VIDEO CONVERSION

By D. P. GOULDER

THE UK101 displays 48 characters per line on 16 lines. This conversion allows the user to select an optional 48 × 32 format. The characters become much more legible, and the doubling of the vertical resolution on display is a useful improvement when plotting graphs or drawing diagrams. Program listings provide double the information per page.

The VDU RAM is increased from 1K to 2K bytes, and is controlled by a modified version of the *new* monitor PROM, so that the line edit facility is available.

In order to check that the display being used has enough resolution for 32 lines, take IC60 pin 12 to +5V momentarily: two identical 16 line pages are displayed.

This conversion includes a means of selecting the three different monitor PROMS, (old 16 line, new 16 line, and new 32 line). Whilst Reset is held, this switch may be operated without losing programs.

One p.c.b. would be required; 8 new i.c.'s and the new PROM are required. Power can be obtained from the existing PSU.

HARDWARE

The current machine has 1K VDU RAM from D000 to D3FF. The hardware counting chain scans this onto the screen. C7 is not used, so each horizontal line is repeated (see p7, Fig. 2 in the manual).

The converted machine has 2K VDU RAM, D000 to D7FF. C7 is now used, and C14 is used to select between the two half pages, D000–D3FF and D400–D7FF. All the other counters are used as before.

A three way switch is provided to select between the three monitors. Six t.t.l. i.c.'s, two 2114's (for the extra RAM) and a new 2716 EPROM are required. If Reset is held while switching monitors, followed by warm start, programs are not lost. Thus program development can be carried out with 32 lines while setting up display graphics for 16 lines eg. for games.

IC102 and IC103 switch the counter outputs from IC60, 61 and 30 so that C7 is brought into use for 32 lines. They are activated by the three way switch. A change is needed to the $\overline{VA}$ signal generated by IC56 (currently active low when D000–D3FF selected, to enable the CPU to access the VDU RAM). IC56/2 is taken to +5V, instead of A10. $\overline{VA}$ becomes $\overline{VA}'$ active for D000–D7FF, $\overline{VA}'$ is used to activate IC105, which takes the place of the switch formed by pins 9, 10, 11 of IC55. IC104 is used to decode A10 and $\overline{\phi 2}$ into the memory select signals $\overline{M1S}$, $\overline{M2S}$, for the two page "halves".

The $\overline{RVE}$ and $\overline{WVE}$ signals also have to be changed to $\overline{RVE}'$, $\overline{WVE}'$ to allow 2K VDU RAM. IC106 provides this decoding replacing part of IC20's function.

IC101 provides six inverters required in various places in the circuit. IC107–110 are the 4 VDU RAM chips (two of which are already on the main p.c.b.: IC39, 40).

Besides the connections for data and address lines to the VDU RAM and the ROM'S, some 25 connections have to be made to the main p.c.b. along with a number of track cuts. (The prototype board is "piggy-backed" on two pillars on the main p.c.b. and the connections made with ribbon cable). The track cuts are needed to disconnect the lines from IC60, 61, 30 to

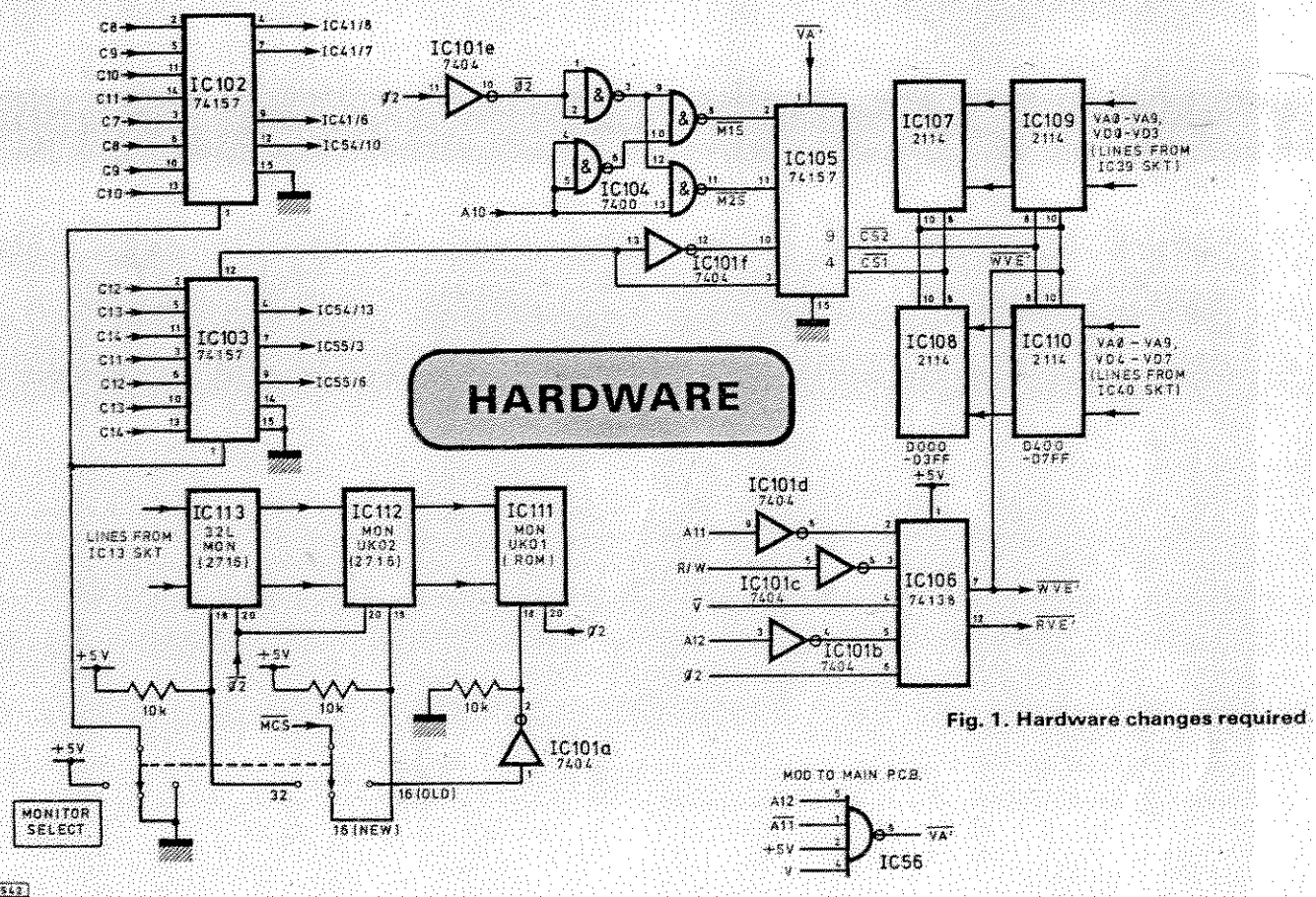


Fig. 1. Hardware changes required

IC41, 54, 55, and to disconnect A10 from IC56, and WVE, $\overline{WVE}$, $\overline{RVE}$.

SOFTWARE

The program in the new monitor requires alteration to:
Screen print routine — to allow 32 lines
Clear screen routine — to allow 32 lines (and optionally remove the scroll when the top of the screen is blank)
More Display
Up one line routine — to allow 32 lines
Form Cursor Address routine — to allow 2K VDU RAM
A total of 36 bytes require to be altered.

The firmware changes required are detailed below. The new program should be put into a 2716 (2K) EPROM, using the same address as the existing monitors (F8ØØ–FFFF).

The monitor program is completely unchanged except for the 36 bytes detailed.

Additionally, the byte at FB5B (currently FF) can be changed to ØØ if desired. This removes the rather irritating scroll up on the screen every time the top line becomes blank for any reason.

POWER SUPPLY

No problems have been found driving the new board from the existing PSU, although the regulator is already mounted on a large heat sink outside the case. A simple 5V, 1A PSU using a 7805 3-terminal regulator could however be constructed if required. All the i.c.'s should be LS types.

POWER

+5V IC101/14, IC102/16, IC103/16, IC104/14
IC105/16, IC106/16, IC107–IC110/18, IC11–IC113/24
0V IC101/7, IC102/8, IC103/8, IC104/7, IC105/8
IC106/8, IC107–IC110/9, IC111–IC113/12

CONNECTIONS TO NEW BOARD

IC60/12 (C7)
IC60/11 (C8)
IC61/14 (C9)
IC61/13 (C10)
IC61/12 (C11)
IC61/11 (C12)
IC30/14 (C13)
IC30/13 (C14)
IC56/6 ($\overline{VA'}$)
IC18/5 ($\overline{V}$)
IC8/22 (A12)
IC8/20 (A11)
IC8/19 (A10)
IC8/39 (Ø2)
IC8/34 (R/W)

—ALL INPUTS TO NEW p.c.b.

IC41/8
IC41/7
IC41/6
IC54/10
IC54/13
IC55/3
IC55/6
$\overline{RVE'}$—CONNECT IN PLACE
$\overline{WVE'}$—OF EXISTING $\overline{RVE}$, $\overline{WVE}$

—OUTPUTS FROM NEW p.c.b.

(PLUS 5V, ZERO VOLTS)

VA0–VA9, VD0–VD7 (FROM IC39, 40 SOCKETS)

—BUS CONNECTIONS

ALL CONNECTIONS FROM MONITOR (IC13) SOCKET, EXCEPT PINS 18, 20

IC19/6 ($\overline{MCS}$) FOR MONITOR SELECT SWITCH

Fig. 2. Firmware changes required

CIRCUIT

The sockets for IC39, 40 were used to obtain the VDU data and address lines. Similarly, the socket for IC13, to obtain connections for the monitor ROM and PROMS.

The pin connections for +5V and 0V to all the i.c.'s are also listed below.

Ribbon cable forms a suitable flexible means of interconnection between the boards.

GENERAL

The prototype is made up on a piece of veroboard, about 205 × 102mm, with the i.c.'s all on d.i.l. sockets. Quite a bit of wiring is involved, so careful checking of all connections is essential.

A few 100n ceramic capacitors should be connected between +5V and 0V to aid decoupling.

SOFTWARE CHANGES (TO NEW MONITOR)

ADDRESS (ABSOLUTE)	ADDRESS (PROM)	CHANGE BYTE FROM	TO (HEX)	ROUTINE
FACA	2CA	1Ø	2Ø	SCREEN PRINT
FAF1	2F1	1Ø	2Ø	
FB2D	32D	D4	D8	CLEAR SCREEN
FB61	361	D4	D8	MOVE DISPLAY
FB85	385	D3	D7	UP ONE LINE

NEW ROUTINE

Delete routine from FB8D (38D) to FBAB (3AB)
(This routine forms the cursor address and stores it in ØØE3, ØØE4)
REPLACE WITH:

FB8D	38D	A9	1A	LDA	#$1A
	38F	85	E4	STA	$E4
	391	AD	Ø8 Ø2	LDA	$Ø2Ø8
	394	ØA		ASL	A
	395	ØA		ASL	A
	396	ØA		ASL	A
	397	ØA		ASL	A
	398	26	E4	ROL	$E4
	39A	ØA		ASL	A
	39B	26	E4	ROL	$E4
	39D	ØA		ASL	A
	39E	26	E4	ROL	$E4
	3AØ	6D	Ø7 Ø2	ADC	$Ø2Ø7
	3A3	69	ØD	ADC	#$ØD
	3A5	85	E3	STA	$E3
	3A7	6Ø		RTS	
	3A8	EA		NOP	
	3A9	EA		NOP	
	3AA	EA		NOP	
FBAB	3AB	EA		NOP	

TRACK CUTS (ORIGINAL p.c.b.)

DISCONNECT CONNECTIONS BETWEEN:

IC60/11 (C8)	AND	IC41/8
IC61/14 (C9)	„	IC41/7
IC61/13 (C10)	„	IC41/6
IC61/12 (C11)	„	IC54/10
IC61/11 (C12)	„	IC54/13
IC30/14 (C13)	„	IC55/3
IC30/13 (C14)	„	IC55/6
IC56/2	„	$\overline{A10}$
IC20/12	„	$\overline{RVE}$
IC20/7	„	$\overline{WVE}$

MODIFICATION ON ORIGINAL p.c.b.

CONNECT IC56/2 TO +5V

SOFTWARE

MICRO PROMPT

The hardware and software exchange point for PE computer projects

EXPANDING GROUP

We have received the first newsletter proper from the UK101 User Group, which is accumulating members in the British Isles and overseas.

The group is doing some important work now, such as investigating the "sticking" FRE function, which Adrian Waters, the club organiser points out, is the tip of a serious ice berg concerning string data storage. A complex sound board is nearing completion, whilst behind the scenes the program library is swelling with games and educational software, and new languages such as PILOT.

The Newsletter, ROM, carries software news, hardware modifications, useful ROM and RAM locations, i.e. routine entry points, and a problem page. Equipment reviews are to become a regular feature.

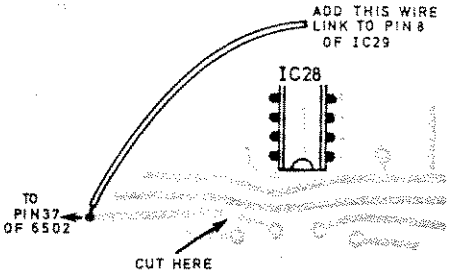
There is no entry fee, and the subscription for six months membership is £2·50, which should be made payable to Adrian Waters, at: 117 Haynes Rd., Hornchurch, Essex.

The following hardware modification was supplied by the 101 User Group, the details having originated from club member *Mr. R. Freeman.*

2MHz conversion

In the normal machine, the clock frequency of 1MHz is presented at pin 37 of the 6502 chip by the 8 output pin of IC29. Because the 6502 can accept a faster clock, many members have increased the speed of their machines by the following modification:

The 2MHz signal can be obtained from pin 8 of IC29 and applied to the 6502 by the Ø2 line at pin 37. The conversion can be implemented by cutting the track of the p.c.b. as shown in Fig. 1 and substituting the new link. Many members have included a changeover switch, although this cannot be used whilst the machine is working, without getting "hung up".



MODEM TO TANDY?

Sir—Firstly, thank you for an excellent magazine, it rivals any we have in the States.

Secondly, I picked up a copy of your February 1980 issue and am very interested in the Modem article by K. Amor. After much difficulty I also obtained part 2 in the March issue—expecting to learn how to connect this System to my Tandy TRS-80 16K Level II—only to be very disappointed. I would be very interested in any information you or your readers might have to offer. I would also be interested in exchanging hardware and software with any of your readers. Thank you.

From the desk of Bryan McPhee,
Capt., USAF,
2742 Virginia Trail,
Browns Mills,
N.J. 08015, U.S.A.

SHIFTY CHARACTERS

Sir—The following useful feature of the UK101 is not mentioned in your series of articles or the instruction manual:

With Shift Lock up, the keyboard returns lower case letters. To input a few upper case letters or figures, press L.H. Shift and the key for the character required (ie. L.H. Shift cancels the effect of Shift Lock being up).

To obtain the normally shifted characters (eg. ") then press R.H. Shift and the appropriate key.

Please continue to publish the very useful articles in Micro Prompt on this excellent machine.

K. W. Lambert,
Halesowen,
W. Midlands.

Sorry, we thought people knew—Ed.

GET KEY FOR UK101

To get a key from the keyboard without stopping the program, as in the input statement, run the following program, then each time you require a "Get Key" statement write:

(Line number) A$ = "Space" : POKE 11, 34 : POKE 12, 2 : X = USR(X)

After this line A$ will be a space unless a key was pressed, in which case A$ will be equal the character of the key which was pressed.

Any $ variable can be used, including array's. This can be used to replace the 'GET A$' statement as used on the PET. To set up the subroutine:

```
10 FOR A = 546 to 597
20 READ B : POKE A, B : NEXT
30 DATA 169, 2, 32, 190, 252, 32
40 DATA 198, 252, 208, 7, 10, 208
50 DATA 245, 169, 32, 208, 28, 74
60 DATA 32, 200, 253, 152, 133, 252
70 DATA 10, 10, 10, 56, 229, 252
80 DATA 133, 252, 138, 74, 32, 200
90 DATA 253, 24, 152, 101, 252, 168
100 DATA 185, 207, 253, 160, 0, 41
110 DATA 127, 145, 105, 96, 0
```

Once run this program can be erased if required.

J. L. Brice, Ashford, Kent.

ERROR MESSAGE ERROR

Sir—I am a UK101 user who has, like many, been frustrated by the rather graphic error messages. The result of this has been the following short program to produce "standard" Microsoft BASIC error messages:

Enter monitor
type ·0222/29 (carriage return)
7A ,,
AC ,,
2D ,,
BF ,,
Reset and enter BASIC.
Type: **POKE 538, 34 : POKE 539, 2**

All error messages will then be standard. The program works by masking off the most significant bit of all characters printed. The BASIC stored messages all have the MSB set on the last character, and it is the omission of an instruction to clear this bit which caused the original error messages. Considering the complexity of the BASIC, such an omission in the error routine is understandable, but I hope this will be corrected.

New error messages	
Syntax error	SN error
Double dimension	DD error
Division by zero	/0 error
Undefined statement	US error
Undefined function	UF error
Bad subscript	BS error
Long string	LS error
Out of memory	OM error
Overflow	OV error
Continue error	CN error
String tempories	ST error
Type mismatch	TM error
Next without FOR	NF error
Function call error	FC error
Illegal direct	ID error
Out of string space	OS error
Out of data	OD error

D. J. Anderson,
London.

MAP READING

Sir—You may wish to pass on to your readers an error discovered in the memory map of the UK101, found whilst implementing an 6821 I/O port, at a dedicated address.

The ACIA which resides at F000—F001 is due to page select decoding repeated at a further 127 locations through Hex page F0. The memory map should thus be amended to show that ACIA resides from F000—F0FF.

Readers might also like to note that an unbuffered data bus, terminates in a patch pad with 0.1 inch pitch spacing, to the left of the AC1A chip i.c. 14. This can only be used with selectable tristate logic, but as most 6502 compatible support devices have this facility, the cost of the AT28's may be saved.

M. C. Mannering,
Walthamstow.

All this does, is point the jump to warm start (addresses 0000-0002) to a simple routine to put 45 and 31 into addresses 536 and 537, respectively and then jump to address A274 to warm-start.

This method of interrupting the warm-start can incorporate any routine (for example to re-activate the error message program of D. J. Anderson, see September PE). I hope this will be useful.

Martin Stiby,
Dunstable.

TRACE PROBLEM WITH CEGMON

Sir—Mr. Beckett's TRACE program works very well on the standard UK101, and promises to be very useful. With the CEGMON, however, there seems to be a problem, and I wonder if anyone can help.

When the TRACE program is loaded and X = USR(X) entered one of three things happens:

1. TRACE operates as intended. This happens rarely, maybe due to some random store contents not under control.

2. The keyboard locks up, and nothing further can be done.

3. TRACE operates only while the space bar is held down, and single stepping (a potentially useful feature) can be achieved by pressing and releasing the space bar.

Occasionally both 1 and 3 are available. Pressing any key makes the subject program run continuously with TRACE, and pressing the space bar stops it.

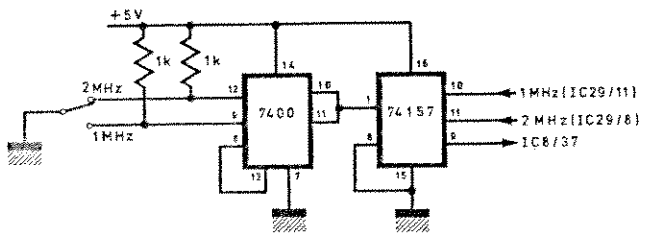
The trace program given with the CEGMON is inferior, since it destroys the printing format, and especially with graphics one cannot follow what is going on.

R. J. Newman,
Chesham,
Bucks.

DYNAMIC CHANGEOVER

Sir—I refer to the letter "2MHz Conversion" in Microprompt, Practical Electronics, September, 1980. You note that the 1MHz/2MHz changeover switch cannot be used while the machine is working. The circuit below allows this frequency change to be performed while running. It is not infallible, but if the machine does hang up, Reset followed by Warm Start will usually restore the program.

The 7400 is wired as a pair of cross-coupled NAND gates, to debounce the frequency change switch. The output is used to drive the 74157, feeding either 1 or 2MHz to pin 9, wired to Ø0 (pin 37 of IC8, 6502).



Break existing link between IC29/11 and IC8/37.

EG 6.3

With reference to the new monitor for the UK101, I have found that programs recorded at 600 baud often show corruption when being loaded (the 50th character of a line repeats the 49th) if the machine is running at 1MHz. The problem disappears if 2MHz is used. Has anyone else noticed this? The original monitor does not show this problem.

D. P. Goulder,
Comberton,
Cambs.

PRINTER INTERFACE?

Sir—I wish to interface a Centronics printer to my UK101 and wondered if this could be done using the PE decoding module. If so, how? I would appreciate details on how to do this if it is possible.

N. Odell,
Sheffield.

PS: The printer requires seven data bits and Strobe, Acknowledge and Busy lines are provided.

REPLY

Yes there is nothing that cannot be done with the Decoding Module!

You could drive your printer from Port B of the PIA, though you may need to buffer each data line with something like 7407 gates.

To get it to print, you will need to take the strobe line low while data is on the output of the port. One way would be to connect bits Ø-6 of the PIA directly to the 7 data lines of the printer (possibly via a buffer), and to connect bit 7 of the PIA to your strobe line.

Take the busy line to CB1, of the PIA and then all you need is some software. This will be a short program which picks up the SAVE vector and:

1) Configures the PIA for output on port 1
1a) Puts bits 7 high
2) Monitors the busy line (on CB1) until it is not busy
3) Outputs the character in the Accumulator to the port.
4) Takes bit 7 low, then high again to perform the strobe
5) Loops back to the UK101 output routine

This would be a fairly short machine code routine that could be located in the spare space at Ø23Ø hex.

D. E. Graham.

IMMEDIATE ERROR BANISHED

Sir—One minor irritation on the UK101 Superboard is the OM error message after the first command in immediate mode having just returned to BASIC from the monitor. This is especially annoying whenever a long instruction has been used, such as

```
FOR X = 7000 TO 8191 : PRINT
      CHR$(PEEK (X) ); : NEXT
```

Here is a patch of six bytes to avoid the OM error. The stack pointer is set to 255 (FF in hex) before the warm start.

```
0000  4C  FA      20JMP $02FA ;
                    TO THE PATCH
02FA  A2  FF        LDX # $FF
02FC  9A          A2TSX
02FD  4C  74        JMP $A274
```

The BASIC program pokes the patch into place.

```
5 REM TO AVOID OM ERROR MESSAGE
10 FOR X = 762 TO 767 : READ Z :
   POKE X,Z : NEXT
15 POKE 1,250 : POKE 2,2
20 DATA 162, 255, 154, 76, 116, 162
25 END
```

Robin H. Tracy,
Coventry.

STRING ANSWER

Sir—In response to R. J. Newman's query in the May 'Microprompt' the function STR$(X) is so designed that a leading blank (ASCII=32) will be inserted into the string from STR$(X) unless the number to be converted is negative—in which case the leading blank will be replaced by a minus sign (ASCII=45).

Eg.

```
100  X = -8 : X$ = STR$(X)
110  PRINT X$; ":" ;
120  PRINT ASC(X$) ; ASC (MID$(X$, 2,
     1))
RUN
-8 : 45 56
```

Christopher Davies,
Hove,
East Sussex.

TAPE VIEWER

Sir—Mr. Derry's letter (PE November 1980) was most welcome to one who has been making clumsy attempts to add auto-run to Basic programs. It would be useful to add that POKE 515,0 should be the first executable instruction in any auto-run program, so that other data on tape will not load and foul up the program.

His section on RAMless messages sent me to Edward H. Carlson's book "All About OSI BASIC in ROM" (Edward H. Carlson, 3872 Raleigh Dr, Okemos, Michigan 48864, USA) which inspired the following program for reading messages and programs without loading them into memory.

```
1 A=61440 : B=A+1
2 WAIT A,1 : ?CHR$ (PEEK(B)); :
GOTO 2
```

RUN the program, use CTRL/C to exit. (It may be necessary to stop the tape if unmodulated tone is playing through.) No error messages, no use of RAM (other than VDU RAM), and no waiting for ten minutes while EXMON loads! Any desired program line or message can be taken from the screen using one of the screen editors now available.

Mitch Park,
New Zealand.

MICRO PROMPT

The hardware and software exchange point for PE computer projects

SIMPLE SAVE BY NAME

Sir—This program enables you to name all your programs and load them back from tape by name. It will work on both the UK101 and Superboard II computers.

To use it just add two lines to your programs both Rem's the first contains the program name the second is just padding.

eg: 5 REM NAME# (Enclose the*
6 REM program name
*with * and #)*

After putting these two lines at the beginning of your program save in the usual way.

ie: Type SAVE (return) then LIST (return).

The program will then be stored on tape. When the computer comes back with OK stop the tape, type PRINT "POKE 515, 0" restart the tape and press return. (This is used when loading back the program to switch off the load flag to stop any other information being loaded.)

If you use the same line numbers in your programs as in SEARCH program, not only will it load but while loading the SEARCH program will be automatically deleted leaving the entire memory for use.

```
10 REM *SEARCH#
20 REM
30 FORX = 1TO24: PRINT: NEXT:
   INPUT "Programme name"; N$
40 PRINT "Press play on tape"
50 FORT = 1TO3000: NEXT:
   PRINT "Searching for"; N$
60 X$ = " ": F = 0: CA = 61440:
   POKE515, 1
70 WAIT CA, 1: P = PEEK (CA + 1)
80 IFF = OANDP <> 42THEN70
90 IFP = 42THENF = 1: GOTO70
100 IFP = 35THEN160
110 IFP= 32THEN70
120 IFLEN(X$) > 7THEN60
130 X$ = X$ + CHR$(P): IFX$ <>
    N$THEN70
140 POKE515, 0: PRINT "Loading";
    X$
150 GOTO180
160 POKE515, 0: PRINT "Found '";
    X$; "'"
170 GOTO60
180 LOAD
OK
RUN

Programme name? 3D
Press play on tape
Searching for 3D
Found 'SEARCH'
Found 'HEX'
Loading 3D
```

D. I. Swift,
Doncaster.

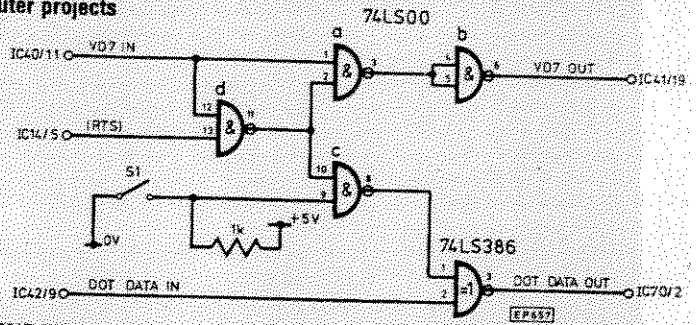


Fig. 1. Selective inverse video. S1 gives full field inverse, which could be made into another software option by using a second latch

SECRET SCREENWRITING

Sir—I wonder if your readers would be interested in a simple extension of the "secret" key polling sub-routine by J. M. Leach of Deal, Kent, for the UK101 published last March.

This program based on the New Monitor allows you to write direct to the screen with full control of the cursor, viz:

RUB OUT: Moves Cursor LEFT: Retyping corrects any mistakes

CTRL RUB OUT: Moves Cursor UP

SPACE: Moves Cursor RIGHT

CTRL=/—: Moves Cursor DOWN

RETURN: Moves Cursor to LEFT margin and DOWN one line ie. carriage return.

```
10   GOSUB 100
15   REM HAS RETURN BEEN
     ENTERED?
20   IF A = 13 THEN 60
25   REM HAS RUB OUT BEEN
     ENTERED?
30   IF A = 28 THEN 62
35   REM HAS CTRL RUB OUT
     BEEN ENTERED?
40   IF A = 220 THEN 64
45   REM HAS CTRL =/— BEEN
     ENTERED?
50   IF A = 237 THEN 66
55   ? CHR$(A); : GOTO 10
60   ? CHR$(10); : ? CHR$(13); :
     GOTO 10
62   ? CHR$(8); : GOTO 10
64   ? CHR$(11); : GOTO 10
66   ? CHR$(10); : POKE 10
100  POKE 11,0 : POKE 12,
     253 : X = USR(X) : A = PEEK
     (531) : RETURN
```

CTRL L Clears the screen as normal and returns the cursor to the home position (top left hand corner). This program allows messages to be displayed on the VDU and changed as required.

The alteration of line 55 to 55? A; : ? CHR$(A); : GOTO 10 allows the values of A to be printed out so that new IF A = "" statements can be devised.

Revd. P. R. Miller,
Milton Keynes.

AUTO-RUN

Sir—When loading a BASIC program it is tedious to have to wait until the program is loaded from cassette to avoid loading unwanted noise at the end.

Typing POKE515,0:RUN followed by Return in immediate mode after the program has been SAVED, records the latter on the tape. The recorder is then switched off, and on loading the program, the LOAD flag is turned off, and the program RUNS AUTOMATICALLY.

Roger Darbishire,
King's Lynn, Norfolk.

SOFT INVERSE VIDEO

Sir—By connecting the circuit shown in Fig. 1, to the video circuit of the Compukit, software control of inverse video is possible, which extends the flexibility of the already extensive graphics. Programs can then include routines for displaying selected areas of the screen inversely (black letters on white background).

This is achieved by inserting an exclusive OR gate in the video line between IC42 and IC70. Its inverting input is controlled by software, operating a latch (this may be the inherent latch in IC14 RTS as shown in Fig. 1, or from a latch provided by any of the published Compukit Extension Projects).

The display data line 7 (VD7) is used for selectivity. When the latch (RTS) is low, VD7 in = VD7 out and the display is normal. When (RTS) is high AND VD7 in is high, then VD7 out is low. Characters having a value above 127 are reduced in value by 128 and displayed inversely.

This is shown by the following program which displays "UK101" in black, onto a white rectangle, below which is displayed "UK101" in white.

```
5 FOR I = 1TO16 :?:NEXT
10 A = 53400 : POKE 61440,81
20 FOR I = A + 64 TO A +79: READ
B
30 POKE I,B: POKE I+ 320,B:
NEXT:GOSUB 100
40 A=A+64 : GOSUB 100: A=A+64:
GOSUB 100
50 GOTO 50
60 DATA32,32,85,32,75,32,32,32,49,32,
48
70 DATA32,49,32,32,32
100 FOR I = A TO A + 15 : P =
PEEK(I)
110 P = P+ 128: POKE I, P: NEXT:
RETURN
```

Line 10 operates (RTS) latch and line 110 puts VD7 high over selected rectangle. Normal video can be restored by Warm Reset or in program by Poking 61440,17

CEGMON COMPATIBLE TRACE

Here is a trace program for UK101 which runs under CEGMON without destroying the printing format. It should also run on the Superboard and under other monitors, providing the CTRL C routine is at HEX FB94. It takes advantage of the fact that BASIC stores any number to be output at locations HEX 0100 to 0105 in decimal digits.

After you've cold started, try this:

FOR I=256 TO 261:? CHR$ (PEEK(I));: NEXT

The result will be the last number output, i.e. the number of free bytes. The program is adapted from the one in the CEGMON manual.

```
0294  A9  FF       LDA  £FF
0296  85  5F       STA  5F
0298  A9  80       LDA  £80
029A  85  64       STA  64        set CTRL O flag (don't print)
029C  20  53  B9   JSR  B953      force current line no. into
                                    0100–0105

029F  A2  05       LDX  £05
02A1  BD  00  01   LDA  0100,X    get digits of line no. and
02A4  9D  36  D0   STA  D036,X    'poke' to top right corner of
                                    screen

02A7  CA           DEX
02A8  D0  F7       BNE  02A1
02AA  C6  64       DEC  64        reset CTRL O flag
02AC  4C  94  FB   JMP  FB94      do CTRL C check (exit)
```

If your screen is not 48 characters wide, then byte 02A5 can be changed from HEX 36. HEX 20 should suit even a 25 column screen.

As the program is in page 2 of memory, it is unaffected by a COLD START. It can be moved elsewhere but for ease of use it should start at XX94, i.e. any memory location ending in 94. As listed, it is turned on by POKE 541,2 and turned off by POKE 541,251. When a BASIC program is running, the current line number will be displayed at the top right corner of the screen, and the extra routine does not slow program execution too much, even in a long FOR–NEXT delay loop.

A simple modification will permit single-stepping line by line through a program, waiting until a key is pressed before going to the next line. Make the following changes:

```
02AC  20  00  FD   JSR  FD00      wait for key to be pressed
02AF  4C  94  FB   JMP  FB94      do CTRL C check (exit)
```

As the program occupies only 27 bytes, it's hardly worth putting it on cassette. Enter the program by RESET M 0294 then enter each byte (HEX pair) followed by RETURN. You can then COLD or WARM start as required.

Here is a subroutine which can be called from BASIC, which will scroll the screen contents down instead of up. This has applications in games where it is desired to create an impression of moving forward through obstacles, or of falling missiles etc.

To use the scroll-down routine from BASIC, try

10 POKE 11,204 : POKE 12,2
20 FOR I = 1 TO 32 : X = USR(X) : NEXT

or for new BAS 1/3 chips:

10 FOR I = 1 TO 32 : CALL 716 : NEXT

To scroll diagonally, POKE 727 with 63 or 65 before calling the routine. It works by loading every screen byte in sequence starting at the bottom, and storing it at a location which is greater by HEX 40 (64 decimal). It ends when the byte loaded is CF, which is what it sees when it leaves the top of the screen, despite the fact that there are no memory locations from CFOO to CFFF.

```
02CC  A0  FF       LDY  £FF
02CE  A9  00       LDA  £00
02D0  85  13       STA  13
02D2  A9  D7       LDA  £D7       (D3 for 16 line screen)
02D4  85  14       STA  14
02D6  A9  40       LDA  £40
02D8  85  15       STA  15
02DA  A9  D7       LDA  £D7       (D3 for 16 line screen)
02DC  85  16       STA  16
02DE  B1  13       LDA  (13),Y
02E0  C9  CF       CMP  £CF
02E2  F0  0B       BEQ  02EF
02E4  91  15       STA  (15),Y
02E6  88           DEY
02E7  D0  F5       BNE  02DE
02E9  C6  14       DEC  14
02EB  C6  16       DEC  16
02ED  D0  EF       BNE  02DE
02EF  60           RTS
```

As it stands, the contents of the top line will be left over the other lines. An extra routine to clear the top line is:

```
02F0  A2  40       LDX  £40
02F2  A9  20       LDA  £20
02F4  9D  00  D0   STA  D000,X
02F7  CA           DEX
02F8  D0  FA       BNE  02F4
02FA  60           RTS
```

POKE 11 with 240 to let X=USR(X) know where to jump.

The following gives an effect of travelling through stars:

```
10 S=53248 : N=5
20 FOR I=0 TO N*RND(1) : POKE
   S+63*RND(2),46 : NEXT
30 CALL 716 : CALL 752 : POKE S,32 :
   GOTO 20
```

This works nicely even if the direction of scrolling is altered during the execution of the program, by POKEing 727 with 63 or 65, depending on which SHIFT key is pressed.

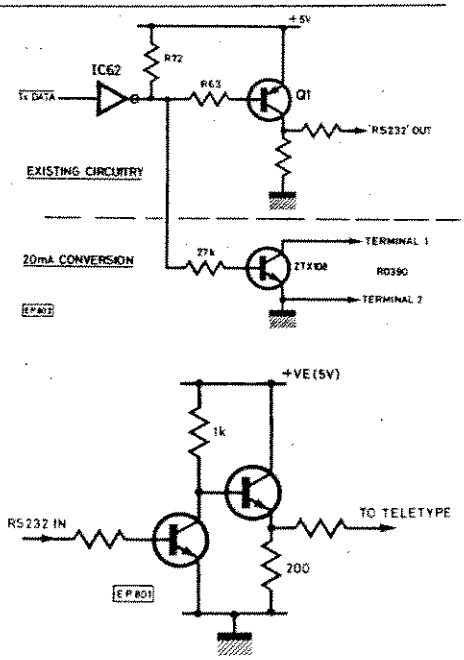
David Henniker, Edinburgh.

390 SUGGESTIONS

UK 101-to-Data Dynamics 390 problems? Try the following:

1) Make either 110 Baud rate mod' of March/June 1980. Break connection between IC62/12 and R72/R63, then reconnect R72/R63 to $\overline{\text{Tx DATA}}$ (IC62/13), to remove data inversion. At the inside rear of the 390 is a 4-terminal block. Connect twin cable to the two left-hand terminals (facing block). Other end of the cable to pins 3 and 7 of J3 (RS232 out). A reduction in value of R65 cures any drop-outs. *M. C. Saltmarsh, Chelmsford.*

2) To obtain the required voltage swing from Compukit's serial output, the following circuit is recommended by *S. Burton, York.*



3) Clock IC57 from C5. Connect IC58/4 to IC58/5. Break between IC57/12 and IC58/4. ACIA will now transmit at 110 Baud. The 390 may require buffering and inverting. A circuit is supplied by *B. J. Hill, Birmingham.*

600 BAUD CASSETTE INTERFACE

By P. MARTIN

THE COMPUKIT as supplied has a 300 Baud cassette interface which performs very well.

This article describes the hardware required to uprate your Compukit's interface to 600 Baud, without modifying it other than the addition of three i.c.s on a small board with five connections. No printed circuit tracks have to be cut.

The modification is designed so that you can use the original 300 Baud or 600 Baud. This means that original tapes at 300 Baud can still be used.

CIRCUIT OPERATION

The clock for the serial interface is derived from the divider chain, and for a Baud rate of 600, C2 needs to be used instead of C3. This is done by IC1. When the Select line is low then C2 is selected. Since the output tones are derived from the ACIA clock when the clock is doubled in frequency the output tones will also double in frequency. This will be a problem with many cassettes as their frequency response drops off quite rapidly after 6KHz. To overcome this, IC2 is used to divide the output tones by 2 when using the higher clock rate. IC3 is used to switch this stage.

CONNECTION TO THE COMPUKIT

There are 5 connections to make to the computer:

1) C1
2) C3
3) Input to IC57 (Pin 2)
4) Q output from IC64 (Pin 11)
5) Output to the cassette jack

Fig. 1 shows where to pick off the C2 and C3 signals. The input to IC57 and the output from IC64 are done in the same manner.

Remove the i.c. from its socket and very carefully bend the required pin to a horizontal position. Put the i.c. back into its socket and solder the required wire to the pin. The output to the cassette can be made directly to pin 9 of J2.

If you find that you have problems with the new interface, it may be that the value of R53 is not correct, and it will need to be changed. The correct value may be found by replacing R53 with a 22k preset pot, and adjusting it to get zero errors on loading a program.

FURTHER MODIFICATIONS

Since the input for 300/600 Baud operation is a single TTL level input by using a simple interface such as a PIA chip, the Baud rate can be controlled by software; this may have applications where you have a cassette file-keeping system using tapes recorded at both speeds.

The second modification concerns even higher Baud rates, and this is where you may run into problems.

The first problem is that standard hardware is not reliable above 1200 Baud due to several factors; the main ones being:

1) The signal generated by the cassette interface is not a sinewave, but an integrated square wave. During playback this contains many high harmonics which during the detection stage can interfere with the wanted signal.

2) The receive circuitry introduces jitter which makes it hard for the monostable to quickly decode the signal.

The answer to the above is a better interface using, for instance, synthesised sinewaves and a phase locked loop detector.

The second major problem when using a high speed interface is the software not being able to keep up with the incoming data. The prime villain is the interpreter, which after it receives a carriage return decodes that line into token symbols, finds where to put it into memory, and then inserts that line. All this takes

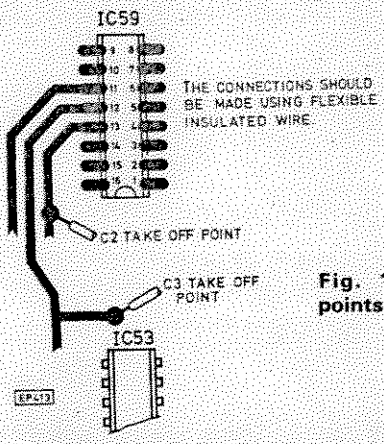


Fig. 1. The pick-off points for C2 and C3

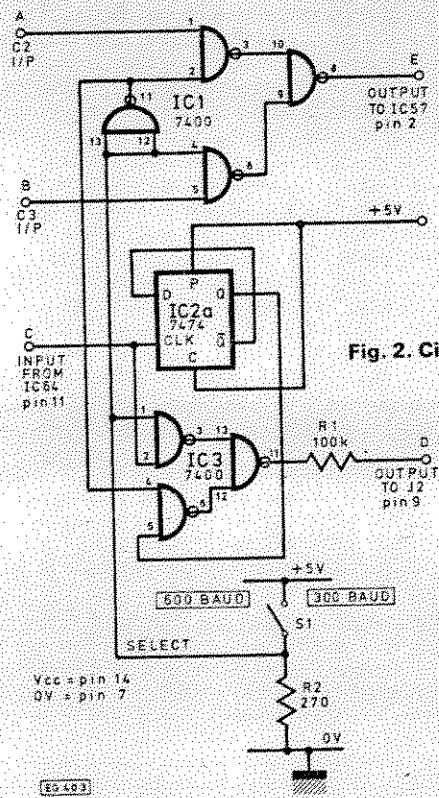


Fig. 2. Circuit diagram

time, and if the interface is delivering another line during this operation then data will be lost. To overcome this the standard Compukit outputs 10 null characters after each carriage return, which gives the interpreter time to do its chores. At higher Baud rates you must provide for more nulls to be printed. One way of doing this is to POKE 13 with the number of nulls you want printed.

CONCLUSION

I have been using this modification for several months now with very good results, and it has certainly made keeping programs on cassette easier. The present Baud rate of 600 is about the highest you can go without substantial modifications to your Compukit and I hope that the problems I have outlined will help those who want a faster interface.

ELAPSED TIME METER

THE meter is connected across the mains supply to the device being timed, after any on/off switch or thermostat.

Because the display is only activated when a reading is required, and is not multiplexed, the circuit shows very little current consumed when counting, and a negligible amount on standby. It will run for at least a year on one PP3 battery.

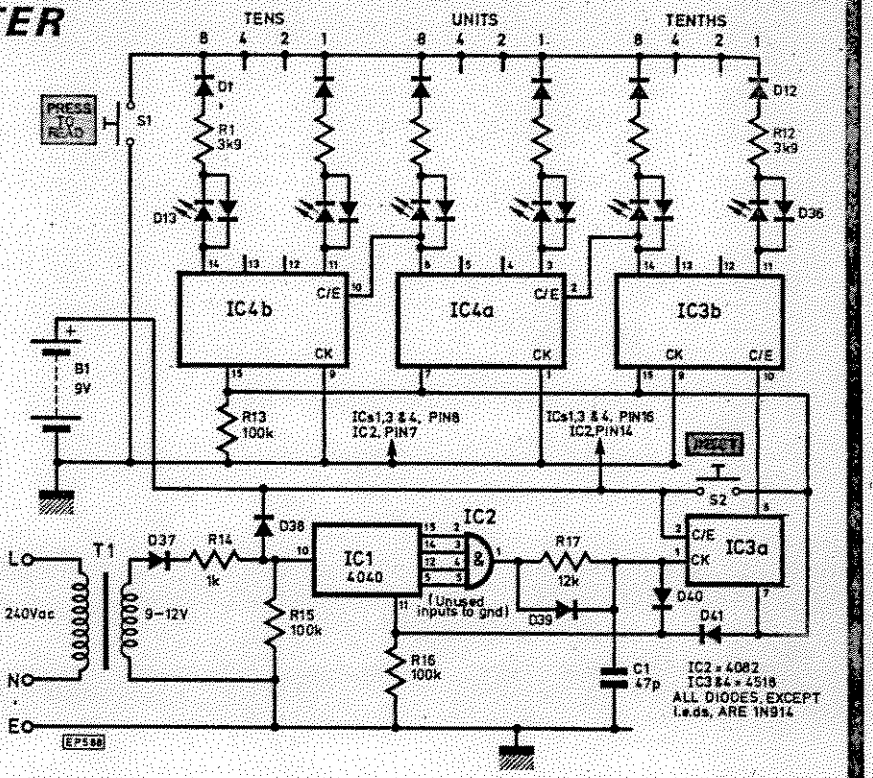
The operation is as follows: when the device being timed is operating, 50Hz pulses are fed from T1 through D37 into IC1. The pulse level is limited to 9V by D38. IC1 is gated by IC2 to divide by 1800. The output of IC2 is stretched by R17 and C1 and clocks IC3a as well as resetting IC1. IC3a gives an output pulse every 0·1 hour, and these pulses are counted by IC3b and IC4, the count being displayed in BCD format when S1 is pressed.

The display l.e.d.s are shunted by diodes, and have other diodes in series, to prevent reverse voltages being applied to the l.e.d.s. The display brightness is kept low to enable the l.e.d.s to be driven directly by the 4518 ICs, and to reduce battery drain.

S2 resets the counters, and D40/D41 ensure that the output pulse from IC2 only resets IC1.

The counter gives a maximum count of 99·9 hours. If a larger count is required, a 4017 could be inserted between IC1 and IC3a, the latter being used for the first (tenths) display. The maximum count will then be 999·9 hours.

A. M. Tucker,
Charminster,
Dorchester.



HEXADECIMAL DISPLAY UNIT

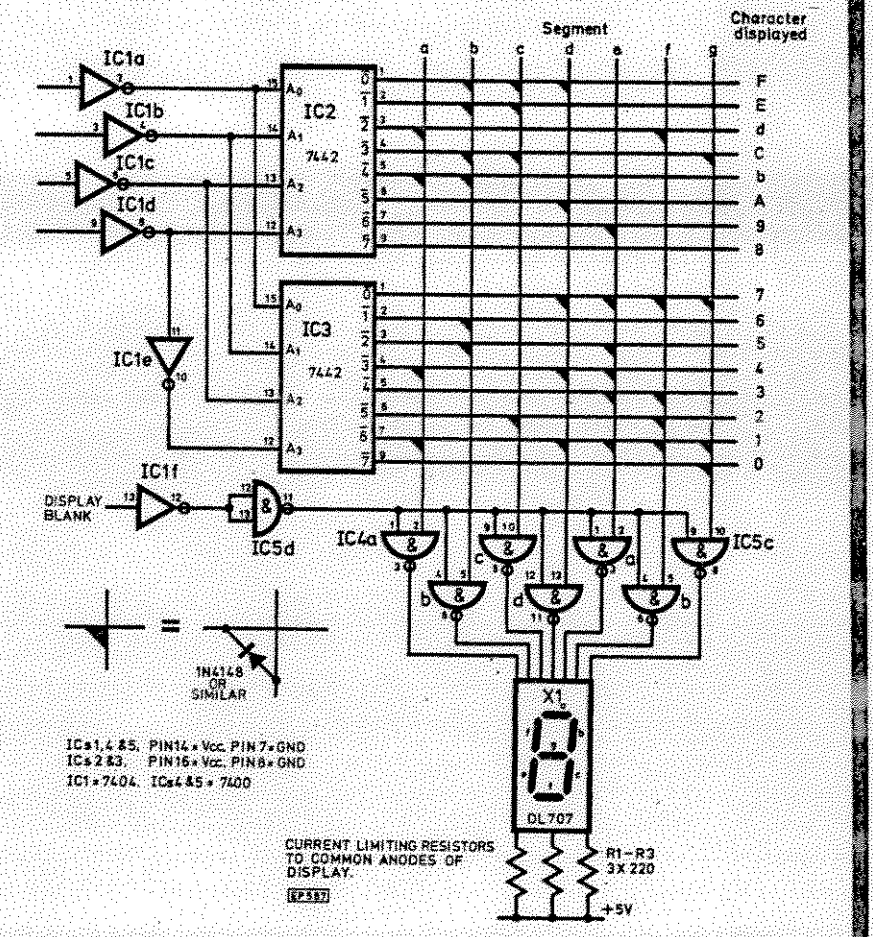
THIS circuit decodes a 4-bit binary number and displays the appropriate hexadecimal character (0–F) on a 7-segment display.

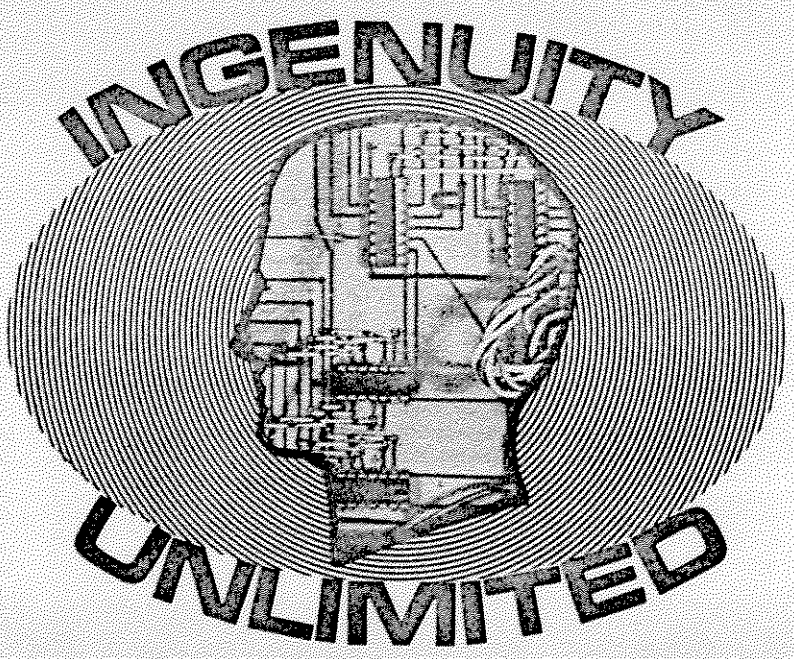
The diode matrix would normally use 78 diodes in a conventional system but this has been reduced to 34 by using diodes in the matrix to stop segments lighting. All segments light otherwise.

Two 7442s are used to produce a 1 to 16 output from the binary input because the 74154 could not pull the segment driver inputs low enough through the diode matrix.

The use of 7400s to drive the display allowed the unused input to be used for display blanking.

J. K. Semple,
Plaxted,
Kent.





A selection of readers' original circuit ideas. It should be emphasised that these designs have not been proven by us. They will at any rate stimulate further thought.

Why not submit *your* idea? Any idea published will be awarded payment according to its merits.

Articles submitted for publication should conform to the usual practices of this journal, e.g. with regard to abbreviations and circuit symbols. Diagrams should be on separate sheets, not inserted in the text.

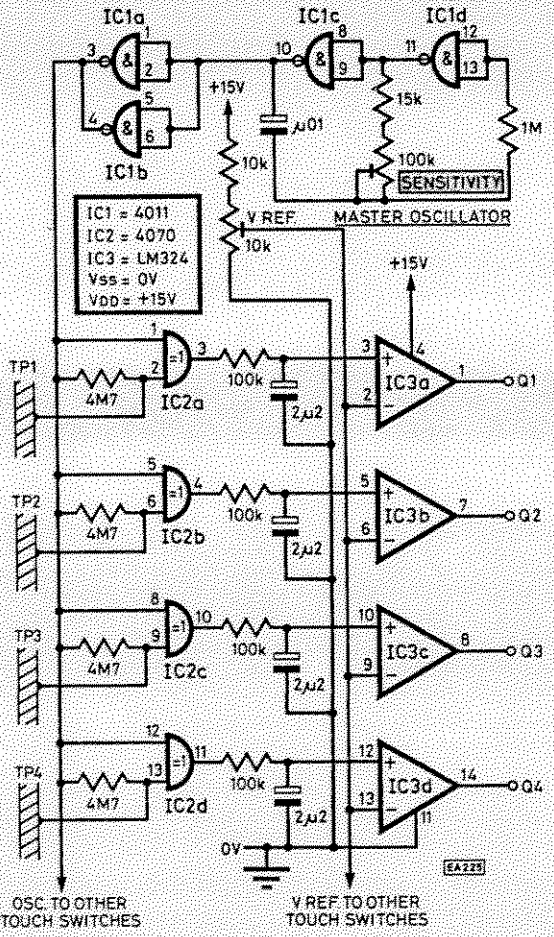
Each idea submitted must be accompanied by a declaration to the effect that it has been tried and tested, is the original work of the undersigned, and that it has not been offered or accepted for publication elsewhere.

CAPACITIVE TOUCH SWITCH

MOST comon touch switches are of the skin resistance type where a finger is used to bridge an electrical contact. Switches of this type are very difficult to make mechanically to give a professional appearance . . . drawing pins are a common resort, and home constructed contacts can look poor as well as being prone to surface oxidation. By sensing the capacitance to ground between a finger and a concealed touch plate, one can get rid of both these problems.

A circuit of such a switch for driving CMOS is shown. IC1 is used to provide a buffered oscillator circuit which drives a phase comparator circuit made up from IC2 an EX-OR device. The sensor plates TP1 to TP4 are connected to the oscillator output via a high value resistor. The phase between the touch plate signal and the oscillator output is measured by the EX-OR gates. As a finger is brought near to the plate the capacitance loading the plate to ground through the finger causes a phase lag of the plate signal. After passing through a low-pass filter this is seen as an increase in phase comparator output which is then detected by quad comparator IC3, causing the relevant Q output to go high.

The touch plates consist of one inch square copper pads on a $\frac{1}{16}$in printed circuit board. The blank side of this is painted and switch touch positions labelled with Letraset. Construction is critical and leads from the sensor plates must be kept below 1in to minimise stray capacitance. This is done by mounting boards containing 4070, LM324 devices on the back of the switch panel perpendicular to it. Veroboard is not recommended.



The circuit is set up by monitoring the voltage at one of the IC2 outputs. The sensitivity control is adjusted to give a mean level of about 1 volt at this pin. On bringing a finger up against the switching area a rise in the appropriate output should be noticed. The voltage reference should be set so that the comparator just triggers at this level. The zero voltage supply line should preferably be connected to the mains earth line.

F. T. Dart,
East Kilbride,
Glasgow.

RESIDENT EDITOR

BY NIGEL CLIMPSON

MOST owners of microcomputers would be pleased if, for minimal hardware changes, they could provide themselves with an extra 1000 or so bytes of ROM storage. Into these 1000 bytes could be placed favourite machine code programs or perhaps some system software to extend the capabilities of their machines. These programs would of course be available from switch-on and be permanently stored.

This can be done for the UK101, but of course there is a snag. You will have to obtain a single supply 2716 EPROM and find some means of programming it.

The secret of this miraculous amount of available space lies in the Monitor ROM. Examination of the contents of the ROM discloses that it contains two almost identical sections at addresses in the ROM of 000-3FF and 400-7FF where 000 is located in the machine's memory map at F800. The major differences in the two sections are that some of the jumps are to different addresses, and that one of the sections contains a routine to clear 2K of video memory as against 1K in the other.

The UK101 only uses one of these two sections and I suspect that this comes about due to the machine's OSI heritage.

The point of all this is that we can dispose of the contents of the ROM from 000-3FF and the UK101 still works quite happily. It should be possible to gain a bit more space if you don't have a disc drive by removing part of the floppy bootstrap which runs from 400-4D4, but beware because some of the subroutines in this section are called from other areas of the Monitor.

```
000  98 48 AC F9 02 AD F8 02 91 F9 88 CE F9 02 20 3B
010  F9 68 A8 A9 00 60 4C 00 F8 98 48 20 45 F9 AC F9
020  02 A9 23 91 F9 EE F9 02 D0 02 E6 FA C8 20 3B F9
030  68 A8 A9 00 60 CA 10 04 E8 4C 99 A3 8A 48 98 48
040  AC F9 02 AE 00 02 AD FA 02 F0 05 AD F8 02 91 F9
050  A9 20 9D 00 D3 CA 88 CE F9 02 CE 00 02 A9 9A 9D
060  00 D3 CE F6 02 AD FA 02 F0 03 20 3B F9 68 A8 68
070  AA A9 00 60 20 BA FF C9 15 F0 1A C9 1C F0 36 C9
080  04 F0 96 C9 02 F0 0F C9 06 F0 6F C9 0D F0 50 20
090  1D F9 4C 99 A3 38 98 48 AD FA 02 F0 08 AC F9 02
0A0  AD F8 02 91 F9 EE FA 02 AD FA 02 C9 10 F0 1F C9
0B0  01 D0 96 AD 00 02 8D F9 02 AD F9 02 E9 40 B0 02
0C0  C6 FA 8D F9 02 A8 20 3B F9 68 A8 A9 00 60 A9 00
0D0  8D FA 02 A9 D3 85 FA AD 00 02 8D F9 02 D0 E6 A9
0E0  00 8D FA 02 85 F9 A9 D3 85 FA A9 20 8D F8 02 A9
0F0  CD 8D F6 02 8D F9 02 A9 0D 60 98 48 20 45 F9 AD
100  F8 02 AC F9 02 91 F9 8D F7 02 C8 EE F9 02 D0 02
110  E6 FA 20 3B F9 68 A8 AD F7 02 4C 99 A3 48 AD 00
120  02 CD F6 02 10 0D 38 AD F9 02 E9 40 8D F8 02 8D
130  F9 02 AD 00 02 8D F6 02 68 60 31 F9 8D F8 02 A9
140  A9 9A 91 F9 60 A9 48 C5 0F D0 98 AD F9 02 E9 30
150  4C 58 F9 AD F9 02 E5 0F C9 0C F0 10 C9 4C F0 0C
160  C9 8C F0 08 C9 CC F0 04 20 1D F9 60 48 AC F9 02
170  A9 20 91 F9 68 18 69 40 90 02 E6 FA 8D F9 02 20
180  1D F9 60 A9 00 85 E1 A8 A9 D0 85 E2 A9 20 91 E1
190  C8 C0 00 D0 F9 A6 E2 E0 D3 F0 06 E8 86 E2 18 90
1A0  ED 60 AD 06 FA 88 D0 05 CE F5 02 F0 09 CA D0 F5
1B0  AE F4 02 4C A2 F9 60 FF FF FF FF FF FF FF FF FF
1C0  FF FF 20 5E FA C9 3B F0 13 C9 24 D0 F5 20 3F FA
1D0  8D 24 02 20 3F FA 8D 23 02 4C 6C FA A9 00 8D 23
1E0  02 8D 24 02 20 2C FA AA 20 2C FA 85 FA 20 2C FA
1F0  85 F9 8A 48 A2 00 20 2C FA 81 F9 68 AA E6 F9 D0
200  02 E6 FA CA D0 EC 20 3F FA CD 24 02 D0 08 20 3F
210  FA CD 23 02 F0 AC A9 00 B9 74 FA F0 06 20 EE FF
220  C8 D0 F5 20 61 FA C9 47 D0 F9 F0 96 20 3F FA 18
230  6D 23 02 8D 23 02 90 03 EE 24 02 AD 22 02 69 00
240  42 FA 20 5E FA C9 41 90 02 E9 07 29 0F 0A 0A 0A
250  0A A0 04 2A 26 22 02 88 D0 F9 AD 22 02 60 A9 80
260  2C A9 10 8D 03 02 20 BA FF 4C EE FF A9 00 8D 03
270  02 6C 23 02 0D 0A 0A 4F 42 4A 20 43 48 45 43 43
280  53 55 4D 20 45 52 52 0D 0A 52 45 57 49 4E 44 20
290  59 41 53 54 20 45 52 52 20 0D 20 54 59 50 45 20
2A0  47 20 54 4F 20 52 45 53 54 41 52 54 00 C9 4C D0
2B0  03 4C 7C FE C9 51 F0 03 4C 39 FE 4C C2 F9 A9 BA
2C0  8D 18 02 A9 FF 8D 19 02 4C 00 FE FF FF FF FF FF
```

and then FFs until . . .

Of course you can't change the contents of the existing ROM and that is where the 2716 single supply EPROM comes in.

HARDWARE CHANGES

Assuming you have programmed the new ROM with the half of the Monitor you need to keep and with your own programs, now you need to make the hardware changes. To ensure that the 2716 is enabled properly some minor modifications need to be made to the computer's circuit board, and this is made slightly harder than necessary by the fact that the ROM addressing cir-

EDITOR-IN-MONITOR ROM

```
3F0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
400  20 0C FC 6C FD 00 20 0C FC 4C 00 FE A0 00 8C 01
410  FF 8C 00 C0 A2 04 8E 01 C9 8C 03 C9 88 8C 02 C0
420  8E 03 C0 8C 02 C0 A9 FB D0 09 A9 02 2C 00 C0 F0
430  1C A9 FF 8D 02 C0 20 A5 FC 29 F7 8D 02 C0 20 A5
440  FC 09 08 8D 02 C0 A2 18 20 91 FC F0 DD A2 7F 8E
450  02 C0 20 91 FC AD 00 C0 30 FB AD 00 C0 10 F3 A9
460  03 8D 10 C0 A9 58 8D 10 C0 20 9C FC 85 FE AA 20
470  9C FC 85 FD 20 9C FC 85 FF A0 00 20 9C FC 91 FD
480  C8 D0 F8 E6 FE C6 FF D0 F2 86 FE A9 FF 8D 02 C0
490  60 A0 F8 88 D0 FD 55 FF CA D0 F6 60 AD 10 C0 4A
4A0  90 FA AD 11 C0 60 A9 03 8D 00 F0 A9 11 8D 00 F0
4B0  60 48 AD 00 F0 4A 4A 90 F9 68 8D 01 F0 60 49 FF
4C0  8D 00 DF 49 FF 60 48 20 CF FC AA 68 CA E8 60 AD
4D0  00 DF 49 FF 60 C9 1C F0 03 4C 74 A3 CA 10 04 E8
4E0  4C 59 A3 8A 48 AE 00 02 A9 20 9D 00 D3 CE 00 02
4F0  CA A9 9A 9D 00 D3 68 AA 4C 59 A3 FF FF FF FF FF
500  8A 48 98 48 A9 01 20 BE FC 20 C6 FC D0 05 0A D0
510  F5 F0 53 4A 90 00 0A E0 01 D0 F3 A9 1B D0 21 20
520  C0 FD 98 8D 13 02 0A 0A 0A 38 ED 13 02 8D 13 02
530  8A 4A 20 C8 FD D0 2F 18 98 6D 13 02 A8 B9 CF FD
540  CD 15 02 D0 26 CE 14 02 F0 2B A0 05 A2 C8 CA D0
550  FD 88 D0 F8 F0 AE C9 01 F0 35 A0 00 C9 02 F0 47
560  A0 C9 C9 20 F0 41 A9 00 8D 16 02 8D 15 02 A9 02
570  8D 14 02 D0 0F A2 96 CD 16 02 D0 02 A2 14 8E 14
580  02 8D 16 02 A9 01 20 BE FC 20 CF FC 4A 90 33 AA
590  29 03 F0 0D A0 19 AD 15 02 19 0C A0 F0 09 98 A0
5A0  00 E0 20 D0 02 A0 C8 AD 15 02 20 7F C9 20 F0 97
5B0  8C 13 02 18 6D 13 02 8D 13 02 68 A8 68 AA AD 13
5C0  02 60 D0 92 A0 20 D0 DF A0 98 8C 0A 90 FC 60 D0
5D0  BB 2F 20 5A 41 51 2C 4D 4E 42 56 43 58 4B 4A 48
5E0  47 46 44 53 49 55 59 54 52 45 57 30 39 0D 5E 4F
5F0  4C 2E 20 1C 2D 0A 30 39 38 37 36 35 34 33 32 B1
600  A2 28 9A D8 EA EA EA EA EA EA EA EA A2 04 A9 D0
610  85 FF A9 00 85 FE 85 FB A8 A9 20 91 FE C8 D0 FB
620  E6 FF E4 FF D0 F5 84 FF F0 19 20 E9 FE C9 2F F0
630  1E C9 47 F0 17 4C AD FA EA 20 93 FE 30 EC A2 02
640  20 DA FE B1 FE 85 FC 20 AC FE D0 DE 6C FE 00 20
650  E9 FE C9 2E F0 D4 C9 0D D0 0F E6 FE D0 02 E6 FF
660  A0 00 B1 FE 85 FC 4C 77 FE 20 93 FE 30 E1 A2 00
670  20 DA FE A5 FC 91 FE 20 AC FE D0 D3 85 FD F0 CF
680  AD 00 F0 4A 90 FA AD 01 F0 EA EA EA 20 7F 60 00
690  00 00 00 C9 30 30 10 C9 3A 30 03 C9 41 30 0A C9
6A0  47 10 06 38 E9 07 29 0F 60 A9 00 60 A2 03 A0 00
6B0  B5 FC 4A 4A 4A 4A 20 CA FE 85 FC 20 CA FE CA 10
6C0  EF A9 20 8D 62 D1 8D 63 D1 60 20 0F 00 30 C9 3A
6D0  30 03 18 69 07 99 5E D1 C8 60 A9 04 0A 0A 0A 0A
6E0  8A 36 FC 36 FD 88 D0 F8 60 A5 FB D0 03 4C 00 FD
6F0  74 F8 69 FF 9B FF 8B FF 96 FF 30 01 00 FE C0 01
700  D8 A2 28 9A A0 0A B9 EF FE 99 17 02 88 D0 F7 20
710  A6 FC 8C 12 02 8C 03 02 8C 05 02 8C 06 02 AD 20
720  FF 8D 00 02 A9 20 99 00 D3 99 00 D2 99 00 D1 99
730  00 D0 C8 D0 F1 B9 5F FF F0 06 20 2D BF C8 D0 F5
740  20 BA FF C9 4D D0 03 4C BE FA C9 57 D0 03 4C 00
750  00 C9 43 D0 03 4C 11 BD C9 44 D0 A4 4C 00 FC 44
760  2F 43 2F 57 2F 4D 20 3F 00 20 8D BF 48 AD 85 02
770  F0 22 68 20 B1 FC C9 0D D0 13 48 8A 48 A2 0A A9
780  00 20 B1 FC CA D0 FA 68 AA 60 60 48 CE 03 02 A9
790  00 8D 05 02 68 60 48 A9 01 D0 F6 AD 12 02 D0 19
7A0  A9 FE 8D 00 DF 8C 03 DF 70 0F A9 F3 8D 00 DF 2C
7B0  00 DF 70 05 A9 03 4C 36 A6 60 2C 03 02 10 19 A9
7C0  FD 8D 00 DF A9 10 2C 00 DF F0 0A AD 00 F0 4A 90
7D0  EE AD 01 F0 60 EE 03 02 4C 00 FD FF FF FF FF FF
7E0  CD 2F 00 00 03 FF 9F 00 03 FF 9F 6C 18 02 6C 1A
7F0  02 6C 1C 02 6C 1E 02 6C 20 02 30 20 00 FF C0 91
```

E/MONITOR RELOCATED

Sir—In response to Mr. Walton's enquiry in the January 81 issue in Micro Prompt, I have successfully relocated the Extended Monitor to run at $1800–$1FFF leaving 6K below it.

The Extended Monitor uses an address table to select the function required. The ASCII value of the letter typed in is converted into an index which points to an address within the table. The 2 bytes of the address are fetched and pushed onto the stack, followed by the Processor Status Word. Control is then passed to the subroutine by executing an RTI (Return from Interrupt) instruction. The return address for the subroutine has been pushed onto the stack prior to this.

The address table is located at $0960–$0999 and is shown in Fig. 1. The procedure for successful relocation is as follows:

(1) Make a new copy of the program at the required location using the relocation facility.

(2) Determine the relocation constant (the amount by which the starting address has been increased), e.g. to locate at $1800, the relocation constant is $1000.

(3) Add the relocation constant to $0960 to give the base address of the new address table. Also, add the relocation constant to the addresses stored within the table.

(4) Step through the table starting at the base address you have calculated and change all the incorrect table entries (this will be most of them).

Character	Table Address	Contents
@	0960	0B53
A	0962	0BB3
B	0964	0C9A
C	0966	0CBF
D	0968	0CD2
E	096A	0C57
F	096C	0DA3
G	096E	0BC1
H	0970	0E33
I	0972	0C12
J	0974	0809
K	0976	0BAF
L	0978	0F43
M	097A	0D91
N	097C	0D35
O	097E	0F29
P	0980	0BB0
Q	0982	0D14
R	0984	0DB7
S	0986	0EC3
T	0988	0C6E
U	098A	084C
V	098C	0F3B
W	098E	0D7E
X	0990	0BB2
Y	0992	0BB1
Z	0994	0FB7
Return Address	0996	081E
Brkpt. Routine	0998	0BC7

Fig. 1 Extended Monitor Address Table

The new version of the Extended Monitor is now ready to run. If after jumping into the new version, all seems well, make sure by erasing the old version using "Fill".

Fig. 2 shows the table values required to run at $1800 and incorporates the following changes:

J transfers control to $0300
U Jumps to Cold Start Basic ($BD11)
Z jumps to the old monitor at $FE00

Having relocated the Extended Monitor, you now need to either burn it into EPROM or else make a new self-executing tape. As the procedure for making a new tape is rather long, I suggest that anyone wishing to know how to do this should write to me (via PE—Ed).

John Riley, B.Sc., Coventry.

Character	Table Address	Contents
@	1960	1B53
A	1962	1BB3
B	1964	1C9A
C	1966	1CBF
D	1968	1CD2
E	196A	1C57
F	196C	1DA3
G	196E	1BC1
H	1970	1E33
I	1972	1C12
J	1974	0300
K	1976	1BAF
L	1978	1F43
M	197A	1D91
N	197C	1D35
O	197E	1F29
P	1980	1BB0
Q	1982	1D14
R	1984	1DB7
S	1986	1EC3
T	1988	1C6E
U	198A	BD11
V	198C	1F3B
W	198E	1D7E
X	1990	1BB2
Y	1992	1BB1
Z	1994	FE00
Return Address	1996	181E
Brkpt. Routine	1998	1BC7

Fig. 2 Address Table for Extended Monitor at $1800

M/C FOR PSG

Sir—When using the programmable sound generator, it soon becomes apparent that machine code is much faster and more efficient on memory than BASIC. But even here, with several more complex sound effects, some sort of loader is required. The following program takes data from a table and loads it into the PSG registers. Put the starting address of the table in $E0, $E1 (lo byte first). The table should follow the format:

REGISTER
CONTENT
REGISTER
CONTENT
REGISTER
CONTENT
$FF

To execute .GO222

```
0222  A000      LDY#$00
0224  B1E0      LDA($E0),Y
0226  C9FF      CMP#$FF
0228  D001      BNE$022B
022A  60        RTS
022B  8DF0F1    STA$F1F0
022E  C8        INY
022F  B1E0      LDA($E0),Y
0231  8DF1F1    STA$F1F1
0234  C8        INY
0235  18        CLC
0236  90EC      BCC $0224
```

The addresses at $022C-D and $0232-3 will need altering for PSGs not located at $F1F0 and $F1F1.

T. D. Allen, Poole.

OVERLOADED BUS

Sir—There is a fault in the UK101 circuit which may well have put a number of computers into cupboards gathering dust. It concerns the loading on the address bus line A10. This processor line is very heavily loaded with LSI integrated circuits (16 RAM inputs and 5 ROM inputs). It is also expected to drive one full t.t.l. load, namely IC21b – $\frac{1}{18}$ of a 7404.

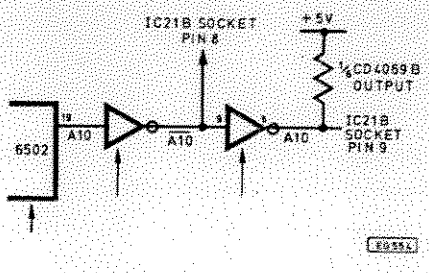
I had rather a lot of problems getting my UK101 to work – 5 or 6 faulty t.t.l. circuits and a solder bridge which destroyed two 74123s. Since getting it working, one ROM and three RAMs have failed. My own preference for a Buffer, after giving considerable thought to the subject, is to use a 74LS output with a pull up resistor to the +5V line. This resistor should ideally be a constant current resistor and I've realised that $\frac{1}{18}$ of a CD40698 with input shorted to earth is the ideal sort of resistor – the gate output, of course.

The circuit I propose to use is shown below left, and I welcome comments on this.

On further consideration of the UK101 circuit I realize that the processor R/W line needs similar treatment. A8 and A9 also look poor, but I would guess that IC171, a 74LS139, has inputs which cause fewer problems.

There are some spare t.t.l. inverters available in IC18 so all one really needs is a CD40698 to make these changes, though I will have to open the case of my UK101 to check this.

J. D. Owen, Pendine, Dyfed.

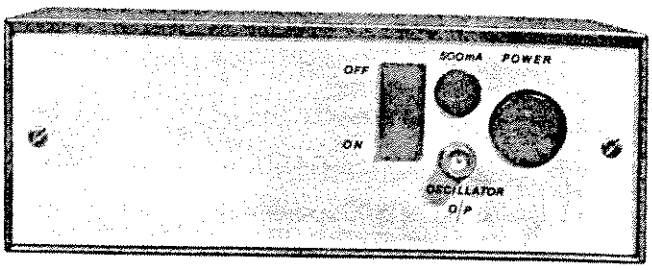


tangular I.E.C. switch mounted on the rear side of the instrument, and a safe 3-pin earthed type mains "Euroconnector" type socket is used.

The supply protection is 500mA/240V "Fast-Blo" 20mm glass type fuse, housed in the appropriate type of fuseholder and mounted directly next to the mains socket.

After completing the assembly of all the components and sub-systems, no alignment or tuning is required and the instrument should work from go. Because of the nature of the ICM 7216 chip, the frequency will be displayed in KHz, and time is displayed in µSec.

The display is multiplexed at 500Hz with about 12·5 per cent duty cycle for the individual digit. The ICM 7216A is designed for common anode display and a typical peak segment current consumption is approximately 25mA. An interesting point is that Intersil claim that in the "Display Off' mode both digit drivers and segment drivers are turned off, enabling the above display to be used for other functions should the instrument be incorporated into a multi-function system. ★



MICRO PROMPT

The hardware and software exchange point for PE computer projects

OF TELETYPE AND BAUD

Sir—As yet another satisfied compukit UK101 owner I have had particularly great pleasure from the many notes and updates in relation to this system. In this connection I have made a couple of observations that might interest other readers, as well. First, I have found that the hardware modification described by Dr. A. A. Berk in his 'compukit update' of March 1980 can be simplified somewhat.

However, there is no need for his point (3). Leaving pin 11 of IC63 connected with pin 11 of IC57 instead of moving it to pin 12 will still give the 'divide-by-nine' effect that is the aim of the rearrangement. The main difference will be that of the duration of the pulse fed from IC57 to IC63. But since it only functions as a clock pulse for IC63, its duration is not cricitical.

I quite agree that it may be worth while to make the modification switchable, and this will be much easier by leaving pin 11 of IC63 tied to pin 11 of IC57. The remaining changes, including the shift from 'C3' to 'C5' of IC60 for the clock pulses to be fed into IC57, can then be completely taken care of using any ordinary type of double pole, double throw switch. I have found this to work perfectly!

On the other hand, in his 'compukit update' of June 1980, Dr. Berk also refers to a soft BAUD rate modification suggested by Mr. C. S. K. Clapp. It works through the modification of the 6850 ACIA routine, by POKEing 82 into its address of 61440, in combination with the modifications shown in his Fig. 2. As can be seen, the resetting of the counter IC57 after modifying is obtained—not by 'clear'ing, as originally—but by loading a binary number into its data inputs a, b, c and d on reaching its twelfth count. After the modification one of these data inputs is tied to the 'RTS'-output of the ACIA. If this output is at a logical 'low', IC57 will then receive zero to its data inputs (all the others remaining permanently grounded). During initialization through the compukit monitor routine, the 'RTS'-output will be set to a logical 'low' level, corresponding to just this case. Consequently, the counter will then go through a 'divide-by-thirteen' cycle, resulting in 300 BAUD, exactly as in the unmodified case.

By POKEing 82 into the ACIA address, one of the changes produced will be the setting of 'RTS' to a logical 'high' level. This, then, results in presetting the IC57 counter to a non-zero number, with the overall consequence of reducing the counting cycle. However, by connecting 'RTS' with pin 4 (data input b) of IC57 the effect of preloading with binary 0010 (decimal 2) will result, giving a net 'divide-by-eleven' cycle. This is clearly wrong. 'RTS', from pin 5 of IC14 should instead be tied to pin 5 of IC57! This gives preloading with binary 0100 (decimal 4), resulting in a 'divide-by-nine' cycle.

The other effect of 'POKE 61440,82' is to change the internal division ratio of the 6850 ACIA from 1:16 to 1:64 (while maintaining the output format of 1 start bit +8 bits +2 stop bits, with no parity, also used in the default case). Thus, the resulting BAUD rate can readily be calculated from the 125 kilohertz pulse train that IC57 receives from 'C3' of the master counting chain:

*125000/(9*2*64)=108·5*

This is, of course, exactly the same BAUD rate as that obtained by the hardware modifications described earlier, and it is definitely close enough to the ideal 110 BAUD for problem-free operation with an ordinary teletype printer. I have, as a matter of interest, made it a point here to support this claim by choosing to write these lines on my compukit and have them printed just on such a printer, hooked onto my 108·5 BAUD converted interface through a 20 milliamps current loop. For convenience, I chose the hardware version, myself (although I also tested out the soft BAUD modification). I now can select to my heart's delight from tele-type 110 BAUD, cassette 300 or 600 BAUD, and finally processor frequencies of 1 or 2 megahertz! All works very reliably.

I have nothing but praise for the potential of the compukit system, considering its reasonable price class. And I hope, among other things, to see a great many more write-ups relating to it, in future issues of your interesting magazine.

Gisle K. Dyvik
Norway

UNCRASH

Sir—During my thirteen months use of my Ohio Superboard, I have discovered a few most interesting and original short cuts for anyone using the machine. One I am submitting for publication here. The routine below can be used for any Ohio machine, and the UK101. Its purpose is to get back into BASIC with the user's program intact. During use of the machine, page zero can be disrupted due to a careless POKE, programming error, etc. This is where my routine comes in. With page zero in disarray, it is impossible to get back into BASIC and/or save your program or in any way avoid the loss associated with a crash. However, by doing the following, if the RAM storing the program has not been disrupted, the user may continue programming in the usual way as though nothing had happened.

1) **PRESS** *BREAK, C, BREAK, M*
2) **TYPE:** */4C, 74, A2, 4C, C3, A8, Ø5. AE*
3) **PRESS** *BREAK, W*
4) *A warm start should now occur. You may now carry on in the usual way.*
5) *NOTE: If locations 11 and 12 have been changed, they will now be restored to their original value. They may be changed back to your values by POKEing them in the usual way.*

R. Wells,
Margate.

MAPPING, GET IT RIGHT!

Sir—In your September issue Mr. M. C. Mannering pointed out an error in the address decoding of the UK101. However a second fault also exists in the decoding: The keyboard address decoding only uses address lines A15–A10 for generating the

keyboard enable signals $\overline{RKB}$ and WKB. This results in the keyboard occupying a 1K block of memory from DC00-DFFF (Hex), instead of just one location as suggested in the manual.

As Mr. Mannering found these faults only show up when experimenting with memory mapped peripherals. Thus anyone contemplating adding an expansion should have a clear idea of what areas of memory are available. With this in mind before expanding my Compukit, I made out two useful charts.

The first is a hardware memory map which is intended to show exactly where devices are found in memory and where free blocks occur. The second chart indicates what address decoding is available on-board, and the range of addresses which they cover.

These charts show the two faults, but also indicate the existence of useful empty blocks of memory and spare decode lines available. Two separate 1K blocks with Read and Write $\overline{Enables}$ are available and would be perhaps suited to an expansion to the video RAM or anything else requiring separate Read, Write Enables. IC17 provides effectively 7 spare decode lines for blocks of 256 Bytes form F100-F7FF. This is achieved by using A10 as another "Enable" line. This way two expansion's may share a common output provided A10 activates one when high and another when low. Spare "Enable" lines suitable for 8K RAM cards are also available.

I think these charts would be very useful for any of your readers wishing to expand their Compukit, as it helps avoid mistakes like placing a programmable sound generator in the ACIA memory block (or didn't you notice that slip?).

David McDonnell,
Eire

Hmmph . . . Yes, we did notice . . . afterwards! See "PSG Bug", Micro Prompt, November 1980—Ed.

XY VIDEO

Sir—I would like to put forward a much more simple way to 'POKE' information onto the screen. The statement is:

*LET P=53196 + X + (Y*64)*
POKE P, N

where X is the x-coordinate
Y is the y-coordinate

X must be > 0 and < the end of the line
Y must be > 0 and < 17
eg. If X=24 and Y=7 and N=161, then a white block will be placed at location 53668.

Gerhart Ellett,
Gt. Yarmouth.

ALWAYS A SIMPLER WAY

Sir—In response to Mr. J. Plews's letter, the INT function in BASIC always rounds down. If one wishes to round to the nearest integer you have to add .5 (ie. PINT (10/(2↑1) + .5) this returns to the value 5.

I'm sure you could have found something better to publish in place of "Just A Little Something". This three line program does the same thing, but for any base less than ten.

```
10 INPUT "NUMBER, BASE"; B$, A
20 FOR B = 1 TO LEN (B$):
   T = T + VAL (MID$(B$, B, 1)) *
   A↑(LEN(B$) – B): NEXT
30 PRINT B$; "=", A: RUN
```

R. Armstrong,
Kilmacolm.
Renfrewshire.

NULL RETURN RECTIFIER

Sir—In the July 1980 Micro Prompt, David Swash pointed out that for the UK101, if Return is pressed in response to an INPUT, then there is a return to command mode, but the program can be restarted using Cont. Although this is true, it is not always very useful, for example if the VDU display is a combination of direct mapping and PRINT statements.

The problem can sometimes be avoided by keyboard polling, or by PEEK (531), this

UK101 HARDWARE MEMORY MAP **Fig. 1**

Location Hex.	Location Decimal	Memory Type	COMMENTS
Ø Ø Ø Ø	0	RAM	PAGE ZERO;
TO			SCRATCH PAD RAM
ØØFF	255		
Ø1ØØ	256	RAM	PAGE ONE;
TO			STACK
Ø221	545		
Ø222	546	RAM	UNUSED—USING NEW MONITOR EPROM
TO			This region seems to exist between Ø223 and
Ø2FA	762		Ø2FA
Ø3Ø1	769	RAM	PAGE THREE;
			BEGINNING OF BASIC WORKSPACE
Ø3FF	1Ø23	RAM	END OF FIRST BLOCK OF 1K RAM
Ø7FF	2Ø47	RAM	END OF 2K OF RAM
ØBFF	3Ø71	RAM	END OF 3K OF RAM
ØFFF	4Ø95	RAM	END OF 4K OF RAM
13FF	5119	RAM	END OF 5K OF RAM
17FF	6143	RAM	END OF 6K OF RAM
1BFF	7167	RAM	END OF 7K OF RAM
1FFF	8121	RAM	END OF 8K OF RAM
			END OF ON-BOARD USER RAM
3FFF	16,383		END OF 16K OF RAM
5FFF	24,575		END OF 24K OF RAM
7FFF	32,767		END OF 32K OF RAM
9FFF	4Ø,959		END OF 4ØK OF RAM
			HIGHEST ADDRESS TO WHICH USER MEMORY MAY BE CONTINUOUSLY EXPANDED
A Ø Ø Ø	4Ø,96Ø	ROM	BEGINNING OF 8K BASIC INTERPRETER
TO			
BFFF	49,151	ROM	END OF BASIC INTERPRETER
C Ø Ø Ø	49,152		FREE; TOTAL = 4K
TO			
CFFF	53,247		
D Ø Ø Ø	53,248	RAM	VIDEO REFRESH MEMORY
TO			TOTAL = 1K
Ø3FF	54,271		
Ø4 Ø Ø	54,272		FREE; TOTAL = 2K
TO			
ØBFF	56,319		
D Ø Ø Ø	56,320	PERIPHERAL	KEYBOARD OCCUPIES THIS BLOCK
TO			TOTAL = 1k
DFFF	57,343		
E Ø Ø Ø	57,344		FREE; TOTAL = 4K
TO			
EFFF	64,439		
F Ø Ø Ø	61,44Ø	PERIPHERAL	ACIA; TOTAL = 256 BYTES
TO			
FØFF	61,695		
F1ØØ	61,696		FREE; TOTAL = 1791 BYTES
TO			
F7FF	63,487		
F8ØØ	63,488	ROM/EPROM	OPERATING MONITOR
TO			
FFFF	65,535		END OF DIRECTLY ADDRESSABLE MEMORY

UK101 ADDRESS DECODING **Fig. 2**

NOTE: all outputs are active low

Range (Hex) Low	High	IC No.	Pin No.	In Use	COMMENTS
0000	1FFF	23	15		8K BLOCK; ENABLES IC22
0000	03FF	22	7		$\overline{RS0}$: 1K RAM ENABLE
0400	07FF	22	9		$\overline{RS1}$: 1K RAM ENABLE
0800	0BFF	22	10		$\overline{RS2}$: 1K RAM ENABLE
0000	0FFF	22	11		$\overline{RS3}$: 1K RAM ENABLE
1000	13FF	22	12		$\overline{RS4}$: 1K RAM ENABLE
1400	17FF	22	13		$\overline{RS5}$: 1K RAM ENABLE
1800	1BFF	22	14		$\overline{RS6}$: 1K RAM ENABLE
1C00	1FFF	22	15		$\overline{RS7}$: 1K RAM ENABLE
2000	3FFF	23	14		8K BLOCK; UNUSED
4000	5FFF	23	13		8K BLOCK; UNUSED
6000	7FFF	23	12		8K BLOCK; UNUSED
8000	9FFF	23	11		8K BLOCK; UNUSED
A000	BFFF	23	10		8K BLOCK; ENABLES IC17 DECODER No. 1
A000	BFFF	15d	13		BS BASIC SELECT FOR 8K ROM
A000	A7FF	17	7		$\overline{BS0}$; BASIC ROM SELECT
A800	AFFF	17	6		$\overline{BS1}$; BASIC ROM SELECT
B000	B7FF	17	5		$\overline{BS2}$; BASIC ROM SELECT
B800	BFFF	17	4		$\overline{BS3}$; BASIC ROM SELECT
C000	DFFF	23	9		8K BLOCK; ENABLES IC20 $\overline{V}$ LINE ALSO DECODES THIS BLOCK
D000	D3FF	20	12		1K BLOCK; DECODES VIDEO RAM ON READ PORTION OF R/$\overline{W}$ CYCLE
D000	D3FF	20	7		1K BLOCK; DECODES VIDEO RAM ON WRITE PORTION OF R/$\overline{W}$ CYCLE
D400	D7FF	20	13		1K BLOCK; ENABLED ON $\overline{READ}$ PORTION
D400	D7FF	20	9		1K BLOCK; ENABLED ON $\overline{WRITE}$ PORTION
D800	DBFF	20	14		1K BLOCK; ENABLED ON $\overline{READ}$ PORTION
D800	DBFF	20	10		1K BLOCK; ENABLED ON $\overline{WRITE}$ PORTION
DC00	DFFF	20	15		1K BLOCK; ENABLES KEYBOARD ON $\overline{READ}$
DC00	DFFF	20	11		1K BLOCK; ENABLES KEYBOARD ON $\overline{WRITE}$
E000	FFFF	23	7		8K BLOCK; ENABLES ACIA AND MONITOR ROM DECODING
F000	F0FF	17	12		256 BYTES; $\overline{ACS}$—ENABLES ACIA ASSUMING A11, A10 = 0
F100	F1FF	17	11		256 BYTES; ASSUMING A11, A10 = 0
F200	F2FF	17	10		256 BYTES; ASSUMING A11, A10 = 0
F300	F3FF	17	9		256 BYTES; ASSUMING A11, A10 = 0
F400	F4FF	17	12		256 BYTES; ASSUMING A11 = 0, A10 = 1
F500	F5FF	17	11		256 BYTES; ASSUMING A11 = 0, A10 = 1
F600	F6FF	17	10		256 BYTES; ASSUMING A11 = 0, A10 = 1
F700	F7FF	17	9		256 BYTES; ASSUMING A11 = 0, A10 = 1
F800	FFFF	19	6		2K; $\overline{MCS}$—ENABLES MONITOR ROM

makes string input difficult. The method I prefer is a very simple subroutine which prevents the Basic Interpreter from returning to command mode on a null INPUT.

The input buffer is in zero page, 13H to 59H, and is terminated by 00H. If Return is pressed in response to INPUT then 00H is stored in 13H. This is the flag for return to command mode in the INPUT routines. Most of the INPUT subroutines are within ROM, and not directly modifiable. However there is a vectored jump to output the characters to the VDU. The routine is corruptible at this point.

The subroutine, for the new Monitor ROM, is:

```
0230   48          PHA
0231   A5  13      LDAZ $13
0233   D0  06      BNE $023B
0235   85  14      STAZ $14
0237   A9  60      LDAIM $60
0239   85  13      STAZ $13
023B   68          PLA
023C   4C  D4  FB  JMP $FBD4
```

This can either be entered directly using the Monitor, or POKEd by a BASIC routine at the beginning of the program. It is protected from Reset, and does not need memory sizing.

To be initialised, the output vector in 021AH,021BH needs to be changed from FBD4H to 023OH. Again this can be using Monitor, or POKE538,48:POKE539,2 Reset restores the usual vector so POKE is preferable.

The 60H in 0238 is the character INPUT by pressing Return in response to an INPUT statement. In my UK101 character set ASCII 60H prints a space but can be distinguished from 20H, the usual ASCII space. For numeric inputs this will produce a REDO error, and should be replaced by 30H. Pressing Return will then INPUT zero.

A slight modification to this subroutine enables a program to demand a password, and for the password to be entered without printing to the screen. The modification is that the accumulator is not pushed to stack, and the pull is replaced by a load accumulator instruction. Whatever is loaded will be printed instead of the typed characters.

This modified routine should be called only for the password INPUT, as in the following example:

```
05 POKE1,0:POKE2,0:POKE 530,1
10 DATA..... { To POKE subroutine
20
30
50 PRINT CHR$(12)
60 PRINT "PASSWORD?";
70 POKE538,48 : POKE539,2
80 INPUT P$
90 POKE538,212 : POKE539,251
100 IF P$ = "UK101" THEN 150
110 T = T + 1:IF T = 2 THEN PRINT
    "PROG. ABORTED AND
    ERASED" : NEW
120 GOTO 60
150 PRINT "ACCEPTED"
160 REM BODY OF PROGRAM
```

This gives you two chances to give the password—in this case UK101—and if you fail erases the program. It prevents the use of Return and then LIST to discover the password, Reset, (W)arm Start, LIST is jammed by line 05, as is Control C. It still allows entry to the Monitor, but protecting against this is probably excessive for most programs likely to run on the UK101.

Bob Potter,
Oxford.

DEVELOPMENT TOOL?

Sir—In the July issue of Practical Electronics *you published two programs, a variable save and a screen editor program, although it is possible to load both these programs into memory it is not possible to run them both at the same time. This is an obvious disadvantage as they are both essential aids to program development, and one without the other is a bit of a hindrance to program development. I, having only just started to unravel the mysteries of machine code and not knowing an awful lot about it, was wondering if it is possible to link these two programs together to produce a very useful program development tool. I am hence writing to you to request your help, or that of your readers, to enable the two programs to be executed as one.*

Johann ckh Riedel,
Solihull.

GREMLINS

A compulsive game has been submitted by *Mr. J. Semple* of Jesus College, Oxford. We are told that 60 is a good score, and 89 is the highest so far. We added sound just for fun, and this is underlined for those who have no sound board. Just leave out the indicated instructions for silent suffering. Note that the sound unit is assumed to be located at 61808 and 61809 decimal (R1 & R2 in Line 10). Only line 10 need be changed to re-vector for the Sound Board if located elsewhere.

```
10 S1=61808:S2=S1+1:INPUT(11)
20 FORI=12016:PRINT:NEXT
30 INPUT"DO YOU WANT INSTRUCTIONS ";A$
40 IFLEFT$(A$,1)="Y"THENGOSUB5010
45 PRINT:INPUT"ENTER GAME SPEED 0-10 ";Z:Z=ABS(Z)
47 FORI=1TO200:NEXT
50 K=57088:POKE530,1:POKE,0
80 GOSUB1000
100 GOSUB3000
110 POKE,239:C=PEEK(K)
120 IFC=247THEN90=-64:GOSUB2010
130 IFC=127THENPOKE530,0:GOTO4000
140 POKE,247:C=PEEK(K)
150 IFC=239THEN90=-1:GOSUB2010
160 IFC=247THEN90=1:GOSUB2010
170 POKE,251:C=PEEK(K)
180 IFC=239THEN90=64:GOSUB2010
190 GOTO100
1000 FORI=1TO16
1010 PRINT:POKE54219,181:POKE54265,161
1020 NEXT
1030 P=53457:POKEP,6:LL=32
1040 S=0:S1=0
1050 F=66:POKE53280,F+32
1060 POKE54220,32
1070 POKE53282,48:POKE53283,48
1080 POKE53456,65:POKE53490,66
1090 POKE54032,68:POKE54066,67
1100 RETURN
2010 LK=PEEK(P+D)
2015 IFLK>64ANDLK<69THEN2025
2020 IFLK<>32THENGOSUB5100:GOTO2100
2025 POKEP,LL
2030 P=P+D
2035 LL=LK
2040 POKEP,6
2050 IFLL<>FTHEN2100
2060 S=S+1:F=F+1:POKEP,188:GOSUB5150:IFF=69THENF=65
2070 POKE53280,F+32
2080 IFS=10THENS=0:S1=S1+1
2090 POKE53282,S1+48:POKE53283,S+48
2100 FOKI=1TO100-10*Z:NEXT
2110 RETURN
3000 CO=CO+1
3010 IFCO<>10THENRETURN
3020 CO=0
3030 PP=53259+46*RND(89)+64*INT(16*RND(76))
3040 IFPEEK(PP)<>32THEN3020
3050 POKEPP,187
3060 RETURN
4000 PRINT:PRINT:PRINTTAB(15);"GREMLIN"
4010 FORI=1T99:PRINT:NEXT
4020 U=10*S1+S
4030 PRINT"  YOUR SCORE = ";U
4040 IFUU>STHENU=U
4050 PRINT:PRINT" THE HIGHEST SCORE SO FAR = ";U
4060 PRINT:PRINT:INPUT"DO YOU WANT TO PLAY AGAIN ";A$
4070 IFLEFT$(A$,1)="N"THENEND
4080 PRINT
4090 GOTO30
5010 PRINT"YOU (";CHR$(6)") MUST CIRCULATE THUS:"
5015 PRINT"A..B..C..D..A..B..C  etc."
5020 PRINT"SCORING ONE POINT AT EACH LETTER."
5025 PRINT"...BUT THE GREMLINS WILL OBSTRUCT YOU"
5030 PRINT"WITH INCREASING NUMBERS!"
5035 PRINT
5040 PRINT"MOVE ";CHR$(6);"          UP KEY E"
5050 PRINT"                  DOWN   B"
5060 PRINT"                  LEFT   C"
5070 PRINT"                  RIGHT  4"
5080 PRINT"    KEY W FOR RESCUE"
5090 PRINT:PRINT:PRINT:RETURN
5100 F(1)=INT(RND(2)*255)+1:F(6)=INT(RND(2)*30)+1
5115 F(7)=9:F(8)=30:F(12)=INT(RND(2)*25)+5:F(13)=9
5120 POKEP,233:GOTO6000
5150 F(2)=Z*10:F(7)=-3:F(9)=30:F(11)=250:F(12)=5
5155 F(13)=4:GOTO6000
6000 FORI=0TO13:POKER1,I:POKER2,F(I)
6010 IFI=7THENPOKER2,255-F(I)
6020 NEXT:RETURN
6100 FORI=0TO13:F(I)=0:IFI=7THENF(I)=255
6110 NEXT:GOSUB6000:RETURN
```

TRACE

Sir—When your UK101 BASIC program starts doing things you didn't intend—or vice versa—then the program "Trace" will help to keep an eye on it. When activated, the program continuously monitors the BASIC interpreter and displays the current line number in brackets at the top of the VDU.

The program is small enough to fit into the free RAM at 546 decimal but equally well it can be loaded at the top of memory in the space protected by the "MEMORY SIZE?" request (its size is 87 decimal bytes).

To load Trace simply run the following BASIC program which can be deleted from memory after use if required. It will ask you where to load Trace and your answer will probably be 546 (free RAM), 4008 (top of 4K) or 8104 (top of 8K). It then prints out two bits of information. The first is a command line which will activate Trace when typed in immediate mode (eg: POKE 11,34:POKE 12,2:X=USR(X)) and the second is a location which can be POKEd at any time in immediate mode with a delay count (1 to 255) allowing the running speed of your program being "traced" to be controlled. This location is set to 3 by default but setting it larger will slow your program down and allow you to keep track of the statement numbers on the screen more easily.

```
62999 REM ** TRACE LOADER. P.BECKETT NOV 80 **
63000 DATA169,0,141,28,2,169,0,141,29,2,96
63001 DATA165,136,201,255,208,3,76,155,255,133
63002 DATA173,165,135,134,174,162,144,56,32
63003 DATA232,183,169,91,141,16,208,169,93,141
63004 DATA22,208,32,110,185,162,0,189,1,1,240
63005 DATA10,157,17,208,232,224,5,240,12,208
63006 DATA241,169,32,157,17,208,232,224,5,208
63007 DATA248,162,0,240,3,32,129,254,232,224
63008 DATA3,208,248,76,155,255
63009 INPUT"Type load address";A
63010 FORI=A1DA+86:READX:POKEI,X:NEXT
63011 S=A+11
63012 S1=INT(S/256)
63013 S2=S-S1*256
63014 POKEA+1,82:POKEA+6,S1
63015 PRINT"For TRACE startup:"
63016 A1=INT(A/256):A2=A-A1*256
63017 PRINT"POKE11,";A2;":POKE12,";A1;":X=USR(X)"
63018 PRINT"TRACE delay count=";A+81
```

Incidentally, the Trace feature is de-activated every time a warmstart is executed. The top of the VDU was used for the printout rather than the bottom so that line-feeds which may be produced by your program don't cause the screen to be obliterated by a column of numbers.

Trace uses routines in the BASIC ROMs as well as the cassette input routine at FE80 as a delay. It was written for the standard UK101 but using the assembler listing (top right), modifications can easily be made if required.

Mr. P. Beckett,
Blackpool.

```
        LDA   #$2D     A9  2D
        STA   $021C    8D  1C  02   } SUBROUTINE TO SET UP
        LDA   #$02     A9  02       } CTRL C VECTOR TO POINT
        STA   $021D    8D  1D  02   } TO THE TRACE ROUTINE
        RTS            60           } BELOW

        LDA   $88      A5  88       }
        CMP   #$FF     C9  FF       } IS THE MACHINE IN
        BNE   A        D0  03       } IMMEDIATE MODE?
        JMP   $FF9B    4C  9B  FF   YES—GO TO CTRL C ROUTINE
A       STA   $AD      85  AD       }
        LDX   $87      A6  87       } NO—STORE CURRENT LINE NO.
        STX   $AE      86  AE       } IN AD/AE
        LDX   #$90     A2  90       }
        SEC            38           } CONVERT INTEGER AT AD/AE
        JSR   $B7E8    20  E8  B7   } TO FLOATING POINT AT
                                    } AC/AD/AE/AF
        LDA   #$5B     A9  5B       }
        STA   $D010    8D  10  D0   } DRAW LEFT AND RIGHT HAND
        LDA   #$5D     A9  5D       } SQUARE BRACKETS ON SCREEN
        STA   $D016    8D  16  D0   }
        JSR   $B96E    20  6E  B9   CONVERT TO ASCII AT 101
        LDX   #$00     A2  00       }
B       LDA   $0101,X  BD  01  01   }
        BEQ   C        F0  0A       }
        STA   $D011,X  9D  11  D0   } TRANSFER 5 CHARACTERS
        INX            E8           } FROM 101 ONWARDS TO
        CPX   #$05     E0  05       } SCREEN. IF A ZERO IS FOUND
        BEQ   E        F0  0C       } GO TO C
        BNE   B        D0  F1       }
C       LDA   #$20     A9  20       }
D       STA   $D011,X  9D  11  D0   }
        INX            E8           } FILL UP REMAINING
        CMP   #$05     E0  05       } CHARACTERS WITH SPACES
        BNE   D        D0  F8       }
        LDX   #$00     A2  00       SET UP COUNTER
        BEQ   E        F0  03       }
F       JSR   $FE80    20  80  FE   } DELAY ROUTINE
E       INX            E8           } LOOP BACK UNTIL COUNT
        CPX   #$03     E0  03       } REACHED
        BNE   F        D0  F8       }
        JMP   $FF9B    4C  9B  FF   GO TO CTRL C ROUTINE
```

UK101 IMPROVED DISPLAY

One minor fault with the UK101 when compared with more expensive computers is the interference produced on the TV screen whenever anything is printed or POKEd into the display RAM.

In the UK101, a monostable switches off the video signal during VDU access to prevent spurious noise and so small parts of the signal don't reach the screen.

In white parts of the display, these appear as dark lines and interference is especially bad during animated games or listings.

This simple modification which doesn't require any extra IC s allows the VDU RAM to be accessed at full speed without any noise at all on the TV screen.

The modification is as follows:

Remove the ICs numbered below, bend out the pins indicated and re-insert them.

IC no 28 bend out pin 9
IC no 42 bend out pin 2
IC no 55 bend out pin 11
IC no 56 bend out pin 6
IC no 69 bend out pin 1

Now make the following connections on the front of the board with insulated wire

Pin 1 of IC 55 to pin 3 of IC 8 (the CPU)
Pin 9 of IC 28 to pin 3 of IC 8
Pin 6 of IC 56 to pin 11 of IC 55
Pin 1 of IC 58 to pin 2 of IC 42
Pin 1 of IC 69 to +5 volts

The hardware adjustments made above are:

1. The address/control selectors, ICs 53,54,55 are switched by 01.
2. The monostable (IC28) producing the load signal for the shift register is triggered by 01.
3. The shift register receives its clock signal from the non-inverted 8 MHz signal.
4. The VDU RAM is selected by IC 56.
5. The blanking monostable is disabled.

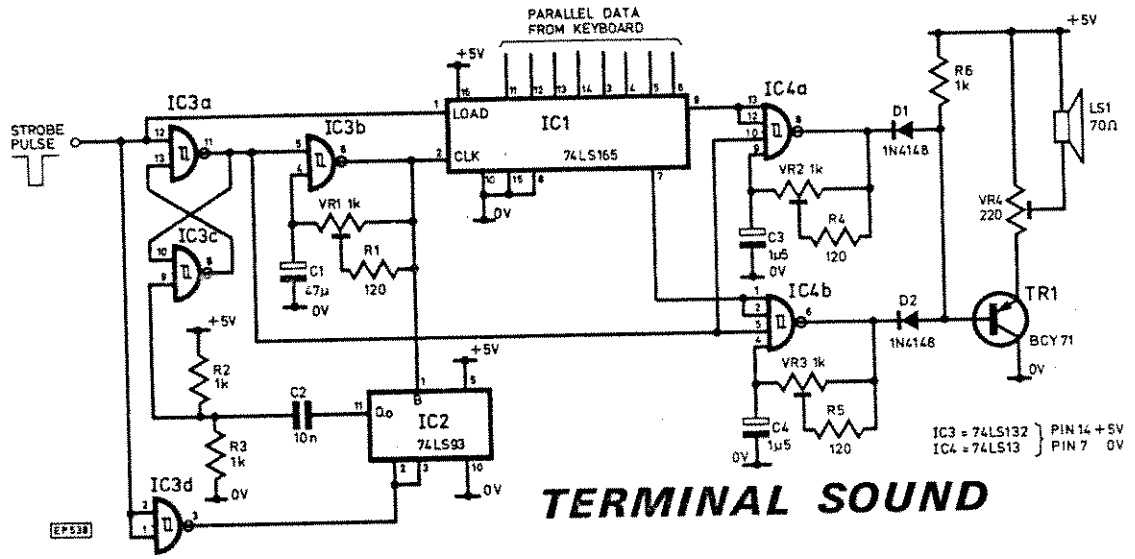
When these alterations are made, the VDU circuitry is synchronised with the 01 signal from the CPU so that the display RAM is accessed by the VDU during one phase of 01 and the CPU accesses it during the other.

This modification has worked well on my UK101 and below is a small program, which, if run before and after the alteration, shows its effectiveness.

Note CHR$(161) is a white block.

```
10 FORI=1TO47:A$=A$+
   CHR$(161):NEXT
20 PRINT A$:GOTO 20
```

Ian Bradbury



TERMINAL SOUND

INTENDED as an addition to microprocessor or VDU keyboards, this produces a different pattern of high and low tones for each key on the keyboard. Although it is not necessary to learn these different patterns, after some use the tone generator becomes a real aid to accurate typing.

The 8-bit data from the keyboard is loaded into a shift register IC1, and latch IC3a,c is set, when a keyboard strobe pulse is produced. Pulses from the oscil lator IC3b shifts the data so that the two tone oscillator IC4a,b produce a high tone for a 1 and a low tone for a 0. These tones are summed and fed to a small loudspeaker. Counter IC2 resets the latch after the last data bit is shifted out.

VR1 sets the speed of shifting, and VR2/VR3 adjust the high and low tones respectively. VR4 sets the volume. The circuit shown is for negative strobe pulses, but is easily rearranged for positive strobes.

T. P. Hopkins,
Didsbury,
Manchester.

NINE-CHIME DOOR BELL

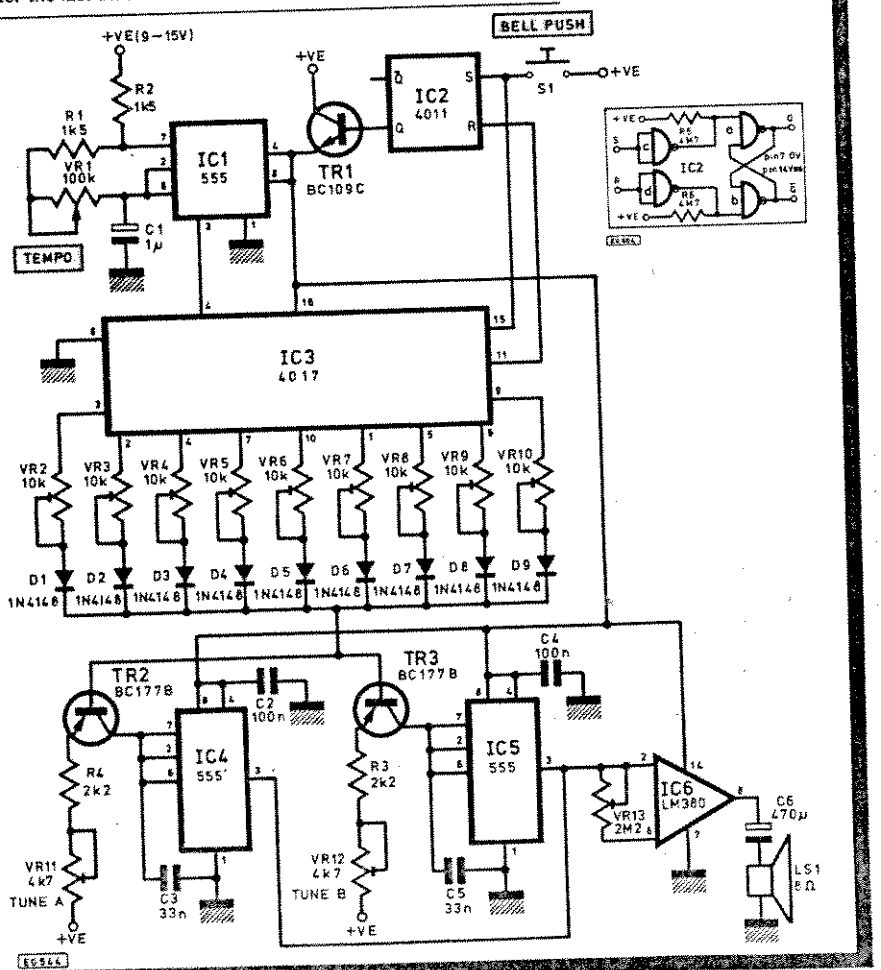
THIS bell is programmed by the user to play a tune with anything up to nine notes. Instead of one oscillator playing the tune, two oscillators are used, which are tuned so as to give a chord on every note.

When the bell push is operated, the latch formed by IC2 is triggered and the decade counter IC3 is reset. The output of the latch operates a switching transistor. This applies the supply voltage to ICs 1, 3, 4, 5, and 6. At this point, a slow oscillator formed by IC1 is started, the output of which starts the counter counting. A tempo control is provided which speeds up or slows down the tune.

Each output of the counter goes via a preset and diode which are commoned and the output voltage from each note (which is programmed by adjusting the pre-sets), goes to two voltage controlled oscillators formed by IC4 and 5. Each VCO has its individually controllable tune preset. In practice these are set up so the notes produced form a melodic chord. The output from each VCO goes to a simple power amplifier formed by IC6. This is provided with a simple volume control which feeds a loudspeaker.

When the last note has played the counter is automatically reset and the latch also resets. The power is removed from all the i.c.s except IC2, which is always receiving power so the door chime may be triggered.

Graeme Durant,
Selby,
N. Yorkshire.



MICRO PROMPT

The hardware and software exchange point for PE computer projects

LISTLESS SOFTWARE

A method for protecting your BASIC from being LISTed by a spy has been sent in by *Mr. Mistry* of Bradford.

Add line zero to your program; which might be: **0 REM**—*some program*—for example.

Then **POKE 769, 0** in Command Mode.

Any alteration to the listing, after this, will crash the system.

For return to normal, **POKE 769, 7**

PIECES OF EIGHT

Sir — I have decoded the memory block 2000–3FFF on my UK101 in the following way, while I use the decoding for I/O and an EPROM programmer, they could be used for other things:

IC23 (74LS138) seems to decode the whole 64K into 8K blocks most of which are not used. The unused pins can just be bent out and used to decode the 8K blocks, ie. pin 14 decodes 2000–3FFF
13 decodes 4000–5FFF
12 decodes 6000–7FFF
11 decodes 8000–9FFF
remembering that 1K EPROM, for example, would appear more than once in the 8K block selected.

To decode 2000 to 3FF into 8 × 1K blocks I soldered another 74LS138 onto IC22.
PIN 4 connects to Pin 14 (Y1) of IC23
7 (Y7) decodes 2000–23FF
9 (Y6) decodes 2400–27FF
10 (Y5) decodes 2800–2BFF
11 (Y4) decodes 2C00–2FFF
12 (Y3) decodes 3000–33FF
13 (Y2) decodes 3400–37FF
14 (Y1) decodes 3800–3BFF
15 (Y0) decodes 3C00–3FFF
I have transferred the Compukit Screen Edit tape to EPROM which runs at 2000–23FF, but does anyone know how to transfer the extended monitor?

J. Walton,
Newton,
Derbyshire.

It should be emphasised that material presented in Prompt has not necessarily been proven by us. Neither can compatability with *all generations* of the computer equipment to which it relates be guaranteed.

Software and hardware designs submitted should be accompanied by a declaration to the effect that it is the original work of the undersigned, and that it has not been accepted for publication elsewhere.

MADE FOR EACH OTHER

Sir—The Transam Triton microcomputer's on board memory ends at address 1FFF. This makes interfacing to Dr. Berk's EPROM programmer extremely simple, provided that it is the only off board memory in use. A0 to A10 from the EPROM board are connected to the corresponding address lines from the Tritom expansion socket and the board enable is connected to the Triton A13, which only goes high for addresses over 1FFF Hex. A10 goes high every time an address containing X4XX is accessed (X = don't care), but the board is not enabled until A13 is active, therefore A10 only becomes effective when we reach 2400, which is just what we want and locates the EPROM block directly following the RAM block. This allows both RAM and EPROM to be accessed under program control, and the Triton's monitor will accurately locate the end of the new RAM with its memory check procedure, which is needed for the correct operation of the basic interpreter.

Obviously A13 and A10 will also become active for addresses further up the map, but if the EPROM board is the only off board memory expansion, then higher addresses should never be accessed except under error conditions, in which case the RAM might be interfered with, but then, that would probably happen anyway under error conditions. Any other memory expansion would no doubt use the Triton motherboard which changes the problem completely.

Triton Socket	*EPROM Board*
MEMW	*R/W*
Ground	*0 volts*
Five volts not available from socket—wire to regulator	
A13	*Pin 6, IC10 (enable)*
A0	*A0*
—	—
—	—
A10	*A10*
D0	*D0*
—	—
—	—
D7	*D7*

One change must be made to the board, due to the fact that we are using a positive going address line rather than a zero going decode line for the ENABLE. We therefore leave out the gate 1C8A which merely inverts the ENABLE, and connect the Triton's A13 to pin 6 of IC10, this is easily done as the track on the top of the EPROM board nearest the l.e.d. is the track which connects the two. Cut the track or simply leave out the through board pin nearest the l.e.d., and connect the A13 signal to this track. Both pins 1 and 2 of 1C8 should then be connected via link L10 to +5 volts as it is bad practice to leave t.t.l. inputs floating.

The redundant gate (IC8A) can be put to use to give a very useful added facility.

As we are allowing the computer to select between the RAM and the EPROM by the use of the address line A10 rather than using a switch and doing it manually, we are getting the best use of the extra memory available. However, we are unable to try routines in RAM before burning them into EPROM as the computer sees them as two separate blocks of memory and internal calls or jumps will not work in both blocks (not with 8080 direct addressing). This can be overcome by using the redundant gate to invert A10 and selecting either the inverted A10 or the non-inverted A10 with a single pole changeover switch (Fig. 1). This effectively swaps the positions of the RAM and EPROM as far as the computer is concerned. Therefore a program can be developed and debugged in RAM at addresses between 2400 (Hex) and 27FF (Hex) which is normally the EPROM's address, then it can be burnt in and run in EPROM after the switch is returned to normal. All this without losing the advantage of simultaneous use of both blocks of memory.

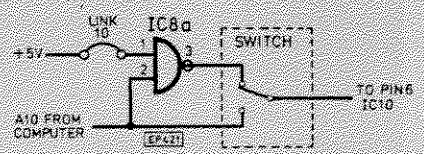


Fig. 1. Address inversion switch

Note that this is not suitable where a decode line has been used for interfacing, as the gate IC8A is already in use.

Iolo Davidson,
Hawling,
Gloucestershire

PREMIER SOFTWARE FOR UK101

Two software cassettes are available from *Premier Publications* of 12 Kingscote Road, Addiscombe, Surrey.

The first is called "Strategy Games Pack", and is Superboard compatible. It contains three well presented and compulsive games: Nine-In-A-Line, Square Solitaire, and Executive Jigsaw. The start of the tape loads in some utility machine code software to support these games.

The second tape is called "Utilities Pack" and comprises a range of subroutines which can be called up by the user's own main program, after which, any unused utility routines are removed.

A subroutine is included for screen location identification via a grid system. Another routine provides a precision random number generator with more linear distribution. There is also a "read data" routine which overcomes the need for a FOR-NEXT loop to find a particular datum. When GOSUB 30 is called, the piece of data is returned as Z$.

There is a kind of direct telewriting subroutine, a routine for driving the cursor around the screen, and much more useful software.

Premier publications: ✆ 01-656 6156.

MICRO PROMPT

The hardware and software exchange point for PE computer projects

IT DOES BOTH!

Sir—Without an assembler, the easiest way to store machine language programs is as a BASIC program of DATA statements, and poke them into the correct memory locations. Unfortunately this requires the conversion of hex instructions into decimal numbers and the writing of the BASIC program. The HECDEC program does both!

Simply enter the instructions as pairs of hex digits separated by Return and enter "X" to mark the end of the data. The BASIC program will then be sent to the cassette. Then LOAD the program back and add any necessary POKE instructions.

Comments on the program:
Line 30 set the dimension of B as required.
Line 40 two hex digits *must* be entered.
Line 70 X terminates the data entry.
Line 160 six DATA items per line.
Line 200 an option to retransmit the program.
RETURN exits.

No check is made for illegal data entry, this could be added if thought necessary.

```
10  REM HECDEC PROGRAM
20  REM NOEL CAFFREY JAN
    1981
30  K= 1: DIM B(2ØØ)
40  INPUT "HEX DATA . . .";A$
50  I =1: N = Ø
60  S$ = MID$ (A$,1,1) : C =
    ASC(S$)
70  IF S$ = "X" THEN GOTO 13Ø
75  IF LEN(A$)><2 THEN PRINT"
    ERROR . . . RE-ENTER":
    GOTO 40
80  IF C< = 57 THEN N = N +
    C–48
90  IF C>57 THEN N= N + C–55
100 IF I= 2 THEN GOTO 12Ø
110 I= 2: N=N*16: GOTO 6Ø
120 B(K) = N: K = K + 1: GOTO 4Ø
130 PRINT ":TURN ON TAPE
    RECORDER": SAVE
140 FOR I = 1TO 1ØØØØ: NEXT
150 L = 1ØØØ: PRINT L; "DATA"; :
    FOR I = 1 TO K–1: PRINT B(I);
160 IF I/6 <>INT(I/6)THEN GOTO
    18Ø
170 L = L + 1ØØ: PRINT: PRINT L;
    "DATA";: GOTO 19Ø
180 PRINT ",";
190 NEXT
200 PRINT: INPUT":ANOTHER
    COPY"; W$
210 GOTO 15Ø
```

Noel Caffrey,
Castleknock,
Co. Dublin.

SCROLL CONTROL

Sir—While writing a program to convert hexadecimal to decimal and vice-versa for my UK101, it became apparent that it was not possible in BASIC to prevent the screen rolling up and leaving the input message on the screen. (At least, this could not be done within a reasonable time.) As this was required, I wrote the following program which enables one to leave the input message on the lowest line of the screen and only the resulting output is rolled up. It does however require a machine code routine. The accompanying program shows how the procedure works and uses the trivial (but short!) example of finding a square-root. Some of your readers may find this useful, especially for "conversion-type" programs; the output is very neat and uncluttered.

The routine as given works on an 8k machine, memory should be restricted to 8100 bytes at start up. The machine code is stored at 1FA4 to 1FC6. For a 4K machine lines 35 and 400 should be altered to read:—

```
35 POKE 11, 210: POKE 12, 15
400 DATA 4050, 35
```

The machine code then starts at 0FD2, memory should be restricted to 4050.

In passing it may be worth noting that in my hex. to dec. program I had to ensure that any input was valid by inputting it into a DIM field, this prevents any "REDO" messages being put out by the interpreter which would spoil the effect.

```
10  REM NEAT OUTPUT ROUTINE
20  REM BY S.F.J.

30  GOTO 50
35  POKE 11, 164: POKE 12, 31
40  LET X = USR(X)
45  RETURN

50  READ SP, N1
60  FOR I = 1 TO N1
70  READ RE
80  POKE SP, RE
85  LET SP = SP + 1
90  NEXT
100 FOR I = 1 TO 15: PRINT: NEXT
110 INPUT "INPUT NUMBER"; N
120 IF N < = 0 THEN 300
130 LET S = SQR(N)
140 GOSUB 35
150 PRINT TAB(10); N; TAB(25); S
160 PRINT TAB(10);
170 FOR I = 1 TO 25
180 PRINT CHR$(132);
190 NEXT
200 PRINT
210 GOTO 110
300 GOSUB 35
310 PRINT TAB(10); N; TAB(26);
    "INVALID!"
320 GOTO 160
400 DATA 8100, 35
410 DATA 169, 0, 170, 169, 64
420 DATA 168, 169, 191, 133, 225
430 DATA 169, 211, 133, 226, 161
440 DATA 225, 145, 225, 198, 225
450 DATA 169, 255, 197, 225, 208
460 DATA 244, 198, 226, 169, 207
470 DATA 197, 226, 208, 236, 96
```

Stephen Jaworski,
Walsall.

POOLS AID

On a light hearted note, here is a program which turns your UK101 into a punter's aid. Beats a blindfold and pin!

```
10  INPUT "HOW MANY
    MATCHES ARE ON THE
    COUPON"; U
20  INPUT "HOW MANY LINES DO
    YOU WISH TO ENTER"; V
30  INPUT "HOW MANY
    MATCHES TO BE CHOSEN PER
    LINE"; W
40  DIMA(W); L=INT
    (W*RND(1)+V)
50  FOR T=1TOL: PRINTTAB(19)
    "THINKS": FORX=1TOW
60  B=INT(U*RND(1)+1): FORY=
    1TOW: IFA(Y)=BTHEN60
70  NEXT: A(X)=B: NEXT: FORZ=
    1TOW: FORQ=1TOW
80  IFA(Q)<A(Q–1)THENS=
    A(Q): A(Q)=A(Q–1): A(Q–1)=S
90  NEXT: NEXT: FORY=1TOW:
    1FT>L–VTHENPRINTA(Y);
100 IFY=WTHENPRINT
110 NEXT: NEXT: PRINTTAB(30)
    "GOOD LUCK": END
```

It will be noticed that an inefficient ranking method is used at 70–80. This adds to the suspense by slowing the execution considerably, so that a typical run may take about a minute.

The writer will be happy to accept 0·01% commission from winners of half-a-million or so.

R. J. Newman,
Chesham,
Bucks.

WARM START INTERRUPT

Sir—Concerning Mr. N. Climpson's Screen Editor on tape, from the July 1980 edition of Practical Electronics, I am sure I am not the only one who becomes infuriated when POKE 536, 45:POKE537,31 has to be typed in every time the machine is warm-started, to re-activate the editor. Hence I have designed a very simple machine code routine, which will re-activate the editor automatically after a warm start:

```
5ØØ76 DATA 169, 45, 141, 24, 2, 169, 31,
      141, 25, 2, 76, 116, 162
5ØØ78 FOR I=546 TO 558:READ
      A:POKEI, A:NEXT
5ØØ8Ø POKEØ1, 34: POKE 2,2
5ØØ82 NEW
```

(This should be added on to the end of the editor, and saved as a whole.)

VARIABLE INTENSITY

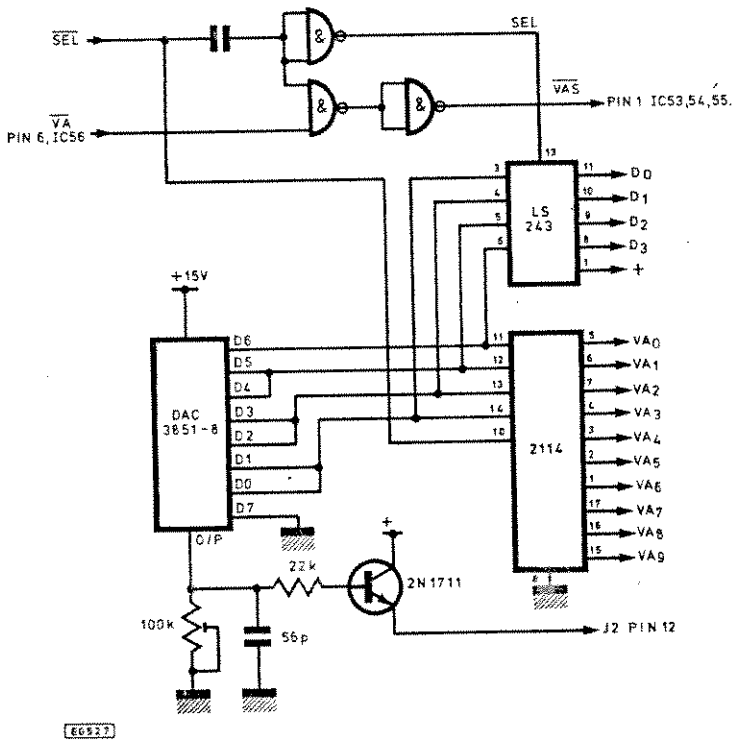
The tone control occupies 1K of memory between D400 and D800 (just after the video refresh memory). To save space and money the unit uses the address decoding of the UK101. A $\overline{SEL}$ signal is derived from pin 9, IC20, which decodes A10, A11 and A12, A13, A14, and A15 (via IC23). The unit also uses the video address bus (VA0→VA9) and is therefore wired to the compukit board and not taken via the expansion sockets.

$\overline{SEL}$ takes the 2114s $\overline{WR}$ low, enabling data to be written into the 2114 through an LS243 (tristate buffer) which is enabled by SEL. $\overline{SEL}$ is also required to take control of the VA bus from the binary counters. This is also the function of $\overline{VA}$ (pin 6, IC56) for the video refresh memory. $\overline{VA}$ and $\overline{SEL}$ are therefore decoded and a new signal $\overline{VAS}$ (video address select) is formed. The p.c.b. track to pin 6, IC56 must be cut for this function.

In normal operation the binary counters address the 2114 through the VA bus. The data corresponding to the location's intensity then appear on the 2114's data bus. This bus is directly connected to the D-to-A converter (non latching). NB: The DAC 3851 used gives a current output, and so a trimpot is used to obtain the correct voltage. Some modification may be required for D/A converters with voltage outputs.

A point worth noting about the Compukit's video circuitry is that reverse characters, i.e. black on white, is available from pins 4 and 5 of IC70; this is much clearer for test than the standard white on black display. It is well worth having this feature switch-selectable or even programmable by 'POKEing'.

R. Mark Charles.



Extra Cases!

30p (plus postage) To Regular Readers.

FOLLOWING the Free Case gift with the May issue we have a quantity of cases available as extras to regular readers (at UK and BFPO addresses only) for 30p plus postage—the normal price from Lascar is £1.95 plus VAT and postage.

Extra cases—as used to house the Light, Noise and Capacitance Meters published in this issue—can be obtained from PE by sending in two correctly filled-in case coupons (name and address on both coupons please) **cut from PE,** a 30p postal order and a 15½p stamp (a case coupon is given here, another will be given next month). **We can only accept 30p postal or money orders** (NOT CURRENCY OR CHEQUES). Do not enclose any correspondence. **Incorrect or incomplete orders will not be accepted**—your stamp will be used to return them.

Send two completed case coupons cut from PE with your 30p postal order (payable to IPC Magazines Ltd), and a 15½p stamp to:

PE Instrument Case Offer, IPC Magazines Ltd., Westover House, West Quay Road, Poole, Dorset. BH15 1JG.

Sorry we cannot undertake to send extra cases to overseas readers.

PLEASE ALLOW 28 DAYS FOR DELIVERY.

PE CASE COUPON

I enclose two completed case coupons cut from PE, a 30p postal order made payable to IPC MAGAZINES LTD., and a 15½p stamp. Please send me an extra instrument case.

NAME .....

ADDRESS .....

.....

.....

.....

Only available to UK and BFPO addresses, allow 28 days for delivery. Incorrect or incomplete orders will be returned.

MICRO PROMPT

The hardware and software exchange point for PE computer projects

MAKE THE MOST OF USR

Sir—Although readers will know that Y = USR(X) transfers control from BASIC to a machine code routine on the UK101, many may not realise that parameters may also be passed back and forth between BASIC and machine code using this function.

Executing Y = USR(X) puts the value of X in the floating accumulator. On returning to BASIC with an RTS, the number in the floating accumulator is assigned to Y. Therefore, by changing the contents of the floating accumulator before returning to BASIC, a new function can be implemented.

The floating accumulator is a group of zero page locations at $AC to $BO. X is put there as a floating point number:

$ AC Exponent
$ AD-$AF Mantissa
$ BO Sign

Although this is useful for implementing certain functions—for instance new mathematical functions—most applications require values to be passed from BASIC to machine code as two byte hex numbers. Whereas this can be achieved by POKING the number into a memory location before entering the m.c. program, transferring numbers above 255 becomes messy and slow. Luckily a subroutine resides at $AE01 which converts the floating point number in the floating accumulator to a two byte hex number. This number is returned in $AE (high byte) and $AF (low byte). Hence, to transfer X to a machine code program as a two byte number in $AE. $AF:

POKE 11, 34: POKE 12, 2: X= USR(X)
Location

Machine Code:

Ø222	*JSR $AEØ1*	*2Ø Ø1 AE*
	Main routine	
	RTS	*6Ø*

Unfortunately, the subroutine at $AEØ1 jumps to BASIC if X is larger than 2^{15} (32768) with a "function call" error. One of the main uses of the USR(X) function would be in passing video RAM addresses to machine code routines to provide fast dynamic screen layouts, such as in LIFE and Space Invaders type games. The video RAM is at DØØØ—D3FF (53248—54271), and so the above problem must be overcome. The simplest solution is to subtract 53248 from X before X = USR(X), then to add it on again in the machine code program, as shown below.

```
100 REM TRANSFER VDU PIXEL ADDRESS TO A MACHINE
200 REM CODE ROUTINE. X= ADDRESS
300 A = 53248: POKE 11, 34: POKE 12, 02
400 . . . . . Main Program. X defined here
.
.
.
1000 X = USR (X – A)
1100 REM (PIXEL ADDRESS—DØØØ) NOW IN FLOATING ACCUMULATOR.
LOCATION

Ø222 JSR $AEØ1 2Ø Ø1 AE;   Convert to Double Byte
CLC          18
LDA $AE      A5 AE;        Add $DØ to high byte
ADC $ DØ     69 DØ
STA $ AE     85 AE;        Replace in $ AE
.                          Result hi $AE
.
.
Main program                lo $AF
RTS          6Ø;           Video address now in $AF, $AE.
```

T.D. Allen,
Poole,
Dorset.

AUTO LINE NUMBER

Sir—I enclose a listing of a program I have written which you might like to print in your Micro Prompt series.

This program automatically outputs the BASIC line numbers, followed by a space. This is done by pressing Control–N instead of Return, at the end of each line. The program as printed uses a start line of 100, with an increment of 10. These can easily be changed by suitable POKEs. The program is loaded via BASIC into the "free RAM" area of the UK101. The current line number and the increment are each stored as three pairs of BCD digits, at locations 570–575 (023A–023F hex). Six zero page locations are used for two indirect pointers, and two temporary stores at addresses 240–245 (00F0–00F5).

If the new CEGMON monitor is used line 9110 should be changed from ....186, 255,201,14 to ....70,251,201,14.

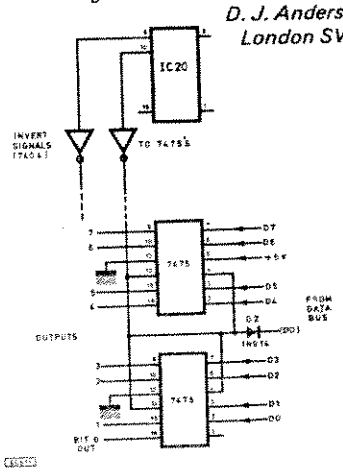
A. Scott,
Ashtead,
Surrey.

```
9000 REM  AUTO-NUMBER
9010 REM  BY A. SCOTT
9020 REM  19/11/80
9030 FORI = 570TO667
9040 READJ:POKEI,J:NEXT
9050 POKE240,58: POKE241,2
9060 POKE242,61:POKE243,2
9070 POKE536,64:POKE537,2
9080 ?"AUTO-NUMBER READY":?
9090 ?"PRESS CTRL-N TO OUTPUT LINE NUMBER"
9100 NEW
9110 DATA0,1,0,0,0,16,32,186,255, 201,14
9120 DATA240,3,76,153,163,169,84, 141,24
9130 DATA2,169,13,76,153,163,169, 3,133
9140 DATA244,160,0,132,245,169, 108,141
9150 DATA24,2,177,240,74,74,74,74, 9,48
9160 DATA76,153,163,169,124,141, 24,2
9170 DATA164,245,177,240,41,15,9, 48,76
9180 DATA153,163,230,245,164,245, 198
9190 DATA244,208,216,160,2,24, 248,177
9200 DATA240,113,242,145,240,136, 16,247
9210 DATA216,169,64,141,24,2,169, 32,76
9220 DATA153,163
```

LOW COST I/O

Sir—I have found a cheap way of providing two 8 bit latched outputs from my UK1Ø1. At present, IC2Ø is a 74138, which is used to decode the video RAM and keyboard latch. It also will decode two other memory blocks at D4ØØ (pin 9 of IC2Ø) and D8ØØ (pin 1Ø of IC2Ø). These pins are normally high, but go low when data is written to the above addresses. These signals from IC2Ø can be used to control data latches, receiving data from the data bus. I used two pairs of 7475s. These are arranged like IC2 and 3, taking care to remember to invert the signals from IC2Ø. The latches can have information POKEd in from BASIC, using locations 53272 and 55296 (pin 9 and 1Ø latch control signals). I intend to use the data from one of the latches for automatic tape recorder control, but at present they are still flashing l.e.d.s!

D. J. Anderson,
London SW4.



It should be emphasised that material presented in Prompt has not necessarily been proven by us. Neither can compatability with *all generations* of the computer equipment to which it relates be guaranteed.

Software and hardware designs submitted should be accompanied by a declaration to the effect that it is the original work of the undersigned, and that it has not been accepted for publication elsewhere.

MEMORY CHECK

Sir—I am writing to tell you of a useful technique for Compukit machine code programming. When writing a m.c. program it is usually best to write it such that it may be run in any part of the RAM. This is often impossible because the program may modify its own instructions. An example is a RAM check program which should check every byte possible by storing a value in it and then recovering the value and checking it has not changed. If this program tried to check the RAM in which itself was stored it would go wrong. To get round this problem you must write a program which can find its own start address so it knows where it is in the RAM, and can avoid itself.

This can be done by executing a "JSR" and then decrementing the stack pointer twice and then reading from the stack the return address of the "JSR" previously executed.

```
JSR     FEC9
TSX
DEX
DEX
TXS
PLA
STA     $Ø3
PLA
STA     $Ø4
```

"FEC9" is a location in the monitor which contains "60" (RTS). The above program would put the address of the last byte of the JSR FEC9 instruction into stores 'Ø3' (High port) and Ø4 (low port). After executing this routine the stack will be unaffacted.

What follows is a fully relocatable RAM check program.

This program can be located anywhere from ØØØ5 up. It will check every byte except for ØØØØ to ØØØ4 and the RAM which it is stored in. To check the RAM to see that every bit cell will set to "1" and "0" it stores 10101010 and then 01010101 in each byte. The program runs through until it meets on error (the end of the RAM if your 2114's are okay). The program will display the first non reacting byte's address using the monitors output routine.

The contents of each byte is restored after checking.

The program runs as a subroutine and will return whatever it found. The only stores affected by running are ØØØ1–ØØØ4 and ØØFE, ØØFF.

Chris Dunning,
Bristol.

ADDRESS	HEX	NEMONIC
0300	20C9FE	JSR $FEC9
0303	BA	TSX
0304	CA	DEX
0305	CA	DEX
0306	9A	TXS
0307	68	PLA
0308	8503	STA $03
030A	68	PLA
030B	8504	STA $04
030D	A900	LDA $00
030F	A8	TAY
0310	8502	STA $02
0312	A905	LDA $05
0314	8501	STA $01
0316	A504	LDA $04
0318	C502	CMP $02
031A	D013	BNE $032F
031C	A503	LDA $03
031E	C501	CMP $01
0320	D00D	BNE $032F
0322	18	CLC
0323	A501	LDA $01
0325	6970	ADC $70
0327	8501	STA $01
0329	A502	LDA $02
032B	6900	ADC $00
032D	8502	STA $02
032F	8101	LDA ($01),Y
0331	8500	STA $00
0333	A900	LDA $00
0335	9101	STA ($01),Y
0337	D101	CMP ($01),Y
0339	D025	BNE $0360
033B	A955	LDA $55
033D	9101	STA ($01),Y
033F	D101	CMP ($01),Y
0341	D01D	BNE $0360
0343	A9AA	LDA $AA
0345	9101	STA ($01),Y
0347	D101	CMP ($01),Y
0349	D015	BNE $0360
034B	A9FF	LDA $FF
034D	9101	STA ($01),Y
034F	D101	CMP ($01),Y
0351	D00D	BNE $0360
0353	A500	LDA $00
0355	9101	STA ($01),Y
0357	E601	INC $01
0359	D002	BNE $035D
035B	E602	INC $02
035D	18	CLC
035E	90B6	BCC $0316
0360	A501	LDA $01
0362	85FE	STA $FE
0364	A502	LDA $02
0366	85FF	STA $FF
0368	20ACFE	JSR $FEAC
036B	60	RTS

EDITOR CUT

Sir—Having now completed the cutting down of the UK101 BASIC SCREEN EDITOR I feel you may care to publish it.

The program now uses 210 bytes (rather than the original 325) and uses locations $FB to $FF as temporary stores—these locations are also used by the reset routine.

The program is poked into locations from $022F to $02FF so does not use any BASIC workspace and also survives a Reset/cold start only needing the two appropriate Pokes.

Having spent about 50 hours cutting this program down I would like to see it made available to other enthusiasts who still have the original monitor.

Incidentally, the RUBOUT key will no longer give line feeds—it can be returned to the original program logic by making line 80, 4th DATA number 106 instead of 40.

Also it is likely that with the mark 2 monitor line 260, 2nd DATA number should be 205 instead of 204—This affected the edit cursor original horizontal position.

Hoping you can find room in Micro Prompt.

J. D. Owen,
Pendine, Dyfed.

```
40 Shortened Editor
50 POKE 250,211
60 DATA164,253,165,252,145,249,200,208,2,230
70 DATA250,208,41,198,253,76,111,2,202,16
80 DATA3,232,16,40,172,0,2,169,32,153
90 DATA0,211,136,206,0,2,198,251,169,95
100 DATA153,0,211,165,254,240,222,164,253,165
110 DATA252,145,249,136,132,253,177,249,133,252
120 DATA169,95,145,249,164,255,169,0,96,32
130 DATA186,255,132,255,201,21,240,55,201,28
140 DATA240,192,201,2,240,217,201,13,240,96
150 DATA72,173,0,2,197,251,16,11,56,165
160 DATA253,233,64,176,2,198,250,133,253,173
170 DATA0,2,133,251,104,201,4,240,137,201
180 DATA6,208,7,165,252,72,32,47,2,104
190 DATA76,153,163,56,165,254,240,6,164,253
200 DATA165,252,145,249,230,254,165,254,201,16
210 DATA240,21,201,1,208,5,173,0,2,133
220 DATA253,165,253,233,64,176,2,198,250,168
230 DATA76,101,2,169,0,133,254,169,211,133
240 DATA250,173,0,2,208,239,169,0,133,254
250 DATA133,249,169,211,133,250,169,32,133,252
260 DATA169,204,133,251,133,253,169,13,96
270 FORZ=559TO767:READ A:POKE Z,A:NEXT
280 POKE536,116:POKE537,2
OK 290 NEW
```

STRING PUZZLE

Sir—For a mild surprise, UK101 owners might care to answer the question "MEMORY SIZE" with the letter A.

Can anyone help with this problem. It seems that the string created by using STR$ is not the same as that which comes from putting the same material in quotes.

The following program illustrates the difference:—

```
10 X = 8: X$ = STR$(X)
20 PRINT X$; ASC(X$);
ASC(MID$(X$,2,1))
RUN
 8 32 56
```

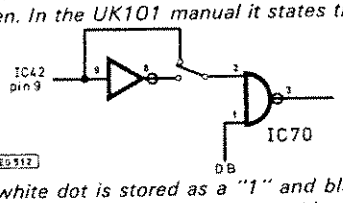
Thus the STR function appears to place a blank character at the left hand side of the 8, yielding "32" when examined, and to find the 8 requires looking at the second character, whose ASCII value is of course 56.

This does not happen if the first statement is X$ = "8".

R. J. Newman,
Chesham.

REVERSE VIDEO

Sir—The following modification will allow black characters on a white screen instead of the normal white characters on a black screen. In the UK101 manual it states that



the white dot is stored as a "1" and black as an "0", so by inverting this, a white dot becomes "0" and a black dot becomes "1". This can be done by the following modification. By inverting the output of IC42 this changes the serial video data from "1" to "0" and "0" to "1". There are a number of spare invertors on the board. I have also put a switch across the gate so that I can use both types.

I suggest that screened leads should be used to connect the switch as interference upsets the video signal.

Gerhart Ellett,
Gt. Yarmouth.

MICRO PROMPT

The hardware and software exchange point for PE computer projects

M/C FOR TELEWRITING

Another method for direct telewriting on the UK101 has been submitted by *T. Wilson* of Sittingbourne, Kent.

First set up in BASIC by pressing C and limiting the memory size.

10 POKE 11, 00 : POKE 12, 5
20 X=USR (X) : P=PEEK (1536)
30 PRINT CHR$ (P) ; : GOTO 20

Then enter machine code by restarting and pressing M. Type in the following:

Ø5ØØ 2Ø ØØ FD	JSR FDØØ
Ø5Ø3 8D ØØ Ø6	STA Ø6ØØ
Ø5Ø6 6Ø	RTS

Using Warm Start you may now type out all the standard characters, and more, by using CTRL plus a key. Use return and LF to start a new line, and CTRL/C to exit.

CLEAR OFF

SIR—The manual supplied with UK101 kit includes a short machine code routine for clearing the VDU screen. The following routine is shorter, and arranged for the specific task for clearing screen. The manual version is a more general routine for loading any section of memory of any length with a specified data byte.

```
nnnn  A9 2Ø       LDA #$2Ø
      AØ ØØ       LDY #$ØØ
      99 ØØ DØ    STA $DØØØ, Y  'fill'
      99 ØØ D1    STA $D1ØØ, Y
      99 ØØ D2    STA $D2ØØ, Y
      99 ØØ D3    STA $D3ØØ, Y
      C8          INY
      DØ F1       BNE 'fill'
```

Michael Wood,
Wakefield.

DREADFUL!

Sir—I am delighted to see that you are devoting space to supporting the UK101, which you published as a series of articles last year.

There is a need for a good system of data storage for this machine. The SAVE routines published are all very well but they lack sophistication and the other essential—simplicity!

If one compares a true SAVE operation on other computers, one can only describe these genuine attempts at a solution as a dreadful way to have to carry on, and with only limited commands.

We had mini-floppies, there is a Philips digital cassette recorder, and now the stringy-floppy.

How about an article/series on how to add on one of these to the UK101? For people like me this would preferably be available as a ready-constructed add-on as an alternative to a kit, which would appeal to many readers.

Allan Batch,
Rugby.

EDITOR'S MAILBAG

Sir—Congratulations to Mr. Climpson for the very useful screen editor published in July's Micro Prompt. It gets rid of the cursor on Superboard after CONTROL Ø too, which is quite an achievement!

It had been pointed out that in the list of page zero addresses given by Mr. Hocking in his save-variables program location ØØ88 is, in fact, the high byte of the current BASIC line number in use and locations ØØ8B, C are the GOSUB pointer.

E. J. Keeley,
National Personal Computer Users Association.

Sir—I have found the Editor program (by Nigel Climpson, in July PE) most interesting and useful, and I wish to thank you for not forgetting the Superboard owners. I have made the following addition to the program to cause it to limit the memory automatically for my 8K machine without the need for remembering the memory size;

50001 POKE 133,184: POKE 134, 30

For a 4K machine the line would be:

50001 POKE 133,184: POKE 134, 14

When saving the program on tape I carried out the following routine:

1. *Type SAVE (Return) LIST*
2. *Start recording (Return)*
3. *When program is completely SAVED do not stop the recorder, but instead* slowly *type RUN (Return)*
4. *When OK appears press return as required to activate the program and then use the Editor itself to copy line 50074, altering the line number to 50000.*
5. *Type LIST (Return), which gives an assurance that only the new line 50000 exists.*
6. *Stop tape recorder.*

To load the program from tape it is only necessary to play back the whole tape. The program is loaded and run, the line is copied and the editor is ready for use. Various error signals are given but all is well as long as the new line appears three times. To activate the editor after a warm start type RUN50000 (Return).

The reason for typing RUN slowly when recording is to give time on playback to stop loading if it is desired to delete line 50076 before use.

For my Superboard I have found that byte 1F65 should be 1C (or 28 decimal in line 50040) to allow the cursor to reach all lines on my screen, and eight Control D's are required to avoid blanks when a line is carried over.

I send you the above information in the hope that most of it is applicable to the UK101 or can easily be adapted for it.

F. S. Dewhurst,
Keighley.

RAMLESS MESSAGES

Sometimes it can be helpful to print messages on tape before a LISTing to give information that does not need to be put into RAM, particularly if a lot of RAM is being used for variables and strings. The problem is that a straight:

PRINT "THIS IS A MESSAGE"

in SAVE mode will be recorded and played back, but there will be an accompanying Syntax Error message which looks tatty. The answer is to start the message with a colon.

PRINT ":THIS IS A MESSAGE"

will play back without a syntax error and will not load into RAM. This means that when the program has been fully developed it can be recorded with additional comments by adding a routine on the lines of the following:

50000 SAVE:?:?:?": THIS IS A MESSAGE ABOUT THE PROGRAM"
50010?": WRITTEN JULY 1980"
etc.
50090 LIST–49999
the program is SAVEd by RUN 50000

The resulting recording will contain the message but not lines 50000 onwards. The colon is also useful for improving the appearance of LISTing, as it can be used instead of REM to produce a blank line for spacing.

A further elegance can be added by getting the Compukit to print "RUN" instead of "OK" at the end of the listing, thus giving you a self running tape. The OK message is printed by a JMP $A8C3 in locations 3,4,5 of the memory. As has been pointed out in the First Book Of Osi, changing the $4C to $6O in location 3 disables the OK, but if instead you put in a jump to your own message routine it will print anything you like. The ROM message printer prints a message which starts at an address loaded into its Y (hi byte) and X (lo byte) registers; the message being terminated by a NULL. Thus, in the routine shown it prints (CR) (LF) RUN (CR) (LF). It is activated by POKE 4,40:POKE 5,2 and will stay activated until COLD start or further POKEs.

Any message is possible provided it is terminated by a Null.

```
0228 AO O2      LDY@$O2; Load address of message (Hi)
022A A9 2F      LDX@$2F ; Load address of message (Lo)
022C 4C C3 A8   JMP $A8C3 ; Message printer from loc. 3,4,5,
022F            .BYTE 13,10, 'RUN', 13,10,0

022F OD
0230 OA
0231 52
0232 55
0233 4E
0234 OD
0235 OA
0236 OO
```

Roger Derry,
London.

SHIFTY CHARACTERS—2

Sir—Oh dear, nearly a whole column wasted with a table of characters of the keyboard of the 101—a table one can so easily find for oneself by playing with the machine. And, Mr. Schofield has missed the whole point of the Shift Lock. Press it, and all the lower case letters (including k) are available where the upper case used to be. That is the reason, presumably, why the normal position of the Shift Lock is down, as it works in the same mode as a standard typewriter keyboard. Viz. up for lower case and down for upper case. Note also that when it is up, the Shift keys are still operative, so that if you wish to mix l.c. and u.c. (and you probably do!) then your strings are entered exactly as on a typewriter. Care is necessary with digits and punctuation, however.

What I would like to know is what CTRL.O does to disable the keyboard (except for Return) if tapped once. Tapping twice (or holding down) will produce the "large house." CTRL.M gives Return. What, if anything, do other control characters do?

May I conclude by saying how I enjoy reading your magazine. I would like to see some circuits which will enable the 101, the Edukit and similar machines to interface with the real world of temperature sensors, infra-red detectors, electric door locks and so on. As a teacher of Computer Studies it is more important to show pupils how the electronics can move mountains rather than win at games.

G. R. Morris, B.Sc.,
Cheshunt, Herts.

Oh dear, don't blame Mr. Schofield, we foreshortened his original material.

CONVERSION—POKE SOMEWHERE?

Sir—I have followed your series of articles on the COMPUKIT UK101 with great interest as several months ago I purchased an Ohio Superboard—unfortunately just before your series. However, undaunted I set about modifying my Superboard to become a UK 101.

I've now got the machine operating via a sub-board on 50Hz. The sub-board runs on a 8MHz clock and plugs into the original counter chain's i.c.sockets. The other aim was to get 48 characters per line instead of the difficult to read 24.

The problem is that there is something in the firmware that tells each line to only be 24 characters wide, because I now have viz:

Line 1 Line 2
Line 3 Line 4
Line 5 Line 6
etc.

With the help of our Digital Engineer at the T.V. station where I work I've PEEKed and POKEd but without success.

If I purchased the UK 101 Monitor ROM and/or the BASIC ROM('s) would this cure the problem? Or can I POKE somewhere.

B.F. Bailey,
N.S.W., Australia.

INT AINT ACCURACY

The following is an extract from a letter from *Mr. J. Plews* of Sheffield:

"If I combine the truncated integer function with an exponential expression, I begin to get problems where accuracy is vital. eg:

PRINT INT(10/(2 ↑ 1))

The 101 comes back with 4"

• • •

"Is there any way of solving this problem by fooling the system etc? Has anyone else encountered this obstacle? Is there any chance of a new interpreter?

I would be interested in your response to this letter."

. . . Help! Actually, if you run your example without INT you *may* find that the 101 is returning 4·99999 etc. Although the result may be as close as a gnat's whisker to the correct figure, you will automatically be ditching this accuracy for the next lowest integer when you implement INT. Any machine would do it! Presumably you don't want all those decimal places.

Try:

```
10 INPUT N
20 PRINT INT(N*10/(2 ↑ 1))/10
```

If N=10 this should return 4·9 . . . a little closer!

Dr. BERK'S UPDATE

Sir—As you requested in your article in Practical Electronics, *March 1980, I would like to let you know that I have carried out your suggested modification to run a 110 baud teletype from the 101 and all the evidence to date would indicate that it is very successful indeed.*

I have made the modification switchable, the teletype being a fairly new Olivetti T.E.300.

Thank you for your most useful and informative articles; I look forward to more of them.

J.R. Haldene,
Midlothian.

NEW ROOM ERROR

Sir—I have unearthed a simple error on the instruction card that accompanies the new UK101 monitor, that nonetheless can cause a lot of difficulty.

The instruction card gives a helpful list of vector addresses and subroutines entries, with Hex and Decimal versions of the addresses. Using these I couldn't get BASIC to set (POKE) my NMI vector and thus couldn't get my data collecting interrupt routine to run during BASIC programs. (All was OK so long as I stuck to machine code in the Monitor.) Although it took quite a while to find, the answer was simple—I had replied on the Decimal addresses on the instruction card, but some of them are wrong. The printed list goes wrong at 021A Hex, listed erroneously as 540, should be 538. Thus NMI vector should go in Decimal 547 and 548 (an IRQ in 550 and 551) and not as printed on the card. The Hex addresses given on the card seem to be perfectly correct. The monitor seems to function well in every respect.

N. Blurton Jones,
University of London.

ARRAY OR DISARRAY?

Sir—I write with reference to a point raised in your March issue, within "Microprompt". The point to which I refer is: ?FRE(N) after running part, or whole of a program containing a DIM statement for a string variable array. e.g. DIMA$(10).

The way to avoid the UK101 "locking up" is to type: CLEAR:?FRE(N).

This clears all variables and arrays. It does of course prevent you from finding out how much memory is occupied by variables or arrays used.

As a matter of interest, numerical values ocupy 6 bytes. Within an array they occupy only 4 bytes, plus an overhead for the whole array, which is dependant upon whether it is 1-, 2-, or 3-dimensional. The overheads for 1-, 2-, and 3-dimensional arrays are 13, 21 and 29 bytes respectively. For example, an array dimensioned with DIMX(9,9) will hold 10 × 10 = 100 variables. It would therefore occupy 100 × 4 plus the overhead of 21 = 421 bytes.

This is true when some values have been assigned to all dimensions. An empty array overhead is 6 bytes less per dimension. For instance, the same array X, above, would have an overhead of only 21 − 12 = 9 bytes whilst empty, or 21 − 6 = 15 bytes with one dimension empty. (Why some one should dimension an array and leave it empty I dont know!)

Each string variable occupies the same number of bytes as characters it contains with an overhead of 6 bytes. How much room a string variable array occupies I do not yet know!

I hope this is of some interest. Finally, I would like "Microprompt" to be larger and in every issue. Certainly not Bi-monthly!

E. Cottam,
Par, Cornwall.

GETTING INTO PRINT PROBLEM

Sir—A short while ago I bought a second-hand Data Dynamics 390 ASCII printer. In spite of the fact that I was assured by the vendor that it was easy to link up to my UK101 (new MONITOR), I have been unable to find out how to do so. The only information that I have is that the printer requires an eleven bit word (one start bit, two stop bits, eight data bits) and that it has a current loop input and works at 110 Baud.

If any of your readers can help me to get the printer working, I should be most grateful.

Alan E. Wilmshurst,
Crowborough,
E. Sussex.

It should be emphasised that material presented in Prompt has not necessarily been proven by us. Neither can compatability with *all generations* of the computer equipment to which it relates be guaranteed.

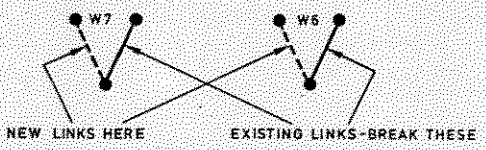
Software and hardware designs submitted should be accompanied by a declaration to the effect that it is the original work of the undersigned, and that it has not been accepted for publication elsewhere.

cuit given in the COMPUKIT manual is not strictly accurate. The circuit changes are shown in Fig. 1 and are also described.

1) Locate pads W6 and W7. When I received my machine I had to cut the tracks on W6 and W7 and rewire them in the opposite sense to enable the standard Monitor ROM properly. For a 2716 these must be returned to their original state. This is because the 2716 requires a low on pins 18 and 20 to enable it.

2) Pin 21 is the naughty one. It is shown in the manual as being held at +5V. On my machine at least it was not, but instead was strapped to pin 20 with a circuit board track. This means its

Fig. 1.



logic level goes up and down with pin 20 which is okay for the Standard ROM but stops a 2716 from working. It is necessary to cut this track and take it to +5V.

Now, as to what to fill those 1000 or so empty holes with is up to you. As an example this is how I partially filled mine.

EDITOR

The screen editor that was published in the July Microprompt now resides at ØØØ-182 with the necessary address changes and an extension to compensate for terminal width. By changing two bytes at 6FØ and 6F1 to 74 and F8 the EDITOR works immediately BASIC is called from RESET.

A rapid clear screen routine is located at 183-1A1, and 1A2-1B6 contain a program to generate tones of selectable frequency and duration through a small decoding circuit and speaker.

Finally a checksum loader occupies locations 1C2-2BD. This is very useful for loading and executing programs saved in checksum format from the Extended Monitor or Assembler/Editor. With all this there is still about 3ØØ bytes to spare in the ROM so what next? Maybe a Renumberer or a Disassembler or a . . .?

This system has been working in my machine for about three months now and I haven't noticed any ill effects from the missing bits, and since the cost of 2716 EPROMs is continually falling you may consider it a worthwhile thing to do. ★

News Briefs

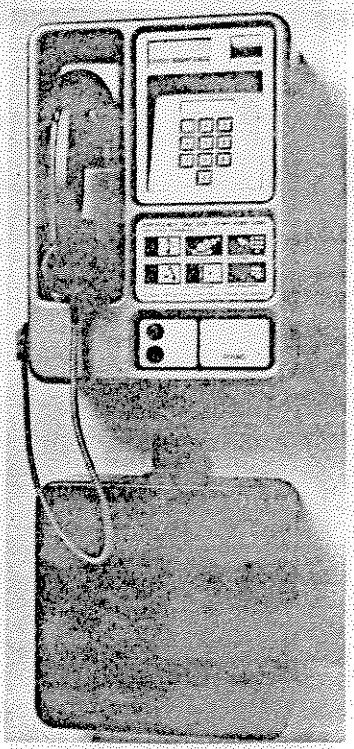
BRAINS AND BRAWN

BEWARE, the intelligent public telephone is on the way! The Plessey designed PP2000 payphone might not only have a higher IQ than its user, it will probably prove tougher too—should it come to blows.

Vandal-proof, fraud-proof, the stainless steel machine is said to be capable of withstanding an attack with a sledge hammer, chisel or crowbar. So far so good, but the microprocessor controlled creep doesn't hit back, it secretly sends out a 999 call.

Up to 80,000 of the "public call office" versions are expected to be in use in the UK *before* 1985 (1984 again). These telephones are honest. If you insert coins to the value of 60 pence for a long distance call, the machine will not only display digitally your diminishing credit as the call progresses, but should you hang up with, say, 10 pence still on the clock, it will refund the difference.

The PP2000 payphone is powered entirely from the telephone line, is very flexible in the face of rate increases or coinage changes, and informs the post office when the cash box is nearly full; and with that kind of versatility Plessey believe they should be able to grab a good share of the estimated overseas market of £50m p.a.



The PP2000 is a microprocessor controlled payphone developed by Plessey engineers at Liverpool

BASIC LINE RENUMBER

This program will renumber a UK101 BASIC program, or part of a program up or down. It will not, however, alter GOTO or GOSUB line numbers. The program can be loaded at any time when programming, and is brought into operation by typing RUN 20000. It is useful to use to "space out" part of a program to allow extra lines to be inserted. To understand the operation of the program, the workings of the BASIC interpreter, in particular, how it stores data in RAM, needs explaining.

```
20000 REM*** LINE RENUMBERER FOR UK101 *********
20010 REM*** I. PAWSON FEBRUARY 1980 ***************
20020 INPUT"RENUMBER OLD LINES FROM";X
20030 INPUT"THROUGH TO";Y
20040 IFX)YTHEN20020
20050 INPUT"AS NEW LINE NUMBERS FROM";Z
20060 INPUT"IN STEPS OF";P
20070 B=256
20080 DIMN(100,2)
20090 N(1,1)=769
20100 FORT=1TO100
20110 A=N(T,1)
20120 N(T+1,1)=PEEK(A+1)*B+PEEK(A)
20130 N(T,2)=PEEK(A+3)*B+PEEK(A+2)
20140 PRINTN(T,2);N(T,1)
20150 IFN(T,2)=YTHEN20170
20160 NEXT
20170 FORT=1TO100
20180 A=N(T,1)+2
20190 IFN(T,2)<XTHEN20240
20200 POKEA,Z-INT(Z/B)*B
20210 POKEA+1,INT(Z/B)
20220 Z=Z+P
20230 IFN(T,2)=YTHEN20250
20240 NEXT
20250 PRINT:PRINT
20260 PRINT"RENUMBERING COMPLETED"
20270 REM*** END OF PROGRAM ********************
OK
```

As the program is in BASIC, all numbers are in decimal. The data is stored from address 769 onwards; the first two locations are the address of the start of the next line. The format is that the second location contents are multiplied by 256 and added to the contents of the first location. The following two locations contain the line number, in the same format as the address. The contents of the line follow, terminated by a 0. The commands, GOTO, GOSUB are stored as a single number, 136 and 140. The line number following is stored in ASCII format, ie 100 as 49 48 48, 200 as 50 48 48 and so on. The main problem, in adapting the program to renumber GOTO's and GOSUB's would be if the new line number had more or less digits than the previous one. This program is presented as a starting point for a complete renumberer, and I should be interested in readers' comments. Line 20140 can be omitted if a print-out is not required.

I. Pawson, Leicester

PEVDU KEMITRON STYLE

Because the Thompson-CSF CRTC chip is really intended for use in serial or "glass teletype" terminals, and is therefore equipped with its own rather slow cursor control system, problems can arise when attempting to use the PE VDU to best advantage as a memory-mapped VDU. As is mentioned briefly in part 3 of the PE VDU article, if a monitor program which generates its own cursor in software is to be used, the cursor control lines $C_0C_1C_2$ are best permanently set to 0, and an initial pulse applied to the ST line (Fig. 4, part 3). This can be easily achieved if the ST line is wired to the reset circuit of the MPU. However, this is not the whole story. We are now left with a flashing character and cursor line at the top left-hand corner of the screen. The cursor line is generated by the CRTC disabling the character generator at the appropriate time, and therefore this can be removed by breaking the track joining pin 15 of the CRTC to pin 11 of the 2513, and permanently grounding the latter.

This leaves the 2513 permanently enabled. To avoid the flashing character in the top left hand corner, the output routines in the interpreter or monitor used can be arranged to ignore completely the extreme left-hand column of the screen, while the clear-screen routine must of course include this column. This is the system adopted in the Kemitron NIBL-MM-memory mapped BASIC interpreter for the SC/MP, one version of which is specially configured to work with the PE VDU—though the principles described above could of course be incorporated in any interpreter or monitor program for any processor.

Details of the Kemitron system can be obtained from Greenbank Electronics, or the Chester Computing Centre, 21–23 Charles Street, Chester.

JUST A LITTLE SOMETHING . . .

```
100 REM★★★ 8 DIGIT BINARY TO
    DECIMAL CONVERT ★★★
110 REM★★★ I. PAWSON DEC
    1979 ★★★★★★★★★★★★★★★
120 PRINT:PRINT
130 INPUT "INPUT 8 BIT
    BINARY NUMBER"; A$
140 IFLEN(A$) ><8THEN100
150 B=0
160 FORX=1TO8
170 Y$=MID$(A$,X,1)
180 READA
190 C=VAL(Y$)
200 B=B+(A★C)
210 NEXT
220 PRINT
230 PRINT "THE DECIMAL VALUE
    IS"; B
240 RESTORE
250 PRINT
260 GOTO130
270 DATA128, 64, 32, 16, 8, 4, 2, 1
280 REM★★★ END OF PROGRAM
    ★★★★★★★★★★★★★★★★★
```

LINE LENGTH HINTS

Sir—T. D. Allen of Poole wants to display 64 characters per line on his TV from a UK101. He has a problem! The suggestions that he makes are all based on software but the problem is mainly in the hardware.

As far as the software is concerned, changes must be made to a ROM in program locations FFE0 and FFE1, cursor starting point and line length respectively.

The main problem is physically displaying the characters. The VDU RAM is scanned as 16 lines of 64 characters with TV horizontal sync pulses added at the end of every line. As everybody knows, a few characters are not displayed at each end of the line. What is not often said is that more characters are lost during the fly-back of the TV trace and these can never be displayed.

The write-up in the UK101 book about VDU operation explains that clock pulse C7 is not used and the result is that every row of dots is displayed twice. The logical conclusion is that using C7 instead of C6 to generate the horizontal sync should give a line twice as long with the 2 rows of dots side-by-side instead of one on top of the other. Putting the right values into FFE0 and FFE1 (try BF and 3F) should then enable a complete set of 64 locations to be displayed, assuming that you can adjust and/or modify the TV as necessary. Display on a normally adjusted TV would not be possible.

You will have gathered that I have not tried this and there may well be problems, although the theory sounds good. It should at least give food for thought.

R. L. Taylor, Shepperton, Middlesex.

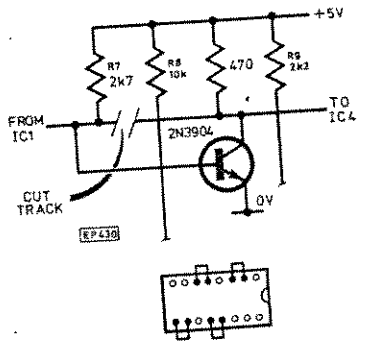
PSG BUG

We received correspondence from *Mr. Gossage* of Harrow, Middlesex, pointing out an error in the address decoding of the PSG:

"It is stated that the Hex address used is F0F0. On the UK101 the whole 256 word block from F000 to F0FF is used as the ACIA location—the two addresses are repeated throughout the page and so cannot be used by the PSG. Secondly, however, the circuit given will not show only at F0F0 but appear at many addresses from 00C0 upwards. Inspection reveals that IC1 is not acting as an 8 input NAND gate but as a 4-wide 2-input NAND OR gate—the output going low when any pair of inputs go high."

"The circuit will apparently work since it is, in effect, 'write only memory' and so although it appears at many memory locations it will not interfere with the micros use of the RAM."

The author, Mr. D. Coutts, has provided a fix for this, and apologises for the error: Change IC1 to 74LS09 (same pin-outs), and add 2N3904 transistor and 470Ω resistor as shown below. This gives the PSG one address



only, and requires minimal interference to the p.c.b. Our correction to Fig. 8, is inlcluded too. Also, Mr. Gossage suggests transposing address lines A7 and A8 to place the PSG at F170 and F171—away from the ACIA.