Interfacing COMPUKIT

Part 1 D.E.Graham

THE COMPUKIT UK 101 is one of the few personal computers with 8K BASIC and full keyboard that does not have an input/output port for interfacing external devices. In this series we propose a remedy for this in the shape of an Address Decoding and Port Module which plugs directly into the Compukit's expansion socket. It is also Superboard II compatible.

The Module has been designed with flexibility in mind, and as well as housing an MC6821 Parallel Interface Adaptor (PIA), which gives two 8-bit input/output ports, the board also provides 7 uncommitted address-decoded *read*, and 14 decoded *write* lines, each of which may be used with interfaces of the reader's choice, and a pair of specially decoded lines that will directly interface an AY-3-8910 or 8912 PSG. In addition there is on-board address decoding for a further 6 blocks of 16 memory locations; again these are completely uncommitted, and each could be used to enable devices with up to 16 independent registers, such as the 6522 Versatile Interface Adaptor, details of which will be given later in the series. The board also houses an independent 5 volt regulated power supply which may be used to run a limited number of external circuits.

During the series the principles of interfacing the Compukit using various devices will be developed, and circuits will be given for a range of interfaces that may be plugged directly into the Decoding Module. Amongst these will be featured interfaces for joysticks, l.d.r. light sensors, 7-segment l.e.d. displays, audio generators, power controllers, and D/A and A/D converters. Software support for each will also be discussed.

The first part of the series is devoted to the Decoding Module itself.

DECODING PRINCIPLES

The 16 address lines of the Compukit can be confugured in 2^{16} or 65,536 different ways, or in other words it can address 65,536 different memory locations. To pick out just one of these, gating circuitry must be used. The circuit in Fig. 1.1, employing a single 16-input AND gate, would give a high output if, *and only if*, each of the address lines was simultaneously high. The Compukit's address lines are active-high, so that the circuit could be used to provide a Chip Select signal when the address FFFF hex (or 65,536) was put on the address bus by the Compukit's CPU. Different addresses could be decoded by simply placing inverters between chosen address lines and the gate inputs. Putting an inverter in lines A0 and A4, for example, would decode for the address FFEE hex (65,519 decimal). In Table 1.1 we give a listing of a hex to decimal/decimal to hex converter that may prove useful for calculating addresses on the Compukit.

COMPONENTS . . .
DECODING MODULE

Resistors

R1, R2	1k ¼W (2 off)
R3	220 1W
R4–R6	10k ¼W (3 off)

Capacitors

C1	2200µ 15V
C2, C5	100n disc cer. (2 off)
C3	10µ 15V
C4	100µ 15V
C6	1000µ 15V
C7	10µ 15V
C8–C13	0·1µF disc

Diodes

D1–D4	1N4001 (4 off)
D5	5V 1·2W Zener

Integrated Circuits

IC1	74LS133
IC2, IC4	74LS138 (2 off)
IC3	74LS154
IC5–IC7	74LS04 (3 off)
IC8	74LS03
IC9	6821
IC10	7805

Miscellaneous

2 off	40-pin d.i.l. socket
5 off	16-pin d.i.l. socket
2 off	24-pin d.i.l. socket
4 off	14-pin d.i.l. socket
1 off	40-pin d.i.l. plug
~~2 off~~	2× 22-pin 0·1in. edge connector
~~1 off~~	2× 25-pin 0·1in. edge connector
1 off	24-pin d.i.l. plug
2 off	16-pin d.i.l. plug
2 off	8T28 to plug into Compukit

Ribbon cable
1A 20mm slo-blo fuse and p.c.b. holder
push-to-make switch.

The ENABLE line of Fig. 1.1 could be used to trigger a data latch (such as the 7475) to latch data appearing instantaneously on the CPU's data bus for use by some external device. In practice, in order to ensure that the ENABLE pulse comes at exactly the right instant, it is desirable to make it conditional on Compukit's Ø2 clock line going high. Fig. 1.2 shows a circuit using a 17-input AND gate that would decode for the address EF18 hex (61028 decimal). This is a

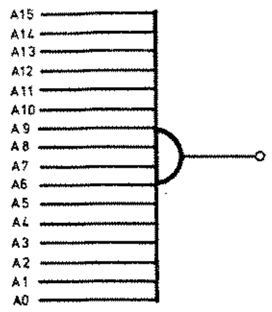


Fig. 1.1. Above. 16 input AND gate

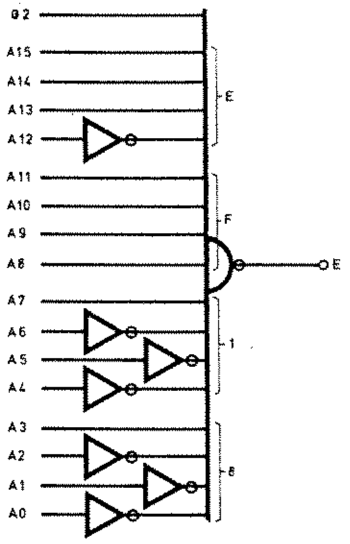
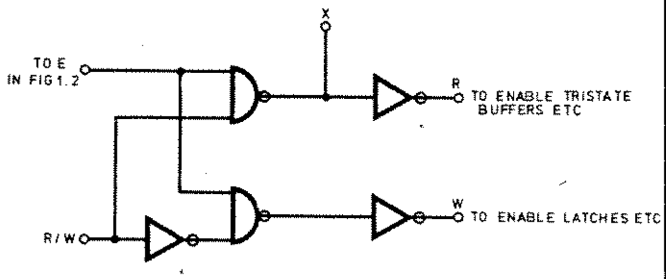


Fig. 1.2. Right. Decoding for EF 18

Fig. 1.3. Below. Separation of R/W decoding



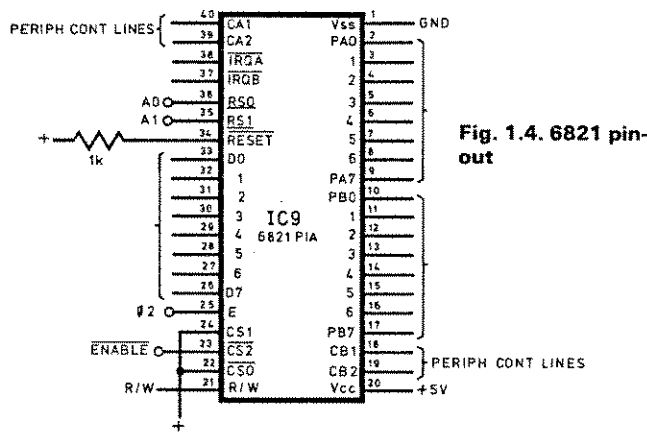


Fig. 1.4. 6821 pin-out

```
90 REM HEX-DEC-HEX CONVERTER
95 REM PE UK101 INTERFACING PROG NO 1
100 FORA=1TO16:PRINT:NEXT
110 PRINT,"HEX-DEC-HEX CONVERTER"
115 PRINT:PRINT:PRINT:PRINT
120 PRINT"    IS DATA HEX OR DECIMAL ?"
125 INPUT"    ENTER H OR D";Y$
130 IFY$="D"THENGOSUB550:GOTO165
140 IFY$="H"THENGOSUB550:GOTO350
150 PRINT:PRINT"    NOT RECOGNISED: ENTER AGAIN"
160 GOTO120
162 REM
163 REM DEC TO HEX ROUTINE
164 REM
165 PRINT:PRINT:PRINT
166 INPUT"    DECIMAL DATA PLEASE";N
168 IFN=0THEN350
170 A=INT(N/4096)
180 A1=A*4096
190 B=INT((N-A1)/256)
200 B1=B*256
210 C=INT((N-A1-B1)/16)
220 C1=C*16
230 D=N-A1-B1-C1
240 X$="0123456789ABCDEF"
250 PRINT,"HEX EQUIVALENT=  ";
260 PRINTMID$(X$,A+1,1);
270 PRINTMID$(X$,B+1,1);
280 PRINTMID$(X$,C+1,1);
290 PRINTMID$(X$,D+1,1)
300 GOTO165
350 REM
360 REM HEX TO DEC ROUTINE
370 REM
390 PRINT:PRINT:PRINT
400 INPUT"    HEX DATA PLEASE";H$
402 IFH$="0"THEN165
403 IFLEN(H$)<>4THENPRINT:PRINT"    4 DIGIT FORMAT ONLY":GOTO400
405 N=0
410 X$="0123456789ABCDEF"
420 FORJ=1TO4
430 FORI=1TO16
440 IFMID$(H$,J,1)=MID$(X$,I,1)THEN460
450 NEXTI
455 PRINT:PRINT"    CHARACTER NOT IDENTIFIED - RE DO"
456 GOTO390
460 N=N+(I-1)*16↑(4-J)
470 NEXTJ
480 PRINT,"DECIMAL EQUIVALENT=  ";N
490 GOTO390
500 END
550 PRINT:PRINT:PRINT"    NOTE THAT ENTERING A ZERO WHEN"
560 PRINT"    DATA IS REQUESTED REVERSES FUNCTION"
570 RETURN
```

Table 1.1 Hex/Dec. and D/H converter program

Table 1.2. Compukit's Memory Map showing gaps

Address	
0000-02FF	Scratchpad RAM for operating system
0300	Start of Basic Workspace
1FFF	End of On-board RAM
9FFF	End of Possible Ram expansion
A000-BFFF	Basic Interpreter
D000-D3FF	Video RAM
DF00	Polled keyboard
F000, ~~F001~~ F0FF	ACIA serial port
F800-FFFF	Monitor ROM

quite arbitrary address, and clearly any of the Compukit's 65,536 addresses could be decoded in this way; although of course since 17-input AND gates are not readily available, one would be forced to use a combination of gates to achieve the same effect in a practical circuit.

There are two further factors which must be considered in decoding for an interface, both of which relate to the R/W (Read/Write) signal. The circuit of Fig. 1.2 will give an output at any time that the address 61208 appears on the address bus. Thus, executing POKE 61208, X or Y=PEEK (61208), would both cause an output from the decoding circuit. But in most applications it is useful to distinguish between read and write operations. If, for example, we are using the signal to trigger a set of latches to give a data output, we will only want this to occur in response to a POKE command, whereas if it were used to turn on a tristate buffer for the input of data to the CPU, we would want this to occur exclusively in response to a PEEK statement.

Differentiation between the two can be achieved by using the R/W line at Compukit's expansion socket. This goes high during a Read Cycle, and low during a Write Cycle. The configuration in Fig. 1.3 would derive two separate Chip Select lines from the output of the circuit in Fig. 1.2, one for a Read to the address 61,208, and one for a Write. As may be seen, even though the two resulting decoded lines share the same address in the Compukit's memory map, they could be used for entirely different purposes. The Write might be used to trigger latches driving a D/A converter, while the Read might trigger tristate buffers to feed the CPU with the counting registers of an external clock, for example.

Finally, in our decoding circuitry we must include a means of controlling the DD or Data Direction line of the Compukit. This determines the direction in which data is allowed to pass through the two 8T28 data buffers on the Compukit's main board. Note, incidentally, that while these two i.c.s are essential in any use of the data bus at the expansion socket, they are not provided in the basic UK101 kit, and must be purchased separately. With the DD line high, data can pass from the CPU to the expansion socket, but not in the reverse direction. When it is low, on the other hand, the converse is true. With no external signal on this line, it is kept high by Compukit's on-board resistor network R9 R74. If we did not service the DD pin at the expansion socket, we could successfully write data to external devices with the circuit of Figs. 1.2 and 1.3, but even though the R line of Fig. 1.3 would go high when a PEEK(61208) was executed, no data from the tristate buffers, or whatever else was enabled, would actually get to the CPU data bus. This could be remedied by connecting point X in Fig. 1.3 directly to the DD pin of the expansion socket. This would bring DD low only when a Read instruction was carried out at the given address, and the associated interfaces could then be both written to, and read from, in a satisfactory manner.

THE DECODING MODULE

The Decoding Module requires a 128 byte address block, a requirement easily met within the Compukit's memory map. This is reproduced in table 1.2, and it may be seen that the Compukit possesses unused blocks at C000-CFFF, D400-DEFF, DF01-EFFF and F100-F7FF hex. For reasons of simplicity we have chosen to locate the module between EF80 and EFFF hex (61,312-61,439 decimal). This falls immediately below the serial port at F000 hex. An address map of the major 8 blocks of the module is given in Table 1.3.

Table 1.3. Address Map of Module

Base Address of (Hex)	Block (Dec)	Block Number	Function
EF80	61312	BL0	Base address for 8 decoded lines
EF90	61328	BL1	Base address for PIA block
EFA0	61344	BL2	Free Block
EFB0	61360	BL3	Free Block
EFC0	61376	BL4	Free Block
EFD0	61392	BL5	Free Block
EFE0	61408	BL6	Free Block
EFF0	61424	BL7	Free Block

The board uses a combination of edge connectors and d.i.l. sockets for external connections, and the pin-outs of these are given in Tables 1.4–1.8. Edge connector SK1 carries the 40 leads from the Compukit's expansion socket, and the wiring between these should be kept as short as possible. The 40-pin socket SK2 allows for further expansion of the Compukit, and has the same pin-out as Compukit's own expansion socket. The two 16-pin d.i.l. sockets SK3 and SK4 carry ports A and B of the PIA, respectively, together with associated control and power supply lines.

The decoded lines produced by the Decoding Module are taken out through the 24-pin d.i.l. socket SK5, carrying six Write and two Read lines, and the 2 × 25 pin edge connector SK6 which carries the remainder. Both SK5 and 6 also carry Vcc and the data bus, and in addition SK6 carries address lines A0–A3, Φ2, $\overline{NMI}$, $\overline{IRQ}$ and $\overline{RESET}$ to allow full use of the six 16-byte blocks.

Table 1.4. Connections to edge connector SK1.

	UPPER ROW		LOWER ROW	
SK1 pin	Function	Connection to compukit exp. soc.	Function	Connection to compukit expansion
1	A2	12	GND	40
2	A1	13	GND	39
3	A0	14	GND	38
4	A3	15	GND	37
5	A4	16	n/c	—
6	A5	17	n/c	—
7	A6	18	R/W	32
8	$\overline{IRQ}$	1	O2	31
9	$\overline{NMI}$	2	A15	27
10	DD	3	A14	26
11	D0	4	A13	25
12	D1	5	A12	24
13	D2	6	A11	23
14	D3	7	A10	22
15	Spare	11	A9	21
16	A8	20	GND	30
17	A7	19	GND	29
18	n/c	—	GND	28
19	n/c	—	D7	33
20	GND	8	D6	34
21	GND	9	D5	35
22	GND	10	D4	36

Table 1.6. SK3 and 4 of PIA.

1	GND	16	AD0
2	CA1	15	AD1
3	CA2	14	AD2
4		13	AD3
5	GND	12	AD4
6		11	AD5
7		10	AD6
8	Vcc	9	AD7

SK4—PORT B of PIA is identical

Next month we will deal with the circuit operation of the Decode Module, showing the printed circuit board layout and component overlay. We shall also cover the operation of the PIA, and the construction and testing of the Decoding Module; and will look at the inputting of data to the COMPUKIT, both via the PIA, and sets of tristate buffers.

Table 1.7. Connections to SK5.

GND	1	24	$\overline{\text{W10}}$
GND	2	23	D6
GND	3	22	D4
D7	4	21	D1
D5	5	20	D3
DO	6	19	W12
D2	7	18	W14
$\overline{\text{W11}}$	8	17	GND
W13	9	16	GND
W15	10	15	GND
Vcc	11	14	$\overline{\text{R5}}$
Vcc	12	13	$\overline{\text{R4}}$

Table 1.8. Connections to SK6 edge connector.

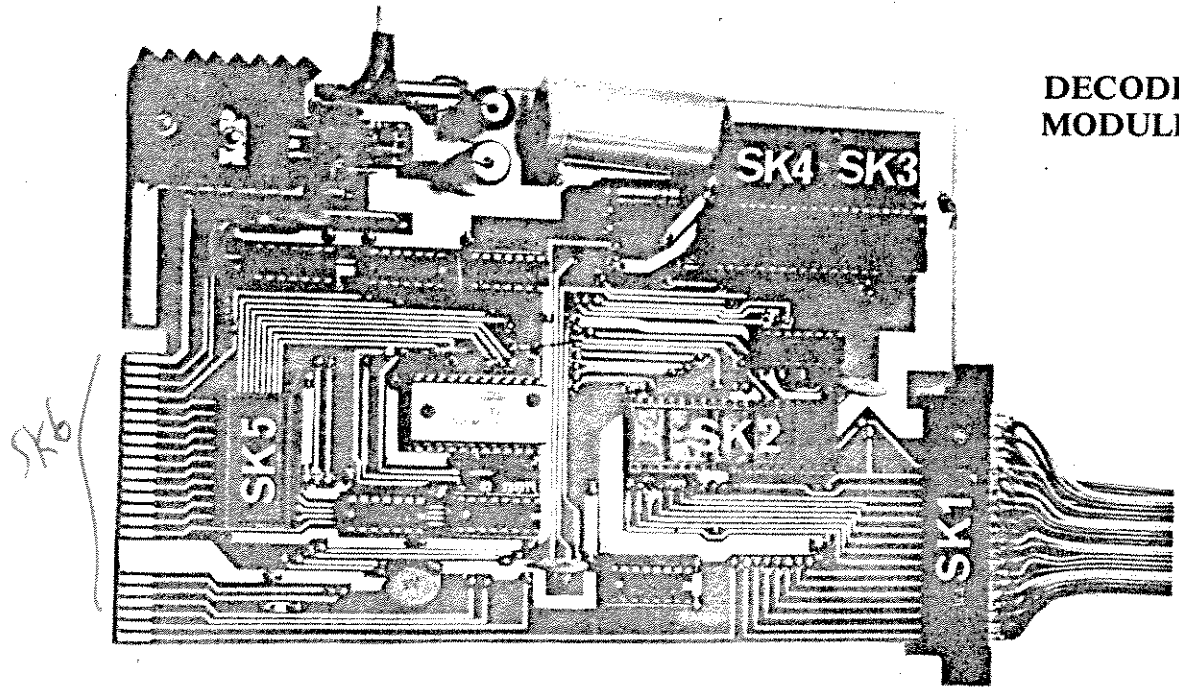
	upper (Component side)	lower
1	Vgg –5V	$\overline{\text{RESET}}$
2	Ø2	W7
3	$\overline{\text{IRQ}}$	W8
4	BC1	R7
5	BDIR	RO
6	W1	R1
7	WO	R/W
8	$\overline{\text{W2}}$	GND
9	$\overline{\text{W3}}$	GND
10	$\overline{\text{W4}}$	D7
11	$\overline{\text{W7}}$	D6
12	W9	D5
13	A3	D4
14	A2	DO
15	A1	D1
16	AO	D2
17	GND	D3
18	GND	Vcc
19	GND	Vcc
20	~~BL4~~ BL3	GND
21	~~BL3~~ BL4	GND
22	$\overline{\text{R3}}$	GND
23	R2	$\overline{\text{BL6}}$
24	$\overline{\text{BL7}}$	$\overline{\text{BL5}}$
25	$\overline{\text{NMI}}$	$\overline{\text{BL2}}$

Table 1.9. Address within Block 1. * Note that all but the three lines with an asterisk are uncommitted, and may be used with interfaces of the reader's choice, but that, as may be seen, a number of others have been earmarked for projects within the series.

Address	Write		Read	
61327	W15	To SK5 for 4 digit 7-segment display	—	
61326	W14		—	
61325	W13		—	
61324	W12		—	
61323	W11	To SK5	—	
61322	W10		—	
61321	W9	To SK6	—	
61320	W8	D/A converter	—	
61319	W7	A/D converter	R7 A/D converter	
61318	W6*	Audio (data)	R6* Audio (data)	
61317	W5*	Audio (address)	R5 to SK6	
61316	W4	To SK6	R4	
61315	W3		R3	
61314	W2		R2	To SK6
61313	W1	inverted	R1 inverted	
61312	WO	inverted	RO inverted	

Table 1.10. Selection of Base Address of Decoding Module. * "O" indicates inverter in use.

State of A11–13* A13	A12	A11	Address hex of 128 byte block	Comments
1	1	1	FF80–FFFF	These two already used by monitor.
1	1	0	FE80–FEFF	
1	0	1	EF80–EFFF	If pads are left untouched, the module assumes this slot.
1	0	0	E780–E7FF	6 possible relocation sites for Module
0	1	1	DF80–DFFF	
0	1	0	DE80–DEFF	
0	0	1	CF80–CFFF	
0	0	0	CE80–CEFF	



DECODE MODULE