic Tester program of Table 2.5 can be used to provide a convenient way of monitoring the state of the port in use.

Next month we will look at the use of the PIA and data latches to output digital data from the Compukit, and will discuss various applications including an IC tester for devices of the 7400 series.