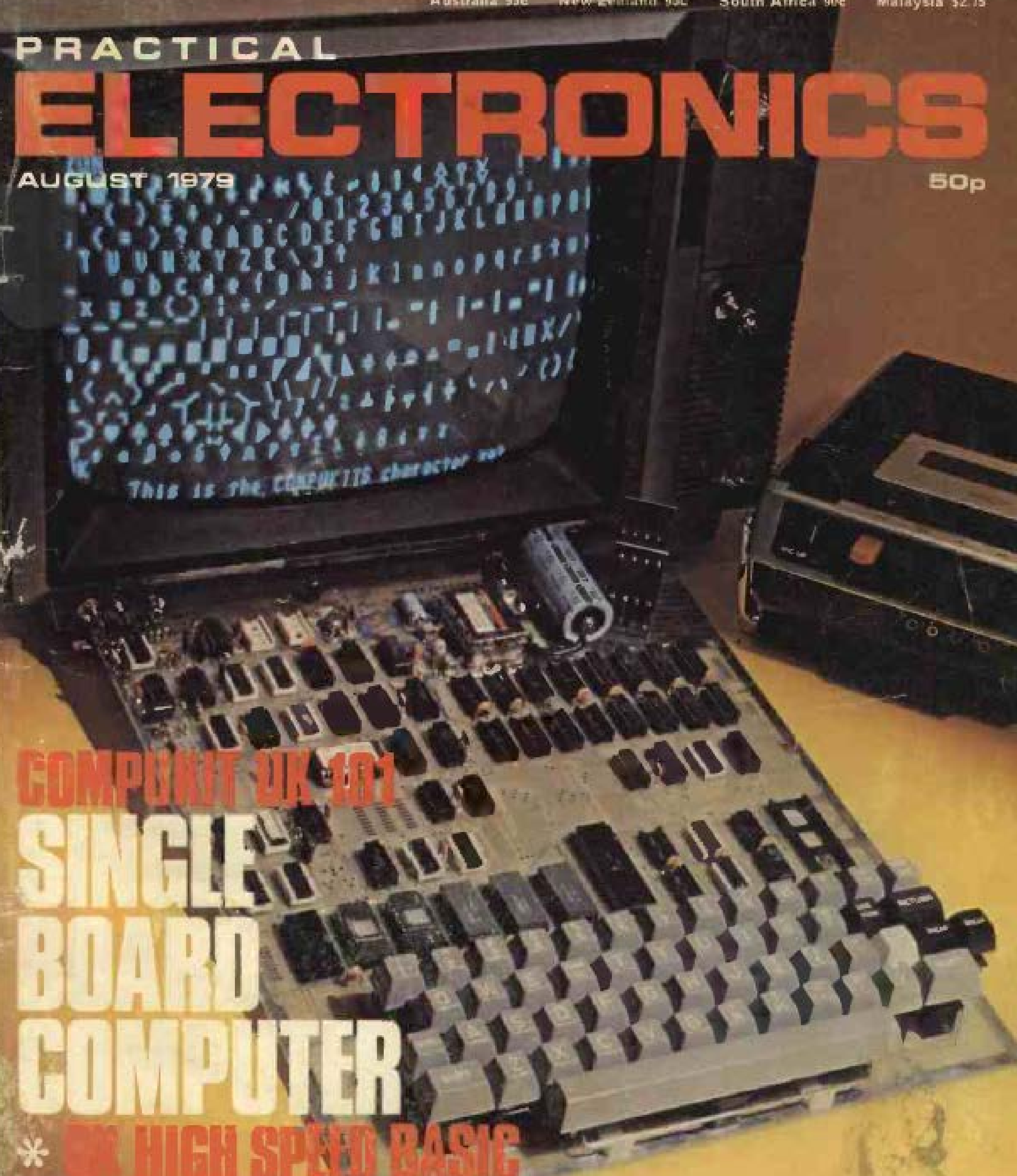
Australia 95c New Zealand 95c South Africa 90c Malaysia $2.75

PRACTICAL ELECTRONICS

AUGUST 1979

50p



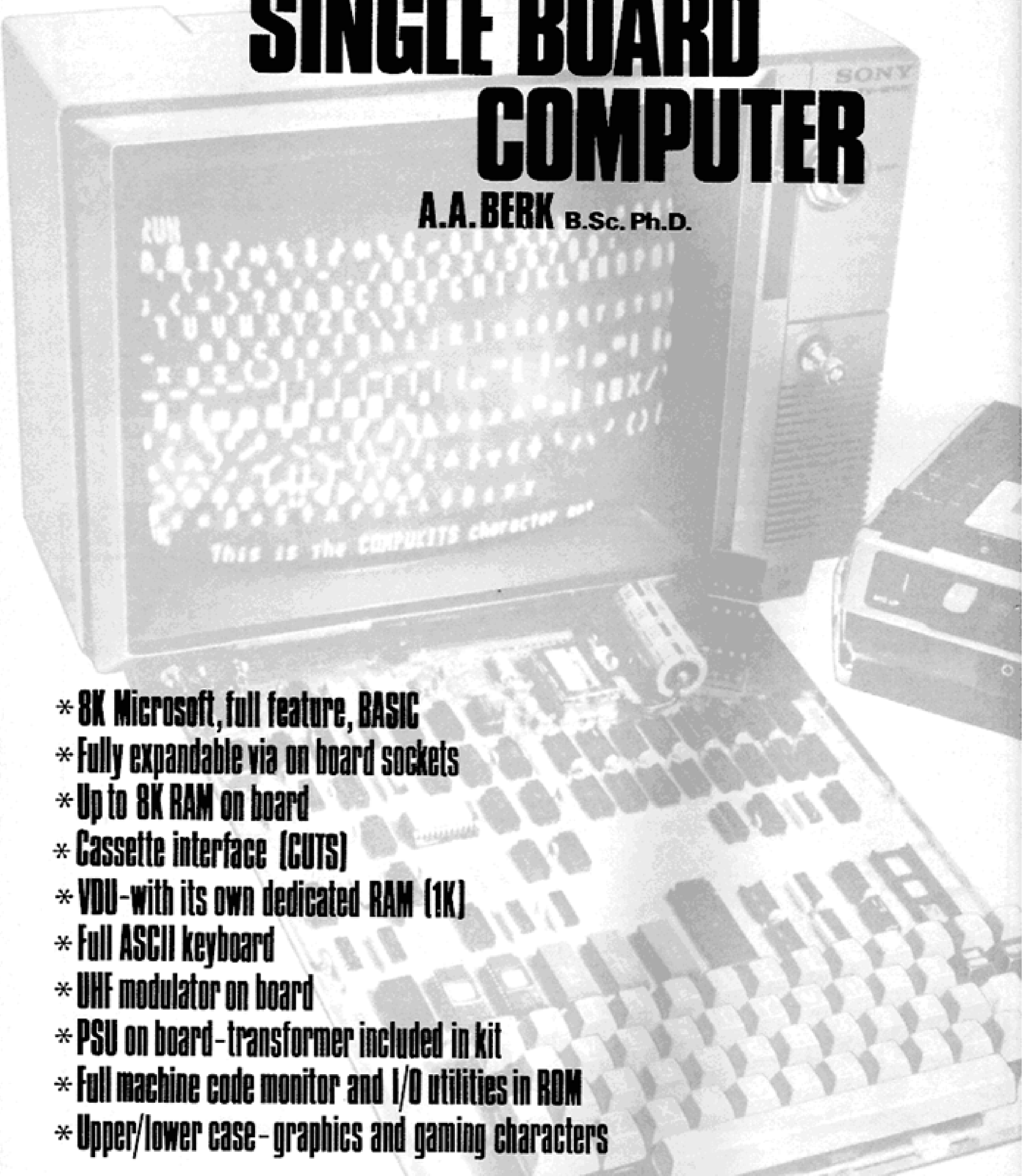
COMPUKIT UK101 SINGLE BOARD COMPUTER

* 8K HIGH SPEED BASIC
* FULLY EXPANDABLE

Also: HOME FREEZER ALARM

COMPUKIT UK 101 SINGLE BOARD COMPUTER

A.A. BERK B.Sc. Ph.D.



* 8K Microsoft, full feature, BASIC
* Fully expandable via on board sockets
* Up to 8K RAM on board
* Cassette interface (CUTS)
* VDU-with its own dedicated RAM (1K)
* Full ASCII keyboard
* UHF modulator on board
* PSU on board-transformer included in kit
* Full machine code monitor and I/O utilities in ROM
* Upper/lower case-graphics and gaming characters

THE COMPUKIT UK-101 has been prepared to provide all sectors, from the amateur constructor/hobbyist to the engineer, programmer, teacher, business man and scientist, with the cheapest computer on the market which is programmable using an 8K BASIC package. In addition, it is fully expandable, capable of being programmed in machine code, and absolutely complete from mains input to TV Aerial feed output. It includes full typewriter-style keyboard, printer, and cassette-storage interface. The BASIC package employed is the industry standard Microsoft interpreter, and will run much of the available software for many other more expensive machines on the market with little modification. Floating-point calculations in this BASIC are extremely fast and versatile, making scientific calculation projects of the most complex kind perfectly feasible in one's own home. The areas of suitable use for the COMPUKIT stretch fully across the board.

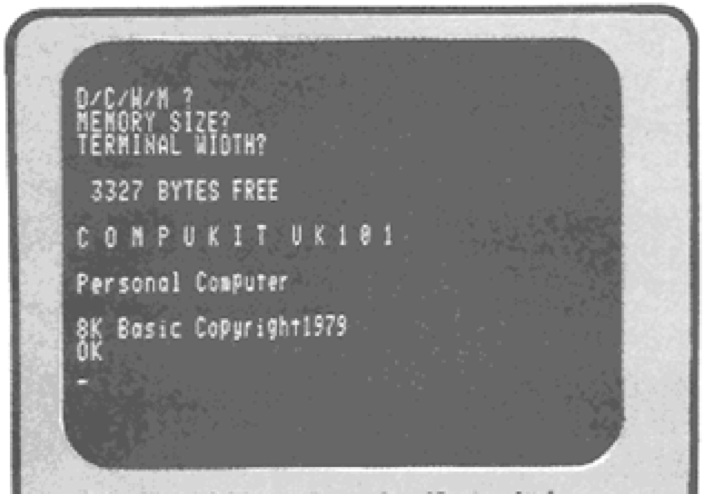
The business man will find that the string handling facilities and excellent video display of the COMPUKIT make the machine perfectly fitted for everything from general accounts and planning calculations, to formatting of invoices and stationery to be printed out on hard copy when complete.

The Statistician will find that the high speed of calculation with facilities such as random number generation and user-defined functions, enable analyses of data and simulation of models to be performed particularly easily. The results may then be displayed with the graphics available in almost any form from bar-charts to tree-diagrams!

The Engineer wishing to control equipment with a computer, will benefit from being able to write the necessary routines in BASIC (making the software easy to design) while being able to jump to fast and efficient machine-code routines for servicing external hardware.

Perhaps most importantly; for the first time, anyone, technical or otherwise, can have an affordable easily programmable home-computer for domestic use—a much neglected area at the moment, but one which is about to take off.

Imagine being able to run a program to help your child (or yourself) revise for exams. Animated diagrams are possible, such as an internal combustion engine shown reciprocating, with mathematical equations to match. Picture an *automobile* program designed to accept daily input from you on petrol and oil use, along with mileage, etc. The machine could accumulate information, and at any time give you the average m.p.g., plot performance versus time or some other parameter, and remind you when to check the tyres and drive to the Garage for a service.



```
D/C/W/M ?
MEMORY SIZE?
TERMINAL WIDTH?

 3327 BYTES FREE

C O M P U K I T   U K 1 0 1

Personal Computer

8K Basic Copyright1979
OK
-
```

The UK-101 introduces itself at switch-on.

- ★ ***Based on the popular 6502 micro***
- ★ ***Supports all Ohio Scientific expansions including floppy disc***
- ★ ***Fast resident BASIC***
- ★ ***Total of 19K addressable memory on p.c.b.***
- ★ ***Memory mapped VDU. Line width selectable from 16 to 49 characters by 16 lines***

IMPROVEMENT

This computer is based on the Ohio Scientific Superboard *(reviewed PE June 79)*, which only displayed 23 characters per line on a normal TV, and produced the USA standard 60Hz frame frequency unsuited to UK TVs. These and other disadvantages have been rectified in the COMPUKIT UK-101. For example, the keyboard has been altered to show certain missing characters on the Superboard, and the BREAK key is less prone to accidental operation.

This month, the COMPUKIT hardware is briefly described in block diagrammatic form, along with some technical details of the 6502 Microprocessor. A section on the specifications of the BASIC package is included, for which the reader will require some previous knowledge of BASIC—a library book would be useful at this point for those just beginning in the world of programming.

A BASIC primer article is to be published in this magazine very soon and will help with an understanding of this very popular microcomputer language.

HARDWARE

Based around the 6502 MPU and a set of binary counters as clocks, the COMPUKIT has been designed and set out to be as flexible and easily modified as possible while making no compromises for those who will only use the machine as it stands.

The 6502 is a reasonably typical 8-bit microprocessor with a 16-bit address bus and 8-bit data bus (see Fig. 1) and the control lines are similar to the 6800. There are two interrupt lines: IRQ and NMI. When forced low, the latter will interrupt execution at the end of the current instruction cycle and branch to a routine whose address (vector) is stored in two bytes at the top of memory. IRQ is similar but may be rendered inoperative by setting (to "1") the MASK flag of the P-register. RES is a reset line whose vector is also stored at the top of memory.

There are five more control lines which have the following effects. R/$\overline{W}$ informs the system as to whether a READ or WRITE operation is being performed. SYNC is used to identify an Op-Code fetch cycle. RDY is used to halt the processor for DMA or slow memory, etc. S.O. is a line which may be used to set the Overflow bit in the P-Register. Ø1out and Ø2out are system clock lines.

Other pins on the MPU's 40-pin package are used for Data, Address and Power Supply (+5V).

The register set of the 6502 is smaller than that of the 6800 and Z80 as only one primary accumulator is included and there are no 16-bit data or index registers available to the user. This is partly made up for by an enhanced set of addressing modes which use pairs of memory locations to hold addresses. However, there are no 16-bit Load, Store or Compare instructions to manipulate these pairs directly and all such operations must be constructed from 8-bit arithmetic operations.

WARNING: Computing is highly addictive.

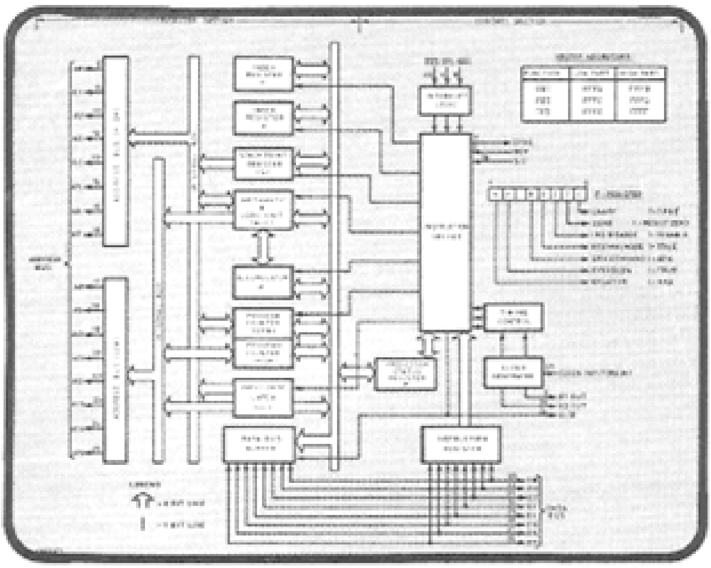


Fig. 1. 6502 internal architecture

The Stack is implemented in Page One (the second 256 bytes of memory), as the Stack pointer is only of 8-bit length—a ninth bit exists but is permanently set to "1".

Page Zero (the first 256 bytes of memory) is very heavily used by the 6502 instruction set as its two Index registers are both 8-bits in length. However, as referred to above, indirect addressing exists whereby an instruction may take its effective address for execution from a pair of bytes, stored anywhere in memory. Various Offset modes exist using the index registers. In this way, pairs of memory locations act as 16-bit registers; but, again, they are not able to be manipulated with 16-bit instructions. Try writing a program to clear a block of memory greater than 256 bytes in length and you'll see what I mean!

If you have worked on the Z80 or 6800 you will find certain aspects of the 6502 code most frustrating while others you will discover to be something of a breakthrough. Readers who are used to the SC/MP or 1802 instruction set will probably be most surprised at the difference when working on such an advanced machine.

The speed of processing is similar to the Z80 and 6800 and all three exist in various speed options. One interesting point is the 6502's "rise to power" as probably the most widely used Micro on the market. APPLE, PET and KIM all use the 6502 and there are several other 6502-based systems in the pipeline, all of which implies that the software backup for the 6502 is very wide and increasing in size.

SYSTEM DESCRIPTION

The Block Diagram, Fig. 2, shows the main parts of the COMPUKIT in a condensed form. The heart of the system is a set of binary clocks fed from an 8MHz crystal controlled clock generator. The clocks are not only used for the MPU but form the main control for the VDU, which is continually polling the VDU RAM for information to be displayed at exactly the right time on the screen. Those familiar with the PE VDU (*P.E. Oct.–Dec., 1978*) will note that the clocks perform most of the functions associated with the Thompson CSF Cathode Ray Tube Controller chip.

The most dominant block on the diagram, is in fact the VDU—a fairly typical memory mapped system run from the binary counters. The screen of information to be presented is stored in the VDU RAM in a binary-coded form (ASCII) for the 128 ASCII characters, and extended to include codes for the various graphic characters (128 in all) available on the COMPUKIT.

Address bits are sent to the VDU RAM from the binary counters in sequence starting at the location to be displayed at top left of the screen. As the counters ripple through all the addresses, the full screen of (theoretically) 1024 characters is accessed and the data at each location passed on to the character generator which interprets it as video dots to be displayed on the TV. Each character occupies 16 TV lines (in pairs to form eight vertically arranged dots) and the binary counters supply the necessary counting information along the character generator row-address lines.

When the MPU wishes to interrupt this process and access the VDU RAM, it switches the Address Switches and tri-state buffers over to the MPU Buses.

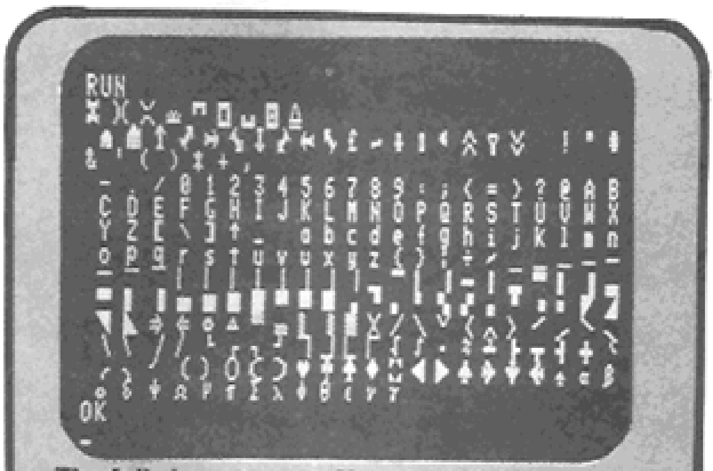
In order to display the information from the character generator on a TV, it must be serialised by the Parallel In Serial Out (PISO) device, mixed with Frame and Line synchronisation pulses, converted from TTL levels and fed to the UHF modulator. The output may then be plugged directly into a TV aerial socket and the TV tuned to around Channel 36.

The Sync. pulses for the TV are generated by a set of monostables from the main binary counter chain and the design includes a display-blanking stage to prevent MPU access to the VDU from causing excessive "noise" on the screen. This allows fast real-time updating of the video information without upsetting the display.

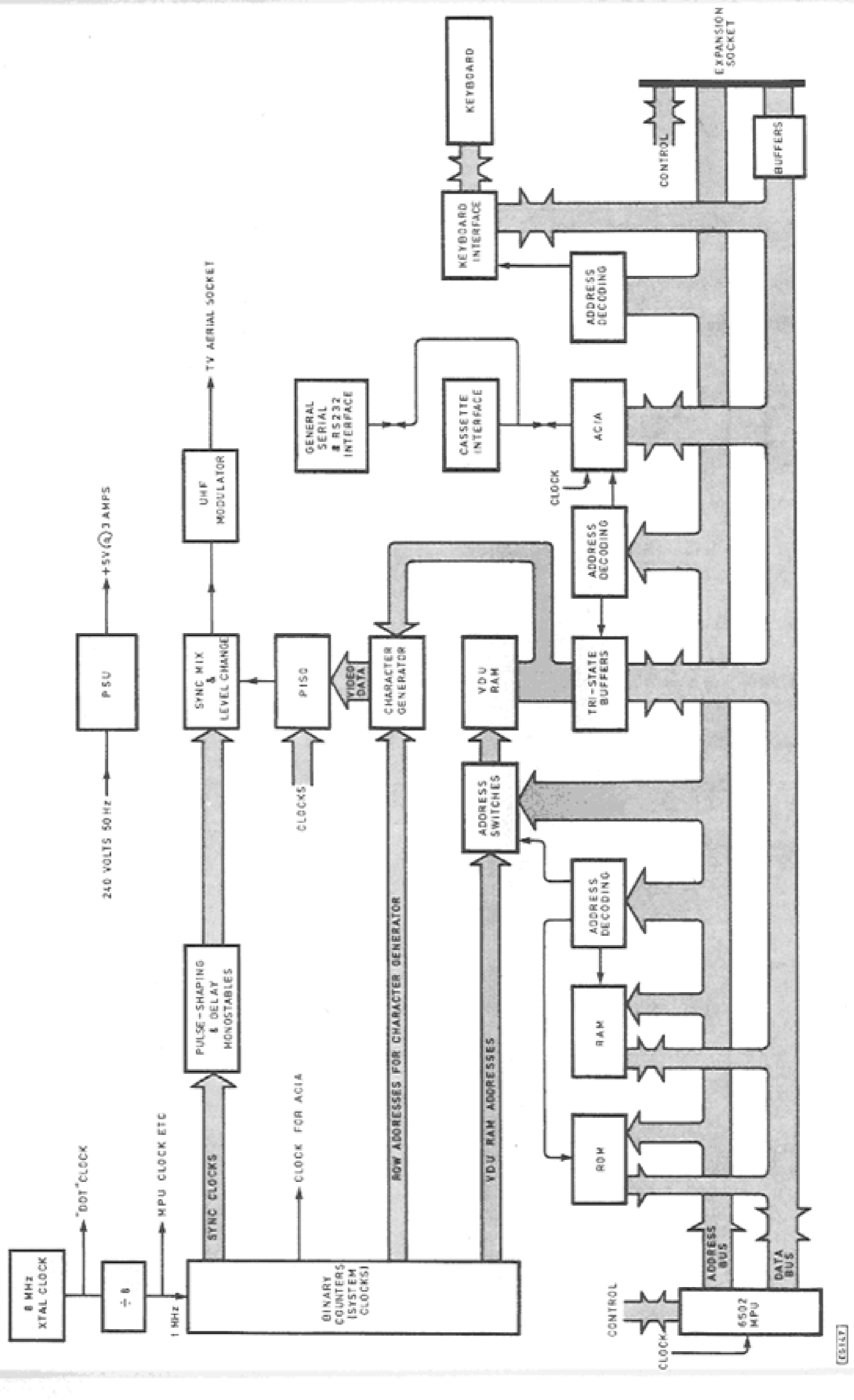
The VDU *Dot* clock on the COMPUKIT oscillates at 8MHz, which is fast enough to display about 48 characters per line on a normal TV. 768 characters are thus displayed on the 16 lines. The rest of the 1024 remain "hidden" off the screen but are perfectly valid Read/Write memory locations. This number of characters keeps the bandwidth down to a value acceptable by the I.F. stages of all domestic sets, and straight video need never be used due to the resulting excellence of definition in the characters displayed by the COMPUKIT.

The "hidden" characters of the VDU are never used by the BASIC interpreter, of course, as the software is carefully modified to use just the 49 characters per line which appear. The number of characters thus used per line is also selectable by the user for those TVs which will only comfortably display 48 or less characters across their screens.

The other sections of the COMPUKIT are shown connected to the Address and Data buses as well as to Address Decoding logic which selects each block as its address comes up on the Address Bus. The various control lines are omitted for simplicity.

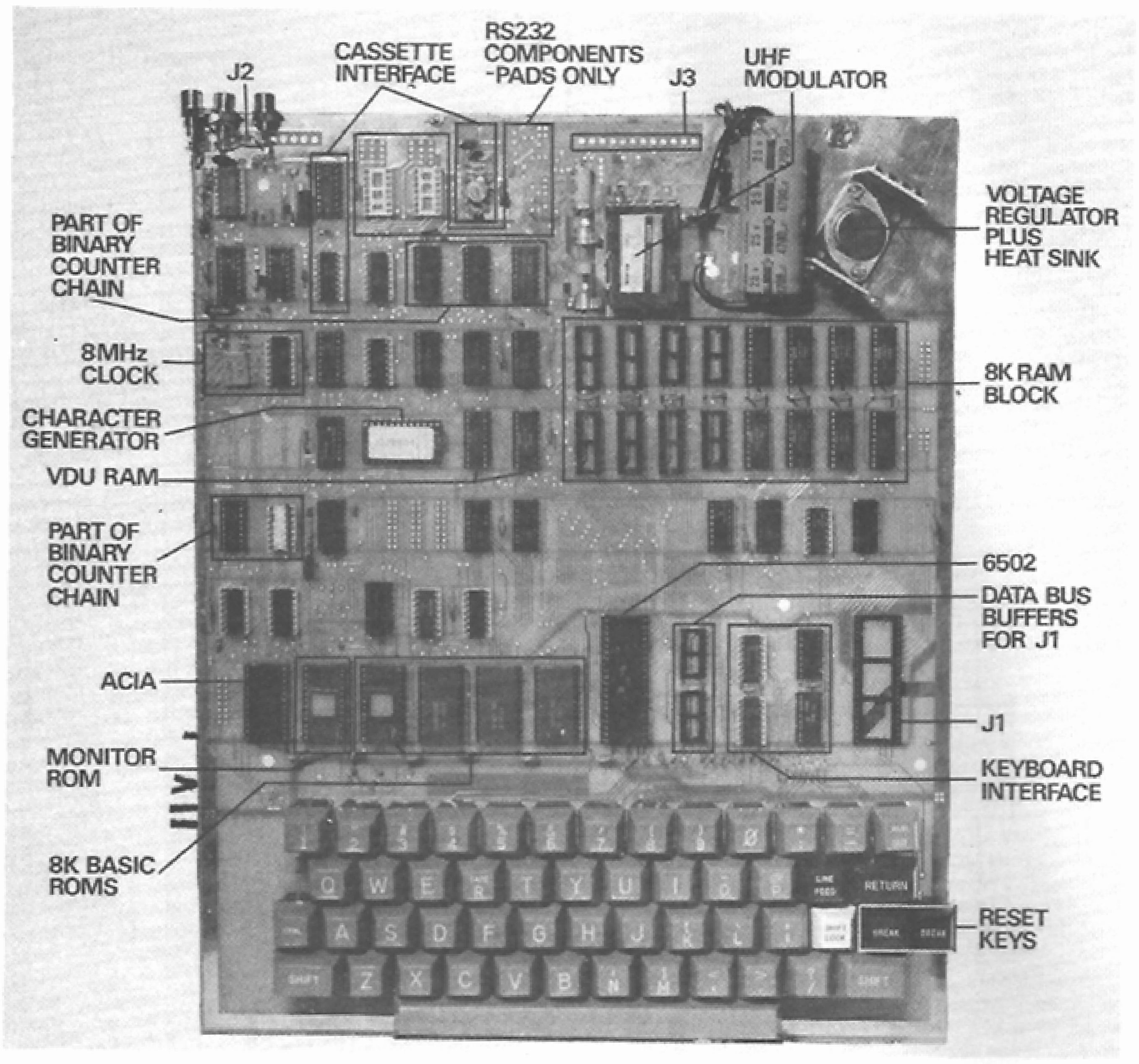


The full character set. Note that alpha-numeric characters are vertically spaced to give uncluttered lines of text, but graphics characters join together to allow pictures to be formed.



UK-101 COMPUTER

Fig. 2. Block diagram



The COMPUKIT UK-101 board. A transformer is supplied but does not mount on the p.c.b.

(Note that this is the prototype board)

The Bus lines are controlled and operated by the 6502 MPU which derives its instructions from the ROM and RAM blocks which take up between them 18K of memory. 2K of this is a Monitor (operating system) with BASIC support, keyboard handling, machine-code monitor, floppy-disc bootstrap, etc. 8K is the BASIC interpreter and 8K is for the RAM block, of which around $\frac{3}{4}$K bytes are used as a scratchpad by the various ROM-based programs.

Another important block is the Asynchronous Serial Communications section—including the cassette interface. This is driven by an ACIA (Asynchronous Communications Interface Adapter—the 6850 chip). The clocking comes, of course, from the binary counter chain and may be fed from higher up the chain to effect faster transfer speeds than the 300 BAUD for which the system is set up. In addition, there is provision, on board, for a RS232 interface run by the same software as the Cassette. This allows connection of the system to any standard serial device from teletypes to modems and extra VDUs.

The keyboard section is also shown connected to the Data Bus and Address decoding, and this is run by software scanning for speed and efficiency. The software scans the keyboard for a key-closure and identifies the character being accessed from ROM.

Finally, and of great importance, is an expansion socket (40-pin d.i.l.) for the Address, Data and Control lines to allow expansion to a separate Motherboard. This could easily be extended to the S100 Bus, but is plug-compatable with the Ohio Scientific OSI-48 Bus and Floppy-Disc interface. Anyone wishing to use the COMPUKIT for control purposes will find the expandability via this socket of great assistance.

The 5V Power Supply Unit (PSU) gives 3 amps.

COMPONENTS . . .

Resistors

R1–R12, R68, R80, R81	4k7 (15 off)
R33	6k8
R34, R50	15k (2 off)
R37, R55, R63, R64	10k (4 off)
R35	560
R36, R60, R61, R65	470 (4 off)
R51, R74	270 (2 off)
R52, R54, R59, R72, R82	1k (5 off)
R53	22k
R56	100k
R58	560
R62	100
R67	27k

All $\frac{1}{4}$W 5% carbon film

Capacitors

C6	150p ceramic
C7, C12, C55	1n mylar (3 off)
C8	100n mylar
C9	68p ceramic
C10, C13	10n mylar (2 off)
C11	1µ mylar
C48	220n ceramic
C57	27p ceramic
C58	47µ electrolytic
C59	3,300µ electrolytic
C60	100p ceramic
remainder	100n disc ceramic (31 off) (These are all supply decoupling capacitors)

Diodes

D1–D10	1N914 (10 off)
D15	1N4001
D17, D18	Any 3A rectifier diode (2 off) (1N5401 will do)

Integrated Circuits

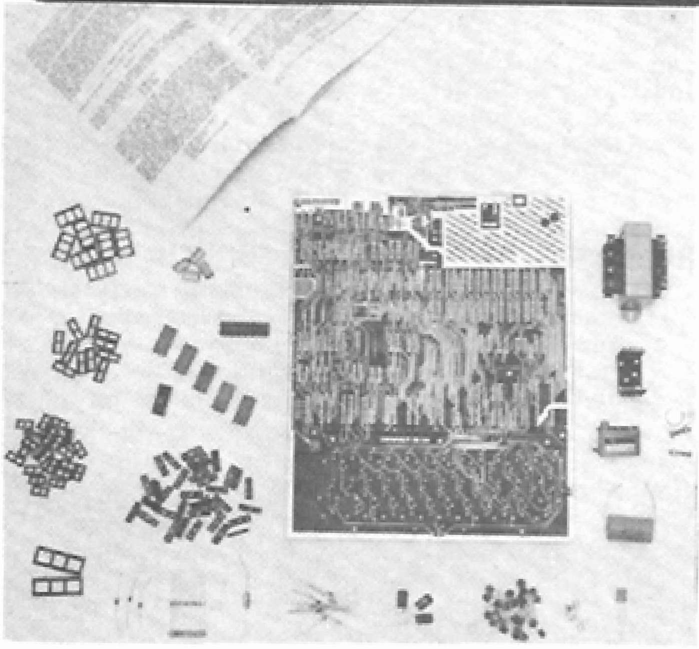
IC1	Any 5V, 3A TO3 type regulator
IC2, IC3	74LS75 (2 off)
IC4, IC5	74LS125 (2 off)
IC6, IC7	8T28 (used only for expansion)
IC8	6502 microprocessor
IC9–IC12	ROMs for BASIC (4 off)
IC13	ROM for monitor
IC14	6850
IC15	7402
IC16, IC18, IC21, IC62	7404 (4 off)
IC17	74LS139
IC19	74LS20
IC20, IC22, IC23	74LS138 (3 off)
IC24, IC25	8T28 (2 off)
IC28, IC65, IC69	74LS123 (3 off)
IC29	7493
IC30, IC59–IC61	74163 or 74161 (4 off)
IC31–IC40, IC45–IC52	2114 (18 off)
IC41	00000 Character Generator
IC42	74165
IC53–IC55	74LS157 (3 off)
IC56	7420
IC57	74163
IC58	7400
IC63	7474
IC64	7476
IC66	CA3130
IC70	7403

Miscellaneous

d.i.l. sockets

40-pin	2 off
24-pin	7 off
18-pin	18 off
16-pin	21 off
14-pin	13 off
8-pin	1 off
FS1	3 Amp fuse plus holder
X1	8MHz crystal
UHF modulator	Astec 8MHz bandwidth
Double-sided plated-through p.c.b.	
Set of keyboard switches and key tops	
Regulator heatsink plus fixing screws	
Transformer—240V/50HZ : 9-0-9V 3A	

Please note that component numbers do not run contiguously



CONSTRUCTORS' NOTE

The *Compukit UK–101* is being produced by **Comp Components** (see advertisers' index), for £219 + VAT. This will include a full power supply on-board (including power transformer), and a UHF modulator. The graphic character set also includes such symbols as £ and π.

The p.c.b. is of a professional standard with plated-through holes and silk-screened component legends, and a full set of i.c. sockets is included in the kit price as is—with 4K of user RAM (a further 4K may be purchased to upgrade the board to its full capacity).

The Compukit comes with full manual (written by the author), but the beginner would be advised to obtain a book on BASIC, and perhaps even on 6502 machine code programming, as the Compukit manual is not intended to be a primer on either subject.

The computer is available for demonstration at Comp Components.

SOFTWARE

The machine accepts its instructions in the programming language called BASIC as well as 6502 machine-code. It is the use of BASIC which will be paramount and its specifications are given below.

There are many versions of BASIC (Beginner's All Purpose Instruction Code) for many different machines. There are Tiny BASICS, Integer BASICS, Double Precision BASICS, etc., etc. Minis and Main Frames usually offer a version of BASIC in Compiler form which includes all the Scientific and other functions plus the ability to manipulate matrices and Disc files. Few if any of these versions, however, allow direct manipulation of memory locations within the computer when programming or debugging. Access to machine-code routines is also a rather complex operation and use of the machine to control even such simple things as a bank of lights is usually impossible. It was not until the advent of the MPU-based computer with its feet firmly embedded in memory address decoding, machine-code programming and control applications that such interplay became common. BASIC has become the high-level language of the micro-computer, and interpreters rather than compilers are the normal implementation method. The interpreter's ease of use made it the order of the day and Microsoft's BASIC interpreter was born. This is a very powerful package and exists in many versions of which the COMPUKIT's is one of the very fastest.

It is quite different from the 1K and 2K BASICS with their incredibly restricted instruction sets. The COMPUKIT's version, as its name implies, occupies 8K (8192) bytes of the machine's memory and is stored in Read Only Memory (ROM). This type of memory has two advantages:

(1) The BASIC interpreter is not lost when the machine is switched off—indeed it is almost impossible to lose it short of crushing the chips in a vice.

(2) The language is available instantly upon switch-on and does not need to be loaded from some external medium into the machine's memory.

Other methods of offering BASIC include storage on disc. The COMPUKIT can be expanded (by a direct plug-in option) to run the Ohio Scientific's Disc System which comes complete with a powerful disc BASIC, as well as 24K of extra memory to run it in.

Despite the fact that the resident BASIC (in ROM) is at the top end of the scale for speed and sophistication, this does not mean that it is difficult to program. On the contrary, BASIC's use has blossomed for one paramount reason—it is sophisticated enough to satisfy the most technical user in industry, business and research while being remarkably easy even for total beginners to learn and use. The reader will recognise that most of the language is written in a kind of formalised logical English.

SUMMARY OF BASIC SPECIFICATIONS

There are three modes of use of BASIC on the COMPUKIT:

(a) Command mode
(b) Immediate-mode execution
(c) File-mode execution

The computer is said to be in the Command Mode whenever the message:

OK
—

appears. This is usually after a system Reset following initial power-up, or at the end of a program. The "—" is a cursor which moves around the VDU screen and keeps the user updated as to his/her position. In this mode, the following commands are allowed:

Software has unexplored dimensions awaiting the adventurous, and sophisticated games are possible which can be as much fun to devise as to play.

CONT (short for CONTINUE), LIST, NEW, NULL, RUN, SAVE, LOAD. These include editing, running programs, continuing after a Breakpoint and Cassette handling. Most of these commands may also form statements within a program.

Immediate mode of execution is similar to Command mode but involves normal BASIC statements which may be ganged together to form complete programs executed when the RETURN key is pressed. In this mode, for instance, the machine is available as a super calculator with Scientific calculations of great complexity able to be executed *immediately.*

To file program statements *for later execution,* and to build up complete programs, statements are given line numbers to distinguish them uniquely and place them in instruction sequence in the computer's memory. The finished program may be started by using the command RUN and this is termed File-mode execution.

STATEMENTS AVAILABLE

CLEAR	LET
DATA	ON . . . GOTO
DEFFN	ON . . . GOSUB
DIM	GOSUB . . . RETURN
END	PRINT
FOR . . . NEXT	READ
GOTO	REM
IF . . . THEN	RESTORE
IF . . . GOTO	STOP
INPUT	WAIT

RELATIONS, OPERATION AND FUNCTIONS

−	SUBTRACT (OR NEGATION)	NOT	Bolean operations from which complex logical functions may be constructed, e.g. IF . . . THEN
+	ADD	AND	
*	MULTIPLY	OR	
/	DIVIDE		
↑	RAISE TO THE POWER OF		

>	GREATER THAN	>=	EQUAL TO OR GREATER THAN
<	LESS THAN	<=	EQUAL TO OR LESS THAN
<> (OR ><)	NOT EQUAL TO	=	EQUAL TO

ABS (X)	SGN (X)
EXP (X)	SIN (X)
FRE (X)	COS (X)
INT (X)	TAN (X)
LOG (X)	ATN (X)
PEEK (I)	SPC (I)
POKE (I, J)	SQR (X)
POS (I)	TAB (I)
RND (X)	USR (X)

The variables I and J denote integers and X denotes any number. This is a collection of scientific and other functions which includes calculation of the number of bytes left free for program storage, manipulations of address locations directly, and calls to machine code subroutines.

STRING FUNCTIONS

ASC (X$)	MID$ (X$, I, J)
CHR$ (I)	RIGHT$ (X$, I)
FRE (X$)	STR$ (X)
LEFT$ (X$, I)	VAL (X$)
LEN (X$)	

These functions allow a string of alphanumeric and graphic characters to be stored, manipulated, sorted, etc. to a very sophisticated degree. A numeric string may, for example, be converted to its numeric equivalent and back again. The use of all these BASIC statements will be clearly explained in subsequent issues of PE, for the benefit of the beginner.

VARIABLES, TYPES, ACCURACY

Numeric variable names may be any alphanumeric string starting with a letter and containing no special BASIC words. Only the first two characters define the variable. The number of variable names thus available is of the order of 900. String variables are similar but their names must end with a $. For example A1 is numeric and A1$ is a string. Numeric and string arrays are also allowed, and increase the already large number of variables available by a considerable factor.

An accuracy of 6½ digits is used for Scientific calculations, and numbers from 10^{-38} to 10^{+38} are allowed. The computer automatically shifts into exponent form (Scientific representation) as necessary. String variables are from 0 to 255 characters in length. Both string and numeric arrays of several dimensions are allowed.

ABBREVIATIONS AND SPECIAL CHARACTERS

The character **?** may be used instead of the word PRINT, and LET and END are optional, as is the variable in a NEXT statement under certain conditions. Spaces are irrelevant to the BASIC and colons may be used to separate statements on the same program line.

The above features allow fast programming with improved use of memory space, in addition, errors in typing may be deleted by use of the DEL key.

This completes the general overview of the system, and next month the details are given in full. Some details of programming in BASIC will be given and will be selected to "tie in" with the introduction to BASIC to be published in *P.E.* in the near future. 6502 machine-code is of a more specialised nature and the reader is well advised to buy a good primer on this subject along with a technical description of the 6502 if it is intended to use the machine at this level. Those with any expertise in machine-code programming on another machine will find that learning the 6502 code is quite simple however, and a 6502 data sheet will probably suffice.

HARDWARE AND SOFTWARE INGENUITY

In this series of articles, applications for the UK 101 will be suggested, and it is hoped that software houses will be interested in providing cassette packages for as many uses as can be dreamed up.

The hardware will also be described in such detail as to allow maximum freedom to modify or add to the basic design, and readers are welcomed to let us know of any software/hardware accomplishments or suggestions they may have.

KEYBOARD HARDWARE

The photograph shows the positioning of the keys on the keyboard; while Fig. 3 shows the arrangement of the keyboard switches and interface logic.

The keyboard looks like a matrix of 8 rows and 8 columns to the computer; the Rows being one *Write-Only* memory location, and Columns being *Read-Only*. These two occupy the same memory location (DFOO in HEX) and the computer distinguishes between Rows and Columns by writing to or reading from DFOO.

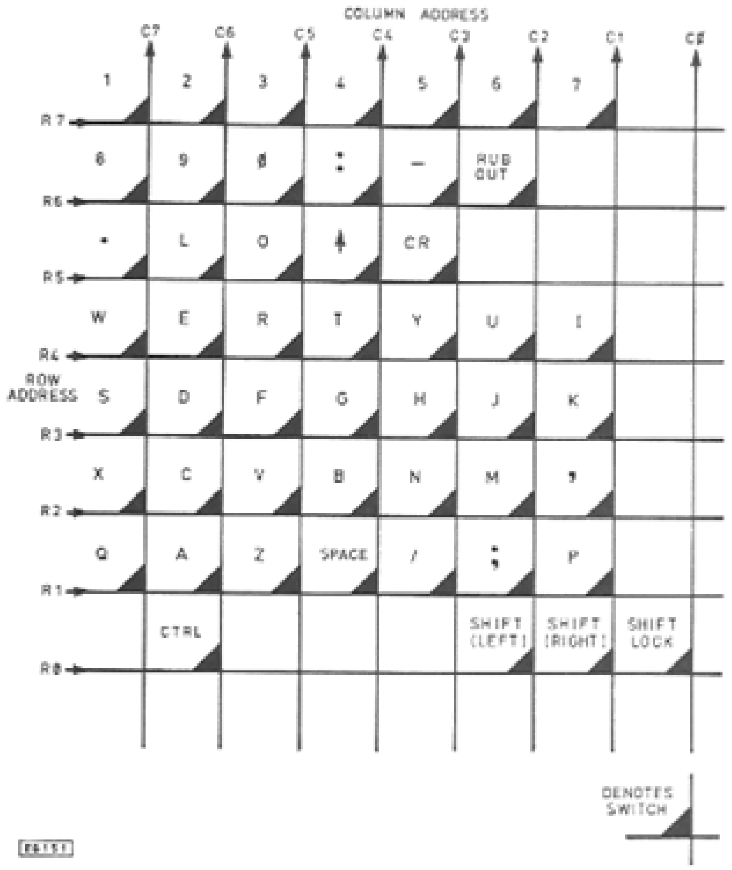
The MPUs R/$\overline{\text{W}}$ line is used to control WKB and $\overline{\text{RKB}}$ accordingly by some logic to be described next month.

Essentially, the software is designed to send a "walking bit" to R0–R7. Thus, the R0 line is set to a "1", then back to "0", followed by the same on the R1 line, and so on, up to R7 and back to R0 again. The "1" scans along the Rows waiting for a key closure to transmit it to a Column (one of the C0 to C7 lines). If character A on the keyboard is pressed, the contact time must be long enough for the "1" to reach R1 (a very short time at worst). The A key then transmits it to C6, and the Computer receives and processes an A character. The fact that the walking 1 was being sent to R1 and immediately received along C6 makes the A key unique. R0 is used to decode the control keys at the same time; thus, if R0 and C1 followed immediately by R1 and C6 is decoded, the computer receives a SHIFT A character. The encoding for the keyboard is thus stored in the Monitor ROM.

When the computer has written to a Row, the Data Bus switches to *Reading* from the Columns. Normally, therefore, the Row information would thus be lost and the 1 would never reach any of the Columns. Hence the reason for the latches IC2 and IC3 which store the walking bit long enough to be read from the Columns as necessary. The diodes on IC2 and IC3's O/Ps protect against multiple key closures shorting two or more O/Ps together. IC4 and IC5 are tri-state buffers which ensure that the keyboard information is presented to the Data Bus only when the computer is reading the Columns, otherwise the keyboard would present its information at a time when the Bus was being used elsewhere, thus causing an electrical conflict. Pull-up resistors are included on the inputs of IC4 and IC5, as C0–C7 are open circuit most of the time.

The keyboard decoding is thus dependent upon a particular program in the Operating System, and whenever that routine is inoperative, for instance while certain parts of a user's program are under way, the keyboard is unusable. However, the INPUT statement in BASIC allows the operator to input data via the keyboard during a program. In addition, the address of the keyboard routine itself is available for use via the USR function within a BASIC program.

The only two keys not mentioned in the diagram are the two BREAK (or RESET) keys. These connect directly to the RESET line of the MPU and must *both* be pressed to reset. This prevents accidental breaking in most circumstances.



POWER SUPPLY UNIT

The power supply is very straightforward since the Compukit is a single supply device with distributed capacitive decoupling on the board. All but the transformer are mounted on the p.c.b. and a heat sink is provided for the 5V regulator. The photograph shows the extent to which supply noise decoupling is allowed for on the p.c.b. 100n capacitors appear at the most important places to prevent spikes and crosstalk being transmitted along the power lines.

The PSU also has a 3A fuse and a polarity compliance diode to give some protection against accidentally reversing the +5V and 0V lines by those using an external PSU for any reason.

The UHF Modulator used on the board is the 8 MHz Astec device which gives an excellent output when supplied with five volts. This device was chosen to ensure that the p.c.b. remained a single-supply system.

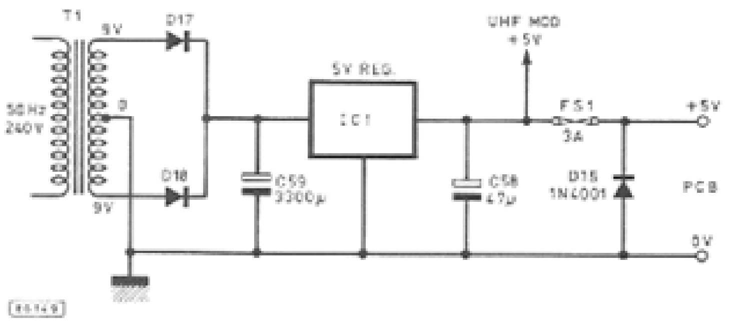
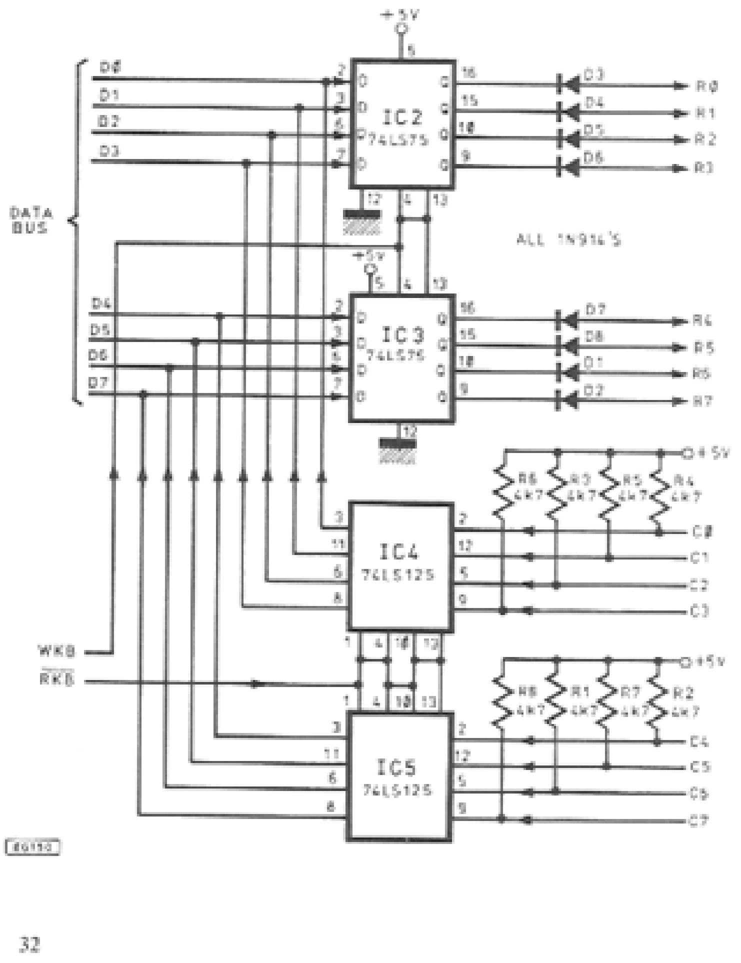
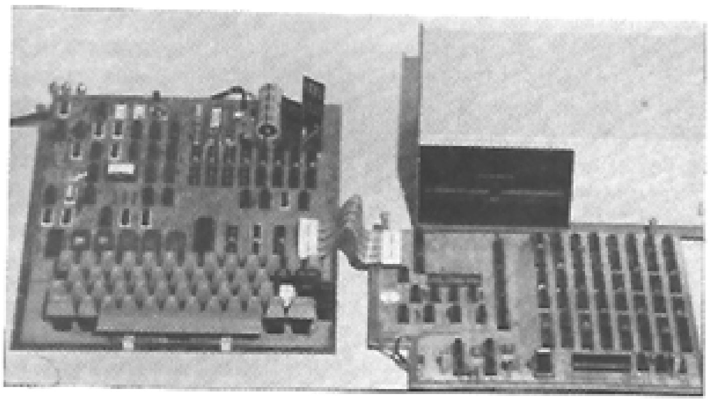


Fig. 3. Above: Keyboard switching matrix
Below: Keyboard interface circuit

Fig. 4. Power supply circuit diagram

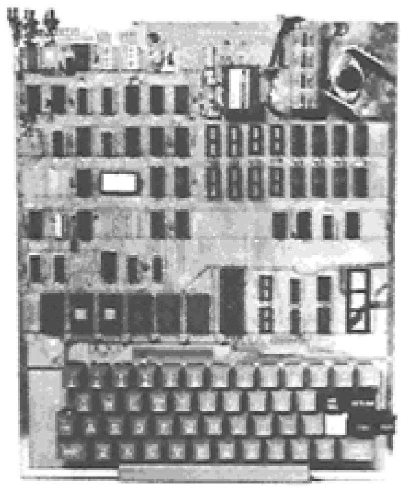




The UK-101 working happily with a floppy disk.

NEXT MONTH: Constructional details of this powerful computer are given, so learn computing the comprehensive way, using a high level language and working down to hex code programming. The UK-101 is capable of both!

COMPUKIT UK 101 SINGLE BOARD COMPUTER



PART 2 A.A. BERK B.Sc. Ph.D.

A CERTAIN amount of the following will be considered unnecessary by the experienced, although some points are very important. The constructor is advised to read through this section at least once!

You will need a good pair or wire cutters, a small screwdriver and a soldering iron of around 15-20 Watts with a narrow bit. The bit should ideally be new—make sure you coat the end with solder *as it first warms up* or a patina of corrosion will immediately form making soldering impossible. Also, iron-clad bits must not be filed for cleaning them or the anti-corrosion property is lost. The thinnest resin-cored solder should be used.

Never try to drill any of the p.c.b. holes out, as this will destroy the plating-through. All solder connections are made to the bottom of the board and no i.c. pins must remain unsoldered even if they *appear* to go nowhere. The board should be protected at all times from excessive abrasion, flexion, and contamination.

ASSEMBLY

Following the component legend very carefully, the best sequence of construction is to start with the i.c. sockets. Locate and push their pins carefully through the holes, taking extreme care to prevent pins from being bent *under* the socket. The socket must be pressed very firmly against the p.c.b. while two pins are soldered down to keep it in place.

Sockets may not be supplied for the following positions: IC67, IC68.

All i.c.s are fitted with pin 1 towards the keyboard except for IC41 (Character Generator) whose pin 1 is towards the RAM block. Socket polarity is normally identified and even though i.c.s will fit either way around, put the sockets in correctly as a reminder for the future. Do not insert the i.c.s yet.

Insert the discrete components, except for the voltage regulator, UHF modulator and large capacitor. The 100n bypass capacitors should be soldered in last, to prevent a mix up. Most of the resistors stand on end. *None* of the components will tolerate overheating, *especially the crystal.* Remember, once a device is soldered in place its removal is *very difficult* because of the plating-through. A solder sucker is very useful for this eventuality, but sockets are particularly troublesome and are usually destroyed by the operation.

KEY SWITCHES

Next insert and solder the keypad switches from top right to bottom left. Each switch is labelled on the p.c.b. and the switch, and with correct key-top, may be inserted carefully in place. Do not use undue force or heat, as the switch is quite delicate until held in place. Operation will be impaired if the switch pins are pushed into the thermo-plastic body.

The pins must be soldered with the switch *pressed firmly against the p.c.b.* All switches except SHIFT LOCK are return sprung, so do not make the mistake of fitting this switch elsewhere, which will stay down when pressed once and return on the second press. The SPACE bar and switch is fitted last. The bar should be placed over the switch and the white plastic locators into their holes. Carefully heat-form the projections beneath the board to hold the bar in place. *Use the back of your soldering bit.*

Before continuing, check for shorts across the key-switch terminals and between Data and Address Bus lines at IC8.

The regulator (with heat-sink), UHF modulator and large electrolytic, may now be soldered in place. Solder flux should be removed with methylated spirit, using an old tooth-brush. Fully inspect the board for solder bridges or broken tracks (a watch-maker's glass is invaluable for this task).

The power supply can be checked at this point to ensure it delivers five volts to each of the i.c. sockets.

Insertion of the i.c.s is a delicate process *and pins are very easily bent between the chip and the socket* (often undetectable), causing hours of fruitless searching for a bug. Pins should be bent straight from their normal splayed out condition and pushed bit by bit, inspecting continually, into their sockets.

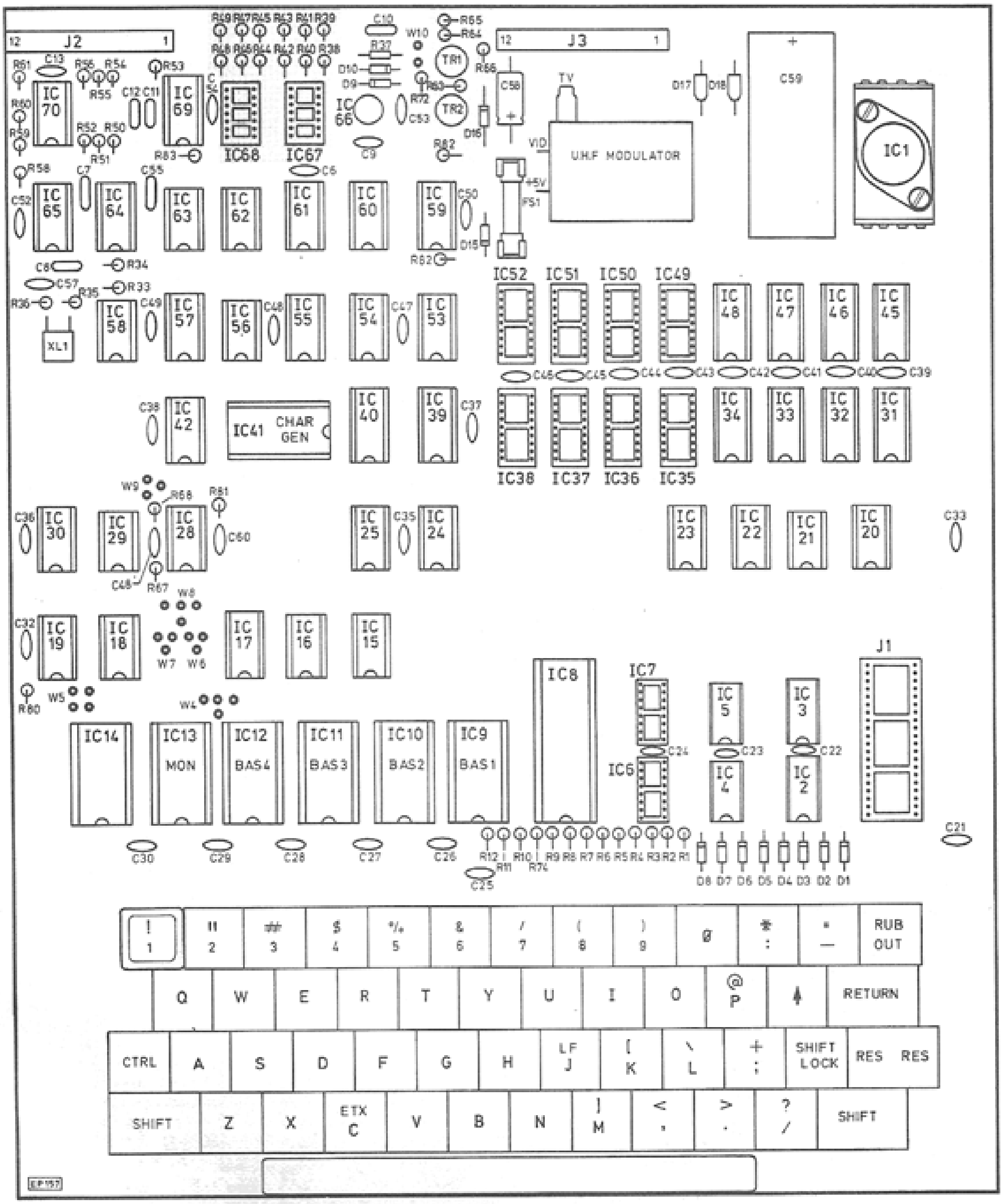
A final check of i.c. orientation should be made. If you are not using the full complement of memory, the right-most RAM sockets (IC31 and IC45) must be populated first in "vertical" pairs.

If all seems correct, connect up and switch on. Tune the TV to the computer somewhere around channel 36, and press *both* RESET keys *simultaneously.* **D/C/W/M?** should appear. Check that SHIFT LOCK is in the down position and press C.

If this causes **MEMORY SIZE?** to appear, and pressing RETURN a couple of times gives the start up message on the screen, then you have a working model of what is probably one of the most advanced computers for its price.

TROUBLESHOOTING

The troubleshooting process is best assimilated while the reader's mind is fresh from the hardware description. There are several categories of malfunction which may arise, and only one or two of a very definite nature can be mentioned here. The tools necessary for troubleshooting are an oscilloscope and a continuity tester. The latter may be all that is necessary but a 'scope considerably speeds the process.



UK101 COMPUTER

Fig. 2.1. Component layout of Compukit single board computer. The p.c.b. supplied by Comp Components has all component positions clearly printed on it. Most resistors are mounted on end. The kit includes TR1, R72, R63-65 serial interface components for a printer, but the remaining shaded components are not supplied. The small signal p.n.p. transistors can be of any type, and are not shown in last month's component list. Copyright of this p.c.b. belongs to Comp Componets, and is too complex to show here. Please note: R83 should read as R62, and R82 as R83

The following assumes that you have checked the five volt supply and all the external connections and that the SHIFT LOCK key is in the down (locked) position.

If the following procedures are ineffective, the unit should be returned to Comp Components, who have a standard charge of £25 (inc. postage) for repair.

(a) UHF Modulator Failure

This is detected by switching on and tuning the TV through the complete range, particularly near to channel 36, and finding no change throughout the band. (A short band of blank screen *should* be detected near to channel 36.) Check supply at modulator and connections, including ground-to-metal case. "Scope" the video input to the modulator—the waveform should be negative-going pulses 64µs apart, with some fast spikes (positive-going) in between. If this is not present, then either the UHF modulator or its connections are faulty.

(b) No Video Information At Modulator

Scoping through starts at the output of IC58 (pin 3) to detect the 8MHz clock. Work through the counter chain including IC29 to check on oscillation of counters. If this is absent, the sequential nature of the chain will allow you to narrow down the point of failure quite closely. The most common fault is a solder bridge or bent i.c. pin. If this cannot be visually detected, the continuity tester must be used to check that all pins go to the right place *and nowhere else!* A chip must be suspected of failure only as the very last resort. Even then, try a chip from elsewhere on the board in its place if possible.

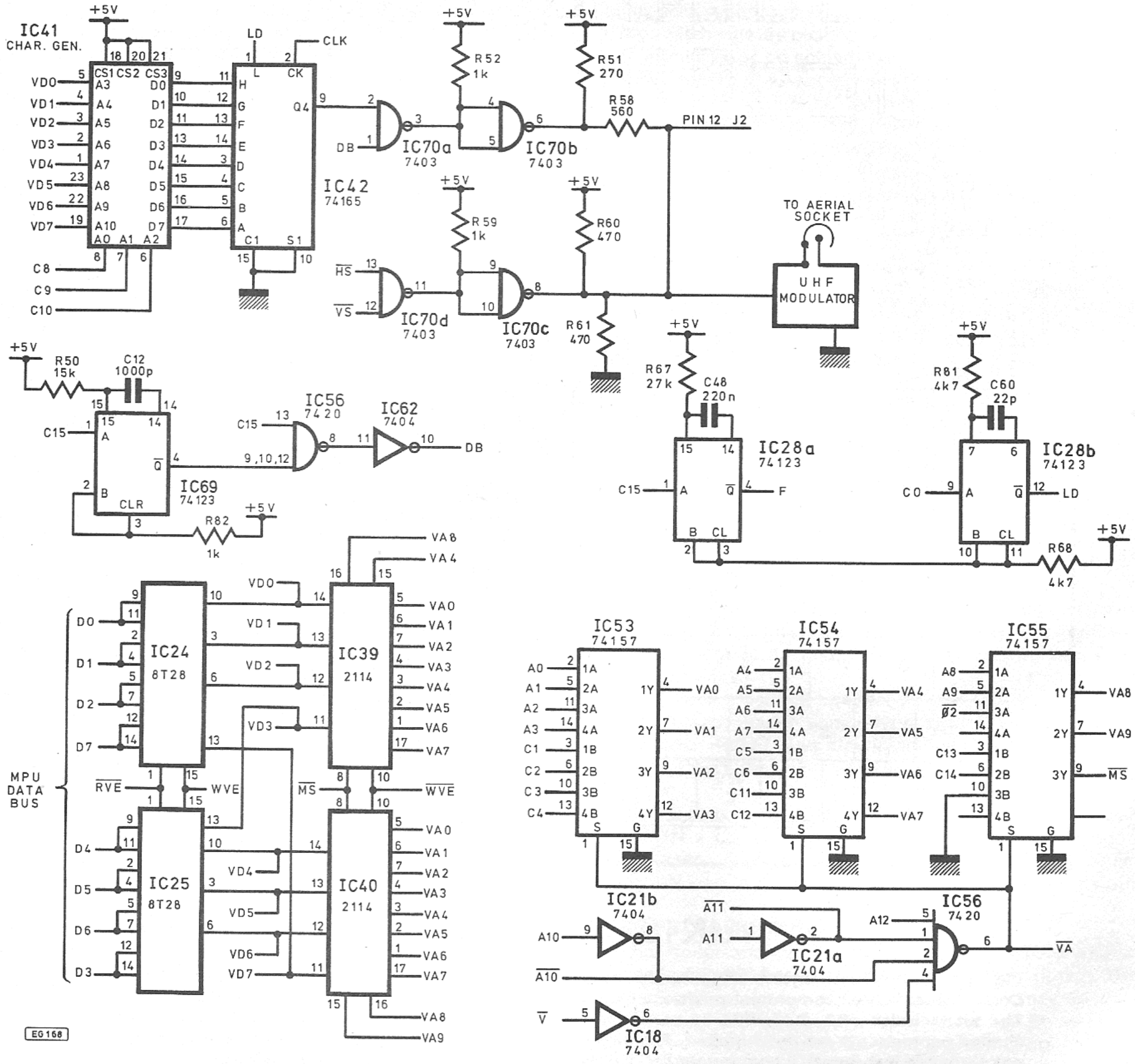


Fig. 2.2. Memory mapped VDU interface. All 7400 series devices are LS types

Once the chain is oscillating, failure may then be due to the area of the 74123 monostables IC65, IC71, IC69—again check through from the counting chains. $\overline{HS}$ should be negative-going pulses at 64µs separation (Horizontal sync), and $\overline{VS}$ at 20mS (Vertical sync). Pin 9 of IC42 should be pure video information in short closely packed spikes.

(c) VDU OK—No Reset

If the two break keys, pressed simultaneously, do not produce **D/C/W/M?** on the screen it is possible that the 6502 (IC8), is not receiving its clock (pin 37 at 1MHz) or its RESET (pin 40). Check both with scope.

The most likely cause, however, is almost always a simple bridge connecting a couple of Data Bus lines or Address Bus lines together. Check for any shorts between the pins of IC8.

All the Data and Address lines should be oscillating and should all be affected by pressing the two break keys. If not, check the relevant lines through from start to finish for shorts and lack of continuity.

If **D/C/W/M?** appears but pressing C has no effect, you have almost certainly failed to lock the SHIFT LOCK in the "down" position; *this must be checked every time a fault condition arises.* If this is not the answer, check that none of the keyboard switches are permanently shorted and check that the C key is working electrically. R0–R7 on the keyboard should be receiving a square-wave signal.

(d) Cassette Interface Not Receiving

The scope may be used to ensure that a sine-wave is present at the capacitor C10 and a square wave at pin 10 of IC69. The waveforms described for the cassette interface may then be checked through. The ACIA should be checked for clock information.

(e) Transmitting

Checking this side is confined to looking for a signal at the MIC and AUX outputs and then working back through the system.

(f) Adjustments To The VDU

A certain amount of adjustment of picture density is possible on R58 if required for contrast. Adjustment of the time-constant of IC71 by capacitor C48 and resistor R67 will move the picture up or down.

INITIAL USE OF THE MACHINE

Check that the SHIFT LOCK key is in the "down" position. *This should always form the first check if the computer seems inoperative at any time.*

The two RESET keys should be pressed simultaneously so that the following will appear in the lower left hand corner of the screen:

D/C/W/M?

This is a question requiring the user to reply via the keyboard with one of the four letters requested.

D is for disc operation and is not covered here. Now press M. This is for the machine code monitor, and six characters will appear near the middle of the screen—four for address and two for data (both in HEX).

This is explained in a later section and the user should now press the two RESET keys again to restore **D/C/W/M?**

Keys C and W are for COLD START and WARM START respectively and have the following meanings. If a program has been written and stored and is, say, in the operation of being executed, the user may RESET at any time. **D/C/W/M?** appears and pressing W (warm start) will revert the machine to its BASIC function without clearing its memory. Key C

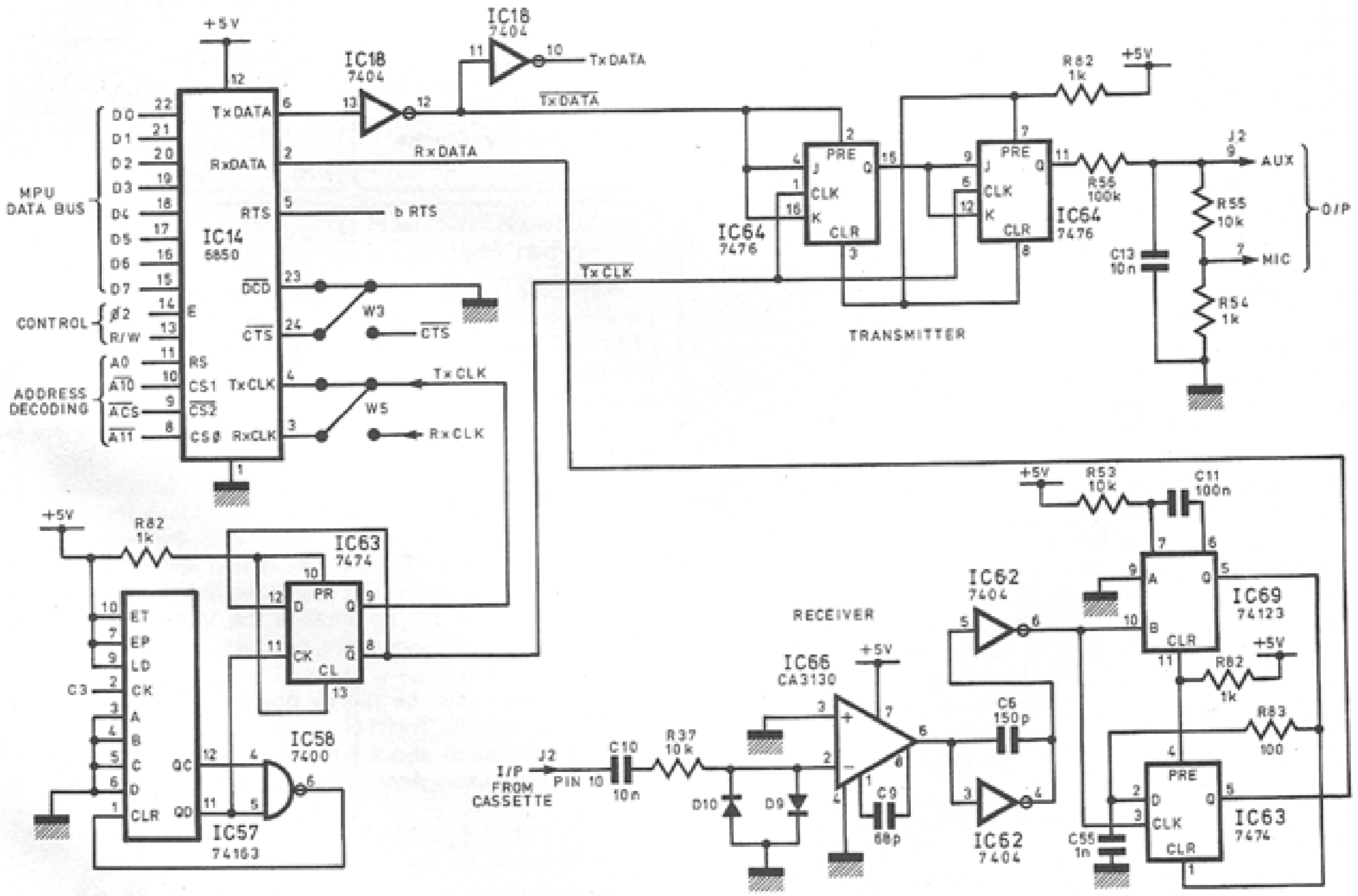


Fig. 2.3. Cassette Interface. 7400 series devices are LS types. Printed circuit "patch-panels" have W numbers. The ACIA signals ø2 and R/$\overline{W}$ go to expansion socket pins 31 and 32 respectively

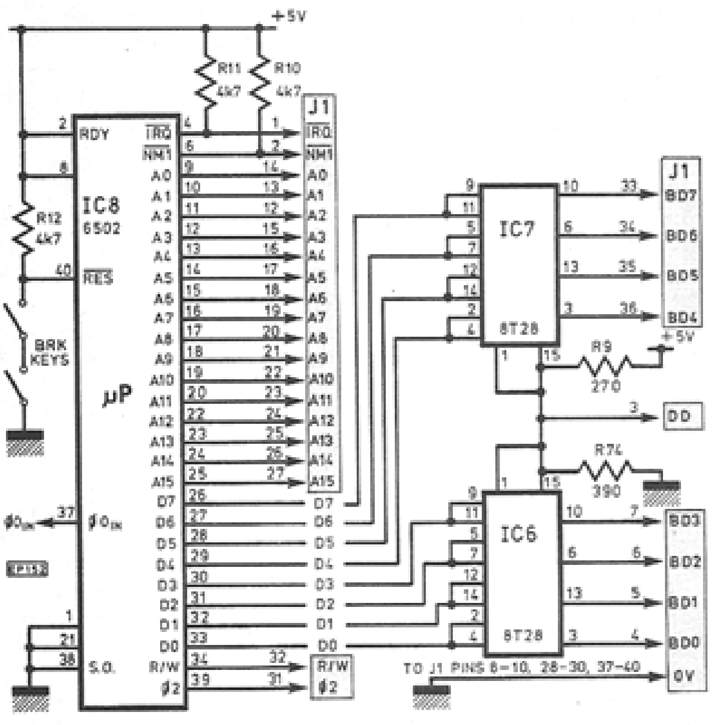


Fig. 2.4. The 6502 microprocessor and expansion socket J1. When pressed simultaneously, the BREAK keys reset the processor

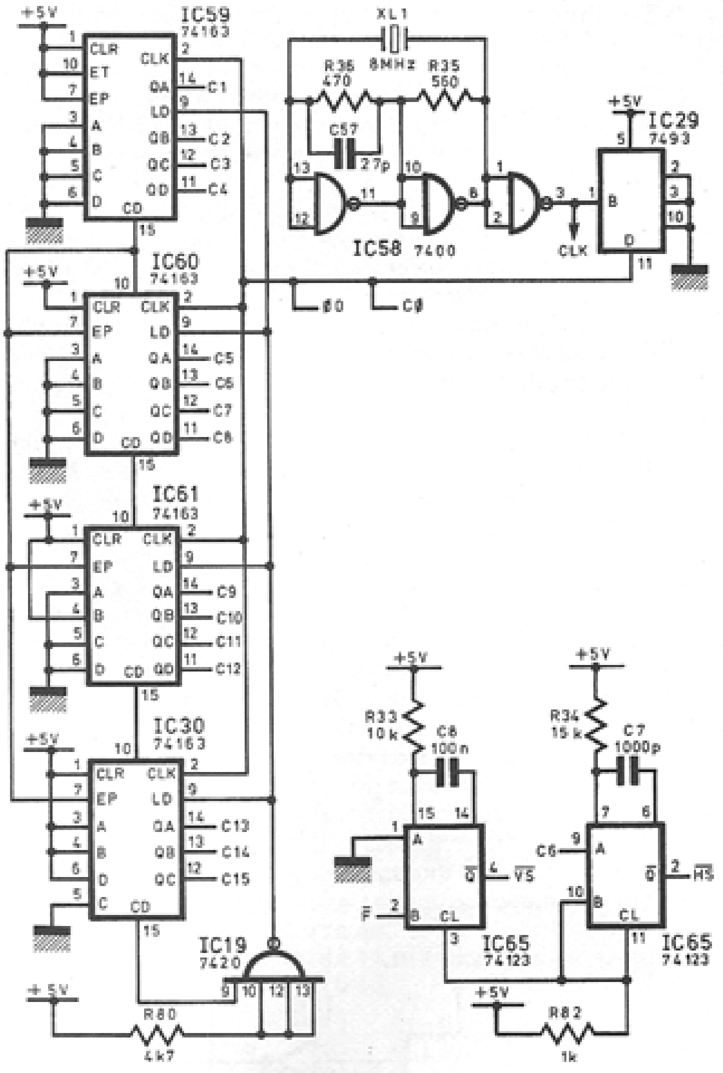


Fig. 2.5. System Clocks

restarts the computer "from the top" and should *now be pressed*–by the reader following this text. The words:

MEMORY SIZE?

should have appeared. If not, check shift lock. If there is no success, switch off and check the p.c.b. very thoroughly, *especially around the ROMS.* Typing any number after **MEMORY SIZE?** defines the number of bytes which may be used by BASIC from the start of RAM. The rest of the RAM is thus protected from being overwritten, and may be used to store data and machine-code blocks—accessible by PEEK, POKE, and the USR function defined later. Pressing RETURN, "defaults" to the full memory for BASIC—*this is jargon for saying that the computer automatically assumes you would have typed a number of bytes equal to the total memory available.* From now on, the computer will not look at any information until you press RETURN. This gives you time to change your mind about things and delete unwanted entries before the computer acts on them. The words:

TERMINAL WIDTH?

should have appeared now, and you are being asked to supply the number of characters across the screen to be printed before each new line starts.

Pressing RETURN defaults to 48, but not all of these characters would appear on a *normal* T.V. screen. Try typing 46 followed by return, this will fit comfortably on *most* T.V.s. At this point, the COMPUKIT does a complete scan of its RANDOM ACCESS MEMORY to determine how many bytes are free for writing in BASIC. This inbuilt memory test can be used to determine whether the memory chips are working correctly, i.e. 3324 bytes should be free in the 4K system and 7423 in the 8K system. The latter is given by the message:

7423 BYTES FREE

followed by:

COMPUKIT UK101
Personal Computer
8K BASIC Copyright 1979
OK

Well done!—you are now ready to start programming a powerful and versatile personal computer. With a little study of BASIC you will be able to persuade it to perform almost any activity for which you are able to write down a logical set of steps.

The "■" character is a CURSOR which tells you where on the screen your next keyboard entry will appear—*try it!*

The program:

```
10 PRINT "HELLO"
20 X=3·6*4·8
30 PRINT "X=";X
```

contains three program lines and three program statements. The first (labelled 10) commands the VDU to display the word **HELLO.** The second to calculate a value for X, and the third to print it.

The program may be run by pressing RUN (followed by RETURN as always). Try it!

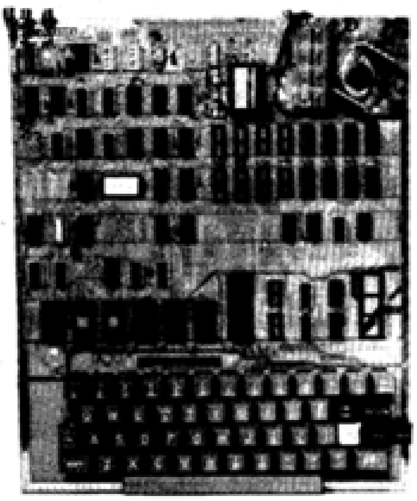
The central point about File Mode is that the program is retained after execution, plus all the variable values—try typing:

PRINT X (RETURN)

in immediate mode, and then RUN again.

NEXT MONTH: Error codes, program recording/playback, and using the machine in BASIC, plus remaining circuit diagrams.

COMPUKIT UK 101 SINGLE BOARD COMPUTER



PART 3 A.A. BERK B.Sc. Ph.D.

ERROR CODES

IF DURING the execution of a program, the computer encounters a word it does not understand, or is asked to perform an impossible calculation, it *may* detect an error of a recognisable type and so inform the user. Some errors are undetectable and simply produce wrong answers or bizarre behaviour. If it does recognise a *standard* error, it will print up one of the standard error codes listed in the table.

This *self-checking* activity makes computer programs easier to debug, but there are pitfalls, because sometimes, though an error of a particular type has been flagged, it may be the consequence of a more subtle error elsewhere in the program. Experience is the only answer!

Table 3.1. ERROR CODES

Code	Definition
D /	Double dimension: variable dimensioned twice. Remember subscripted variables default to Dim. 10.
F /	Function call error: parameter passed to function out of range.
I /	Illegal direct: INPUT cannot be used in immediate mode.
N \	NEXT without FOR.
O /	Out of data: more READs than DATA.
O ˥	Out of memory: program too big or too many nested GOSUBs, FOR NEXT loops or variables.
O ϟ	Overflow: result of calculation too large for BASIC.
S ┘	Syntax error: typing mistakes etc.
R \	RETURN without GOSUB.
U √	Undefined statement: attempt to jump to non-existant line-number.
/ ◢	Division by zero.
C ┘	CONTINUE errors: inappropriate attempt to CONT after BREAK or STOP.
L √	LONG string: string longer than 255 characters.
O √	Out of string space: same as O˥.
S ʃ	String temporaries: string expression too complex.
T ˥	Type mismatch: string variable mismatched to numeric variable.
U \	Undefined function.

EDITING

The program may be edited by writing further lines or rewriting existing ones.

Try typing the following (to be added to last program):

15 PRINT "BYE"

Typing LIST will insert this new statement between lines 10 and 20. Similarly, typing:

10 PRINT "THIS"

will *overwrite* line 10. Typing 10 followed only by RETURN will simply wipe out line 10 altogether. If a mistake is made in typing a character, it may be deleted by pressing the RUB OUT key, which if pressed several times will delete that number of previous characters. Try the following correction:

PRINT "HELLL (press RUB OUT) **O"**

The third L will be deleted and the VDU will show:

HELLO

The entire current line being typed may be deleted *before* RETURN is pressed, by pressing (SHIFT) P, which displays an @ sign and places the cursor on the next line to await further instructions.

USE OF CASSETTE

Check that pin 10 of J2 is connected to the earphone output, and pin 9 (or 7) to the auxiliary or microphone input, and pin 8 and/or 11 to the Earth of the cassette machine. Any ordinary cassette recorder should be suitable, but the very cheapest cassette *tapes* are prone to giving continual errors. The best volume setting is found by trial and error for playback. Recording may be on automatic level or manual. A machine having a tape counter is preferable.

PLAYING BACK A PROGRAM

(a) Rewind tape to "leader" or blank area before program starts.
(b) Place computer in command mode and type NEW (and RETURN).
(c) Type LOAD but *do not* press RETURN, to avoid spurious "noise" being interpreted as data and loaded to the computer.
(d) Switch recorder to PLAY.
(e) Wait for a second or two—or for the "leader" to pass through, then press RETURN.

Some random noise characters may still be printed on the screen, and if one of these is a number, it could be interpreted as a program line. However, without this unlikely event the program is printed on the screen line by line. When playback is complete, pressing SPACE and then RETURN returns the machine to BASIC and the new program is resident for listing or running.

RECORDING A PROGRAM

This assumes a BASIC program is stored in the Computer ready for saving on cassette.

(a) Rewind tape to blank noise-free portion of tape.
(b) Type SAVE (RETURN).
(c) Type LIST but do not RETURN.
(d) Switch cassette to RECORD and allow "leader" to pass, plus a further 5 seconds to allow settling to constant speed, then press RETURN. The program will list on the screen as it is being recorded.
(e) When recording is complete, wait a few seconds, turn off tape recorder and type LOAD (press RETURN), then press SPACE and RETURN.

USING THE MACHINE IN BASIC

After
OK
-
appears, the machine is said to be in the Command Mode. At this point, two types of data may be entered, *always terminated by pressing RETURN.*

(i) COMMANDS
(ii) BASIC Statements

These are described below:
N.B.
Spaces are always ignored in Commands and BASIC Statements (except in *literals* and *string* arguments).
If you make a typing error, press RUB OUT.

(1) COMMANDS

CLEAR This causes all variables (numeric or string) to be set to zero (or null)
LIST This can be used in several forms as detailed below:
LIST Causes the whole stored BASIC program to be listed line by line until either the listing is complete or CONTROL C is pressed.
LIST n Will list that line only
LIST n- Will list all lines from n to the end of the program
LIST -n Will list all lines up to n
LIST n-m Will list from line n to line m. This allows any part of a program to be viewed at will
NULL n Inserts n null before sending data to serial I/O devices
RUN Starts program execution from the first line with all variables cleared
RUN n As above but starts program at line n
NEW Wipes out current program
CONT Continues execution of program after Control C, or after a STOP statement encountered within the program
LOAD }
SAVE } Cassette commands dealt with elsewhere

CONTROL C

This is effected by pressing the "CTRL" key, and (with CTRL pressed) typing a "C". It suspends Computer activity and prints a message to give the line number at which the break occurred. The Computer then returns to COMMAND MODE. Many BASIC Statements may also be used as commands if unaccompanied by a line number—for instance:

GOTO n

would cause the Computer to begin executing from line number n without clearing all the variables. Similarly, many of the above may be used in programs—thereby causing a program to command the machine.

(ii) BASIC Statements

There are two modes of use of the BASIC language when using an *interpreter* such as this. These will be called:

(a) Immediate Mode
(b) File Mode.

IMMEDIATE MODE

If a BASIC statement is typed while in the Command mode, it is executed immediately a RETURN is encountered, and lost after execution. In this mode the Compukit, with its fast powerful floating-point calculation ability is able to act as a super calculator. For instance, answers to such calculations as:

$$X = \frac{15{\cdot}7 \times 13^{\sin(0{\cdot}781)}}{87 \quad \times 10^4}$$

are found immediately. In this case, the user would type:

PRINT **15·7 * 13 ↑ SIN (0·781)/87E4**

to get the answer:

1·09796E – 04

The immediate-mode use of the machine allows instant indication of remaining program space, by typing:

PRINT FRE (N)

The answer (after RESET) on an 8K machine should be 7420.

An important use of this mode is for program debugging. The last values of the variables are retained when a program stops. Type:

PRINT A, B, C, etc.

where A, B, C are the variables whose values are required. Quite complex immediate-mode programming may be written by employing colons to separate the various statements. In order to write and *retain* a program, the File Mode must be employed.

FILE MODE

To retain a program line for *later* execution, a line number must precede the instructions.

This numbered program line must not be confused with a display line. The computer accepts a maximum of 71 characters on a program line, and depending upon the Terminal Width set up after a system reset, the program line may occupy up to around four and a half lines of VDU display (if terminal width is 16).

Table 3.2. COMPUKIT MEMORY MAP

0000 — 00FF	Page Zero
0100 — 01FF	Stack
0130	NMI Vector
01C0	IRQ Vector
0200 — 0221	BASIC Flags & Vectors
0203	LOAD Flag
0205	SAVE Flag
0218	Input Vector
021A	Output Vector
021C	Control C Check Vector
021E	Load Vector
0220	Save Vector
0222 — 02FA	Unused
0300 end of RAM	BASIC Workspace
A000 — BFFF	BASIC-in-ROM
D000 — D3FF	Video RAM
DF00	Polled Keyboard
F000 — F001	ACIA Serial Cassette Port
F800 — FBFF	ROM
FC00 — FCFF	ROM — Floppy Bootstrap
FD00 — FDFF	ROM — Polled Keyboard Input Routine
FE00 — FEFF	ROM — 65V Monitor
FF00 — FFFF	ROM — BASIC Support
FFFA	NMI Vector
FFFC	Reset Vector
FFFE	IRQ Vector

MACHINE CODE MONITOR

The machine code monitor program provides a simple but adequate method of loading and running machine code routines, including loading from cassette. To prevent these routines being overwritten by BASIC, MEMORY SIZE? (After RESET) must be answered with a number restricting the BASIC's use of RAM. The number n, thus typed, restricts BASIC according to the following map.

Address in Decimal	Use
0 255	Page Zero
256 768	Scratch-pad RAM used by BASIC and system monitor
769 n-1	BASIC workspace
n End of RAM	protected against use by BASIC

It is clear from the above that n must be at least greater than 769. In a 4K machine, the end of RAM occurs at memory location 4095, and 8K finishes at 8191.

After RESET, the machine code monitor is entered by pressing M. The display:

0000 4C

then appears.

The first four characters form the address field, and the second two represent data (all in *hexadecimal* notation). Typing any hex characters at this point will load the address field; the data field is kept constantly updated as the address changes. Mistakes may be corrected by typing further characters, as these will continue to be loaded into the right hand position and then rotated left as further entries are made.

The following commands are available:

- **/** Changes to data mode to allow data to be loaded. RETURN then opens the next location.
- **.** Changes back to address mode.
- **G** (Used after setting up an address with **.**). This jumps to the address showing on the screen and begins execution.
- **L** Transfers control to cassette—loading 00FB with 00 transfers control back to the keyboard.

After L, the monitor is in data mode and simply accepts all its commands from cassette instead of the keyboard. Thus the cassette tape must have a series of commands, stored as ASCII codes from BASIC, to control the Monitor. To load a program from cassette, it must be stored byte by byte, separated by RETURNs and ending with:

. 00FB/00

This loads 00FB with 00 which is the flag to switch the monitor back to accepting commands from the keyboard. The program can be run from cassette, if desired, by ending with G after setting up the start of the routine in the address field.

The following gives a list of important address locations in the machine code monitor.

Starting address	Effect of jumping to address shown
FE00	Restart location. Ending machine-code programs with a jump to this location has the same effect as pressing M after D/C/W/M?
FE0C	Bypasses UART and Stack Pointer initialisation as well as clearing decimal mode, but still clears screen.
FE43	Enters directly into address mode. Bypasses initialisation and screen clearing.
FE77	As last, but for data mode.

The following are *subroutines* which may be of use in user programs.

Starting address	Effect of jumping to subroutine
FE80	Inputs an ASCII character from the cassette UART.
FE93	Returns stripped ASCII number if 0–9 or A–F otherwise returns FF.
FEED	Inputs an ASCII character from the keyboard.

To test the machine-code monitor, the message program used to illustrate USR (last month) may be adapted as follows.

Place the monitor in address mode either by pressing RESET followed by M, or by pressing *full stop* if already in the monitor. Enter the characters 0500 followed by / to access data. Type in the following pairs of digits—each pair separated by pressing RETURN.

```
A2  00  BD  00  06  C9  5F  F0
07  9D  E5  D1  E8  18  90  F2
4C  43  FE
```

This ends with a jump to location FE43 which places the monitor in address mode after the message has been displayed, thus preventing the clear screen routine in the monitor from erasing the message immediately after its appearance.

The following pairs of hex digits are ASCII codes for the characters of the message. The list may be of any length but must start at 0600 and end with the pair 5F.

Press *full stop* and the n type 0600 followed by / and the following pairs separated by RETURNs:

```
43  4F  4D  50  55  4B  49  54  5F
```

To run the program, type

.0500G

This will display the message for which the ASCII codes are given above, and leave the machine code monitor in address mode for further use. Memory size need not be specified unless BASIC is to be entered and the above protected against being overwritten.

GRAPHICS

Character resolution graphics are used by the Compukit whereby 255 different graphic characters are available to fill any given character slot. To view the available characters, the BASIC function CHR$ may be used by typing, for example:

PRINT CHR$(24)

followed by pressing RETURN. This causes a £ sign to be printed. Each number between 1 and 255 inclusive, corresponds to a character, as 24 does to £. (0 corresponds to a null character).

Two of these numbers correspond to *non-printing* commands for the print head (whose position is continuously shown by the cursor).

PRINT CHR$(10)

causes a line-feed, i.e. the cursor jumps to the next line and the screen scrolls upwards.

PRINT CHR$(13)

causes a carriage return.

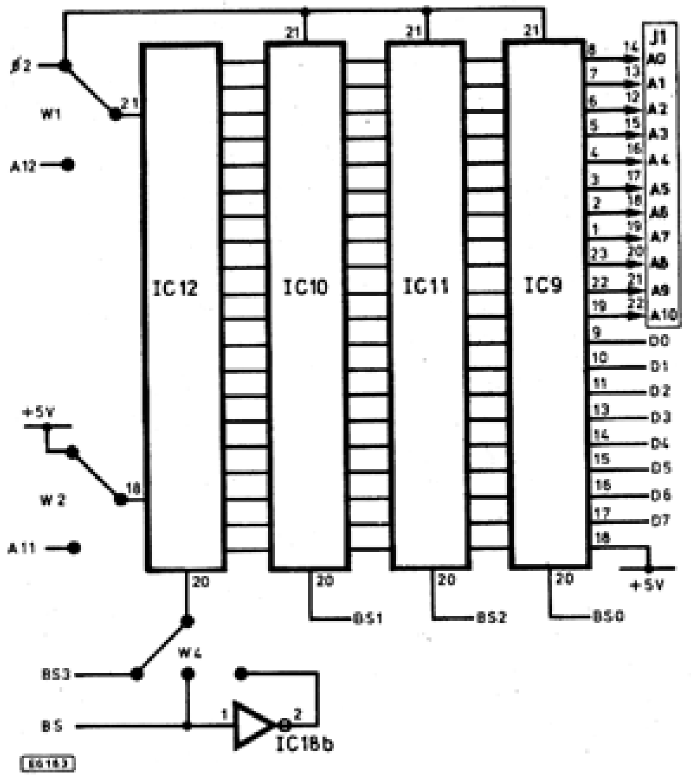
The rest of the numbers correspond to ASCII characters, special characters and graphic characters.

The ASCII characters start at 32 (SPACE) and finish at 127. These are all accessible from the keyboard. The upper-case is set with SHIFT-LOCK down, and lower case with SHIFT-LOCK up.

The other characters are inaccessible from the keyboard directly, and they must be printed using CHR$(I). These general graphic characters are best seen by writing a program to print them on the screen. This will be given later.

The following is a list of special (as distinct from *graphic*) characters, with their corresponding numbers:

number	character	number	character
0	null	244	δ
10	line-feed	245	Ψ
13	carriage-return	246	Ω
24	£	247	µ
32	Space	248	π
179	⇒	249	∢
180	⇐	250	λ
211	√	251	ϕ
212	∫	252	β
241	α	253	ε
242	β	254	γ
243	ω	255	ɣ



In order to select a particular graphic character, a list of those available may be displayed on the screen with corresponding number next to each one. The following program achieves this by allowing the user to specify which block of characters are to be displayed; there are too many to appear at once! The instructions for the program are as follows.

The program is loaded and run. The words:

WHICH BLOCK?

appear. Answer with a number between 1 and 4 inclusive followed by a RETURN. The numbers 1 and 2 display the graphic characters available, 3 shows the special characters already shown, and 4 displays the ASCII set.

To exit the program, just press RETURN instead of a number.

The line numbers chosen for the program put it well above any other program you may be working on. If the program under development ends with an END, then the following program will never be entered by the command RUN, and so RUN 10000 will be necessary. This allows the graphic program to remain in memory as a reference for use as necessary. It will be lost if NEW is typed or if RESET is pressed followed by C.

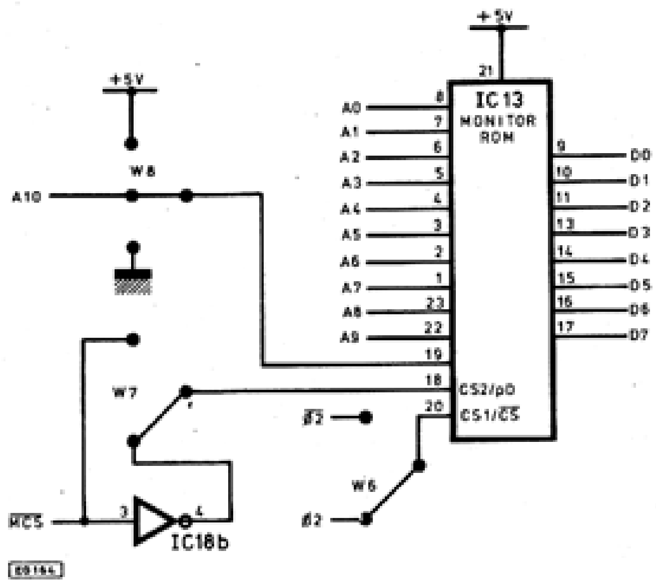
The Program Listing

(Spaces may be omitted, and PRINT may be typed as ?)

```
10000 INPUT "WHICH BLOCK" ; B : FL = 0
10010 IF B = 1 THEN S = 1 : F = 31 : GOTO 10060
10020 IF B = 2 THEN S = 128 : F = 219 : GOTO
      10070
10030 IF B = 3 THEN S = 220 : F = 255 : GOTO
      10060
10040 IF B = 4 THEN S = 32 : F = 127 : GOTO 10070
10050 GOTO 10000
10060 FL = −1
10070 FOR I = S TO F
10080 IF I = 10 OR I = 13 THEN 10110
10090 PRINT I; CHR$(I);:H = I + 3
10100 IF INT(H/7) = H/7 THEN PRINT : IF FL THEN
      PRINT
10110 NEXT
10120 PRINT
10130 GOTO 10000
```

Fig. 3.1 (left). BASIC ROMs

Fig. 3.2 (below). Monitor ROM



When Block 2 is requested, some of the vertically adjacent symbols run into each other. Use CHR$ in the immediate mode to inspect individual characters, e.g. :

PRINT CHR$(161)

reveals that this character fills the entire character slot.

The fact that characters run into each other in this manner allows the user to build up quite complex graphic patterns as well as graphs and bar-charts, etc. The user may find it useful to store the above program on cassette tape for future reference.

HARDWARE—BINARY COUNTING CHAINS

The clocking requirements for the system are supplied by the crystal oscillator and binary counting chains (see Fig. 2.5). Two gates of IC58, plus X1, form an 8MHz oscillator buffered by a further gate in IC58, and divided by 8 (IC29, which has a spare ÷2). Before IC29's ÷8 function, the CLK line feeds the Dot clock of the VDU with 8MHz. This governs the length of time available for displaying one of the dots of a character on the TV screen. Given the speed with which the electron beam strobes across the screen and the Dot time, the width of a dot may be calculated. A frequency of 8MHz gives a dot size sufficiently small to fit about 48 characters across the screen (each 8 dots wide), while of low enough frequency to pass easily through the UHF modulator and IF stages of a TV set.

The D output of IC29 (at 1MHz) then feeds the ϕ0in line of the MPU, the C0 line of the VDU, and the counting chain of 74163's (or 74161's) IC59–61 and IC30. The constraints on the counting chain are that it must produce ripple-count outputs for C1–C6 in between line-sync pulses separated by 64μs. Note: 2^6 = maximum of 64 characters per line. There must be three outputs (C8–C10) for the row inputs to the character generator, and a further four outputs (C11–C14) for the 16 horizontal lines of characters. The entire picture must then be repeated at 50 times a second with a suitable frame-sync pulse. The final count output from the bottom of the chain is then inverted, and fed to load the chain elements.

Capacitor C3 is used to set the BAUD rate for the Cassette and serial interface via a further counter, IC57, and some decoding logic IC63 and IC58.

THE VDU

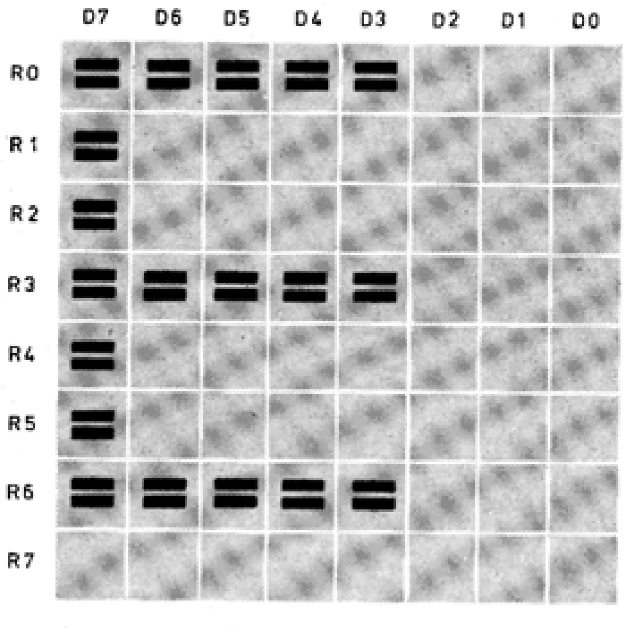
The block diagram of Fig. 1.2 shows the basic parts of the VDU and the circuit diagram (Fig. 2.2) gives the details referred to below.

The VDU RAM holds a screen full of characters (1024 in all). Through IC53–IC55 the RAM address lines (VA0–VA9) are either fed from the counter chain, or the MPU Address Bus, depending upon the state of $\overline{VA}$ (VDU Access). When $\overline{VA}$ is at "1", C1–C6 and C11–C14 are connected through to VA0–VA9, and when $\overline{VA}$ is at "0", the MPU busses have direct Read/Write access to the VDU RAM. Reading or writing of data is controlled by the bi-directional buffers IC24 and IC25, which also disconnect the VDU RAM from the MPU Data Bus when the counter chain is supplying addresses to VA0–VA9. Thus when $\overline{VA}$ is in the zero state, the VDU RAM acts just like any other block of Read/Write memory, here based at location address D000 hex. This allows the screen to be read or written to during a program. With $\overline{VA}$ at "1", the ten VDU RAM addresses are derived from the counters sequentially. The RAM is in the READ condition when not selected by the MPU, and the contents of the RAM locations are sent to the character generator for interpretation into bit patterns forming characters on the screen.

Each character in the Character Generator (IC41) is stored as an 8 × 8 matrix of white and black dots. White is stored as a "1". The characters appear on the outputs of IC41 (D0–D7) one row at a time, see Fig. 3.3. Here an "E" is being displayed on one of the 16 lines of text on the TV screen. C8, C9 and C10 from the counter chain determine which row (R0–R7) is being output at any time. The sequence of events is as follows: C1–C6, C11–C14 contain an address of a location in VDU RAM and hence of a character on the screen. The contents of this location (8 bits in parallel) are fed to IC41 which then parallel outputs the 1's and 0's (white and black dots) of one row of the character along D7–D0. Here, five 1's and three 0's are output to form the top row of the "E". IC42 serialises this parallel information at 8MHz, and sends it out in a stream to IC70 to be mixed with TV sync. information, etc. displayed along a TV line as the electron beam strobes across the screen. This takes 1μs, and each successive 1μs sees IC42 loaded with another character-row for the same treatment. Note: LD is fed from C0 to 1MHz via a monostable (half of IC71) to give a short negative going pulse, and CLK is at 8MHz. This is the Dot clock, so named because each cycle displays one of the 8 dots of a character on the screen. After the top row of the "E" has been displayed, the top row of the next character on that line must be fetched. Again, C8, C9, C10 will not change, but C1–C6 will, hence selecting the next VDU RAM location, and so on until C1–C6 have displayed one row of 64 characters. Some of these are lost at the end of the line, as the Dot clock is only at 8MHz. When C1–C6 have finished rippling through, C7 changes and the whole is repeated. C6 synchronises the TV line (at 64μs intervals) and thus starts a new line via IC65 on its downward edge. C7 is not used in the process and thus C1–C6 must count through twice before C8, 9, 10 increment to a new row of the character; this causes each row of dots to occupy two TV lines as shown in Fig. 3.3.

As C8, 9, 10 increment, the complete set of 16 TV lines builds up a row of text. The next step is to increment C11–C14 to address the next row of characters stored in the VDU RAM. The complete frame of 256 TV lines is built up as C1–C6 count through. Normally, in TV transmissions, another frame slightly different from this, is interlaced in the

Fig. 3.3 Dot structure of an ASCII Character on the VDU screen



spaces between the lines of the first frame. Also, each half frame is composed of more lines. Here, C15, via IC71, provides a frame-sync. pulse to the TV, and the above process repeats exactly—each line occupying its previous position. The resolution thus obtained is not as high as a normal TV picture, but is more than adequate for 16 lines of VDU information.

The frame-sync. is delayed by half of IC65 to allow the TV picture to be moved up the screen, and hence prevent the bottom left character from being lost. This is the most important character slot on the screen and must be displayed clearly. The value of R33 and C8 may be adjusted to ensure its readability on any TV.

About 48 characters are able to be displayed on a normal TV, and hence about 16 characters are lost from the edges of the screen. At least 5 are missing from the start of the line and the rest from the end. The software of the COMPUKIT thus uses just those slots from the 6th to the 54th to prevent loss of information. The others are available to the user, however, and may be forced into display by adjusting a TV or monitor to "underscan". The RAM locations are still perfectly valid and may be used as normal.

A note about graphics should be made at this point. Since an 8 × 8 matrix of dots is used for characters in general and only a 7 × 5 matrix is used for the ASCII characters, spaces of varying sizes are left between text characters, both horizontally and vertically. However, the COMPUKIT's character generator is very rich in blocks, lines and special patterns which use the full 8 × 8 array of dots. By this means, adjacent graphic characters may be chosen to run into each other, and graphs, large patterns, block diagrams, etc. may all be constructed from basic components. Also, some extra characters are included such as £, π, ♠ etc. for a very full variety of uses.

ADDRESS DECODING AND MEMORY

Address decoding is performed via 74138's and 74139's with some extra gating. The address map defines the operation of this block and it is described here in full electrical detail. A TTL data book will provide all the information necessary to understand how this block works. $\overline{RS0}$–$\overline{RS7}$ are *selects* for the RAM (8 blocks of 1K, each comprising two 2114's). BS0–BS3 select the BASIC ROMs, and $\overline{MCS}$ selects the monitor ROM. $\overline{ACS}$ selects the ACIA for the cassette. RKB and WKB are Read and Write selects for the keyboard and WVE and RVE for the VDU.

The RAMs are addressed so that IC31 and IC45 are at the lowest addresses and hence form the first 1K block of RAM (based at 0000). Addresses increase from right to left in pairs (the 2114 being arranged as 1K by four bits), IC32 and IC42 are next and so on. The ROMs are arranged to allow other options. When the 64K bit ROM is available, the four BASIC ROMs may occupy one package. A11 and

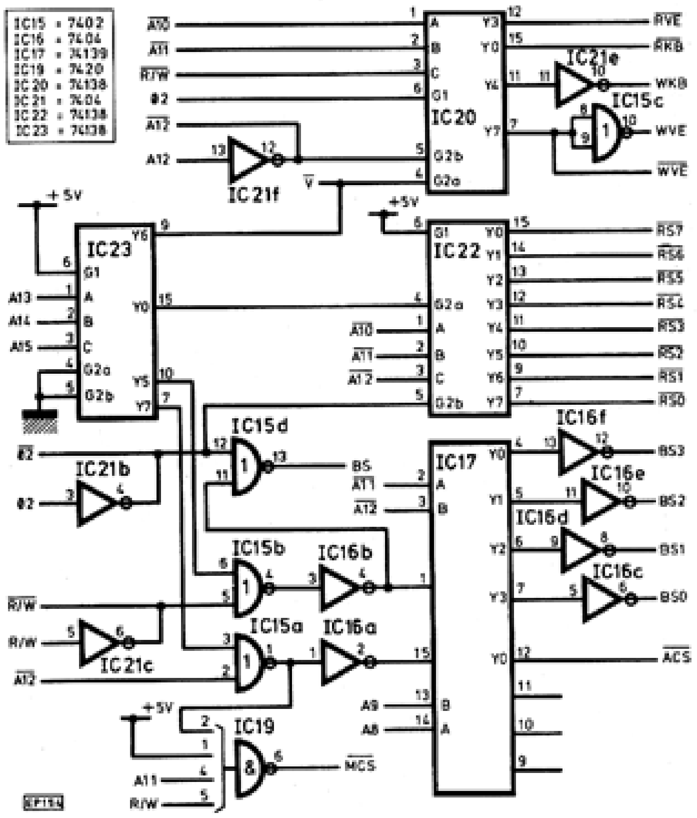


Fig. 3.5 (above). Address decoding

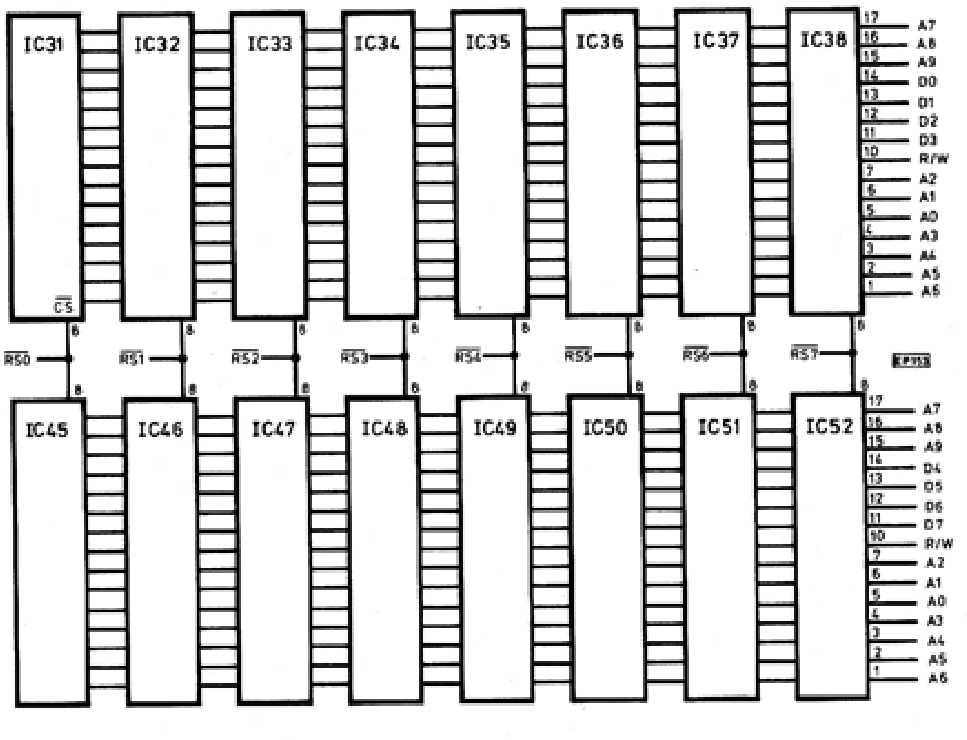


Fig. 3.4 (left). RAM configuration for 8K bytes

A12 will be needed and an address decoded line to select it. This already exists on the COMPUKIT; BS supplying the necessary address decoding. W1, W2 and W4 are pads next to the ROMs bringing these lines in. When this option is available, there will be three spaces free for ROMs or EPROMs of the user's choice. The COMPUKIT even allows for an active high or low BS line via IC18.

The Monitor ROM also has some flexibility in packaging and this is catered for as shown.

PROCESSOR AND EXPANSION SOCKET J1

The processor is shown (Fig. 2.4) feeding all the Busses and control lines internally as well as externally, via J1, whose data lines are fully buffered by IC6 and IC7. External devices decide the direction of data flow through these buffers by DD. This socket allows any external logic to overtake the MPU system via interrupts and can easily be extended to control anything. External memory may be added via the socket; disc storage, S100 Bus expansions, etc. may all be plugged in directly.

SERIAL AND CASSETTE INTERFACE

The serial interface is controlled by IC14 (ACIA). This is primarily to drive a cassette interface. However, sockets and pads are provided for extra components to allow the ACIA to drive an RS232 interface if required. This will not be described here but is shown in the diagram.

The ACIA receives its clock from C3 of the counting chain via IC57, IC63 and IC58. Options exist, as shown, to separate the Tx and Rx clocks. In addition, driving the clock from C2, C1 or C0 will increase and BAUD rate from 300 by a factor of 2, 4 or 8 respectively.

The ACIA's Tx and Rx data lines are fed to the cassette interface as shown. The transmitter uses a 7476 (IC64) to present a high or low tone to the recorder as a "1" or "0" to be recorded. This follows the usual Kansas City recording format.

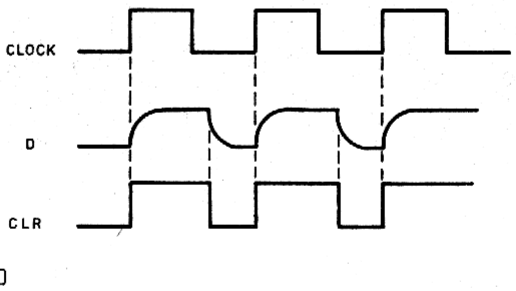


Fig. 3.6 Timing diagram

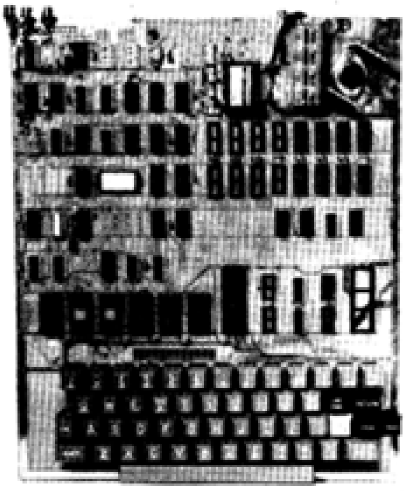
Receiving depends upon the time-constant of a monostable. IC66 and IC62 are used to convert the sine-wave input, from cassette, to a square-wave suitable for the monostable IC69, and the clock input of a D-type flip-flop (IC63). While the tone is high, the 74123's time-constant is set such that the Q-output has no time to reset to zero before the next positive edge at B forces it high again. D and CLR of IC63 thus remain high, as Q does, and Rx DATA presents a constant "1". When a low tone arrives, the cycles arriving at B are long enough to allow Q to reset, after its positive-going timing pulse, before B encounters a further positive-going edge forcing Q high again. This gives the timing diagram shown in Fig. 3.6 for IC63.

The leading edge of D is slowed by R62 and C55. The zero on CLR now sets Q to zero and, because D's rising edge is slowed down, IC63 sees a zero on D when the clock goes high, thus preserving the zero on Q, and hence the circuit decodes a constant zero for as long as the low tone continues.

This sort of circuit is quite reliable at 300 BAUD and any instability will be due either to a large variation in tape speed, or to the value of R53 and C11 having been incorrectly chosen, thus allowing the negative-going edge on D to arrive too soon.

NEXT MONTH: Conclusion of series

COMPUKIT UK 101 SINGLE BOARD COMPUTER



PART 4 A.A.BERK B.Sc. Ph.D.

CONCLUSION OF SERIES

AS INDICATED in the previous articles, the Compukit is hardware expandable in many ways. Expansions to the machine are in the process of being produced and include a Colour Graphics Board, and a large memory board which will bring the machine nearer to its maximum addressing ability in RAM and EPROM/ROM. By the time this article appears, these boards should be available.

Software expansions include a sophisticated Machine Code Monitor, disassembler and assembler which is included with the machine. Many programs, including games, already exist and it is hoped that others will become available in the near future from those software houses which have shown interest in the machine.

The hobbyist who wishes to expand the hardware of the machine himself may be interested in two useful and important methods of doing so. These are described below. Firstly, it is essential to bear in mind a picture of the machine's memory map:

When expanding the system or adding I/O ports, certain addresses are used for specific functions and must not be overlapped. It is also important to allow for future expansions by keeping clear of memory space which may eventually house extra RAM. In general, later expansion boards will add memory consecutively with the 8K of on-board RAM, so that the BASIC interpreter will find it during the usual memory test when C is pressed after Reset.

To decide on suitable addresses, the Address Map shown in Table 4.1 should be consulted. The addresses given are in hexadecimal notation and, as can be seen, BASIC workspace can be as large as 40,191 bytes (0300 to 9FFF) before overlapping with the BASIC ROMs. There is plenty of space from C000 to CFFF (4096 bytes), and from D400 up to F7FF, apart from the keyboard and ACIA. It is in this last portion that parallel or serial I/O ports can be neatly added with less danger of interfering with later additions.

Table 4.1. Memory map.

Address	Function
0130	NMI vector
01C0	IRQ vector
0000–02FF	Scratchpad RAM for operating system
0300	Start of BASIC workspace
1FFF	End of on-board RAM (8K)
A000–BFFF	BASIC interpreter
D000–D3FF	Video RAM
DF00	Polled keyboard
F000, F001	ACIA serial port
F800–FFFF	MONITOR ROM

In order to expand the Compukit, therefore, a certain amount of address decoding is necessary to locate any peripherals at the right place in the machine's memory. There are two main ways in which this may be achieved. The top 1K of the 8K of RAM (ICs 38 and 52) may be left unused and its address decoded output (RS7) supplied to the expansion as an $\overline{\text{ENABLE}}$ signal. A more general method is to add an expansion board to the system containing its own address decoding. Both of these are described.

PARALLEL I/O PORTS

Suppose we wish to add 16 I/O ports to the machine in the most straightforward manner possible for some control purpose. A 6820 or 6821 (with greater drive) is most suitable for the job. This chip is the famous PIA or Peripheral Interface Adapter containing a number of I/O drivers and latches as well as several control lines.

A couple of d.i.l. plugs will allow the circuit in Fig. 4.1 to be connected to the Compukit with minimum effort to control almost anything. The circuit shown includes some lights and switches—just imagine the l.e.d.s to be relays and the switches to be sensors of some kind.

To use Fig. 4.1 and appreciate the circuit's full potential, the data sheet for the device must be obtained and studied. This is a very useful chip and each of its sixteen I/O lines (PB0-PB7 and PA0-PA7) may act as either input or output. There are four external control lines for various purposes (CB1, CB2, CA1, CA2), and interrupts, via IRQ, may be generated by external devices. In order to use the chip, which looks life four memory locations (here decoded as 1C00, 1, 2, 3) internal registers must be set to a pattern of bits which informs the device of those lines which are to be inputs, and which are outputs. Data to be written to outputs is sent to the appropriate location within the PIA which subsequently clocks it through to the output latches. Similarly, incoming information is stored in a register and may be retrieved by reading the correct memory location in the PIA at the program's convenience. The Interrupt structure may be used to force the MPU to "look" at the PIA when an external device sends its information through.

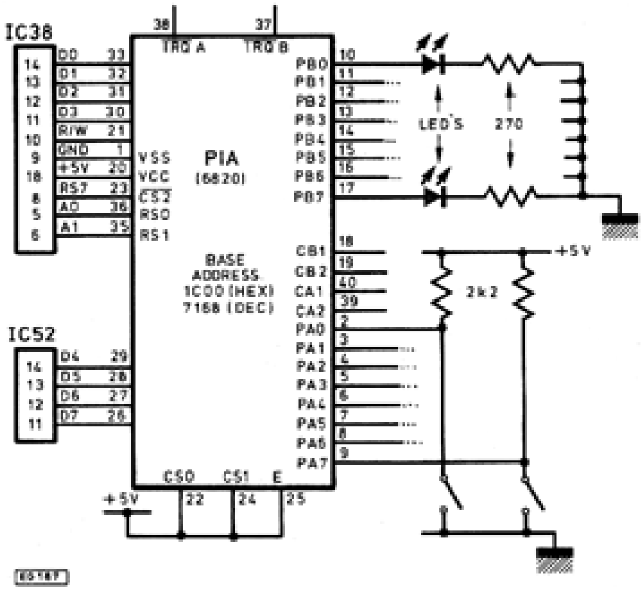

IC38
D0 D1 D2 D3 R/W GND +5V RS7 A0 A1
VSS VCC CS2 RS0 RS1
PIA (6820)
IRQ A IRQ B
PB0 PB1 PB2 PB3 PB4 PB5 PB6 PB7
LED'S
270
+5V
BASE ADDRESS 1C00 (HEX) 7168 (DEC)
CB1 CB2 CA1 CA2
2k2
PA0 PA1 PA2 PA3 PA4 PA5 PA6 PA7
IC52
D4 D5 D6 D7
CS0 CS1 E
+5V

Fig. 4.1. Switches and lamps interface

An interesting feature of such a device is that the time taken to change an input line to an output is similar to the instruction speed of the MPU system driving it. This allows the possibility of swopping between input and output very fast to make any given line (or lines) appear to the operator as if it is performing both functions simultaneously. Handshaking between microcomputers can be arranged in this way, and parallel processing by a set of machines may be envisaged.

The interface in Fig. 4.1 may, of course, be adapted to run many other devices including UARTs, USARTs or just tri-state buffers and TTL latches. In fact, by using the lower ten address lines from IC38 and IC52, any 1K (or less) memory mapped device may be attached to the Compukit. So far, the author has successfully driven the PE VDU and the coming EPROM Programmer. The advantages of using BASIC to control these devices are enormous. Tasks which appear most daunting when a machine code microcomputer is used, become almost trivial in the high level language.

Several extra terminals may be added to the basic machine in this way, adding considerably to the system's viability in small business applications.

However, the above expansion method, though quick and easy to implement, does tie up 1K of on-board RAM for each expansion used, and as such may be regarded as wasteful in the long run. This should lead the user to attach his peripherals to the machine via the 40-pin d.i.l. socket J1. The specifications for this socket are given in Fig. 4.2. All bus lines are brought out to the socket, and they allow external memory mapped devices to communicate with the MPU directly. The BD lines are buffered data lines, with direction controlled by the DD signal which selects Read or Write through ICs 6 and 7 as shown in Fig. 4.2.

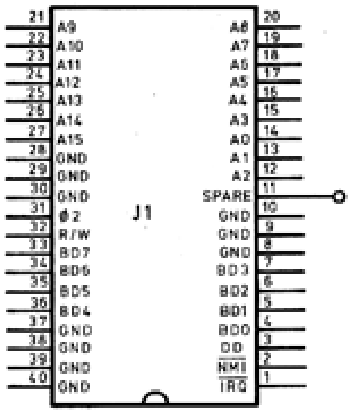
All sixteen address lines are present, as is R/W, IRQ, NMI and Ø2. For correct memory timing this last signal should be fed to an *active high* enable line from all external devices. There is also one spare line not connected to anything, but brought out to a pad next to the socket. The rest of the pins are Ground connections.

To use this socket, each external device must generate its own address decoding, the details of which depend upon the amount of memory each expansion takes up. Any such device should occupy a unique address position and hence each address line must be involved in its "fetching".

For devices taking up 8 bytes of memory, for instance, Fig. 4.3 gives a straightforward method of decoding. Here, the 8 bytes are arranged to lie at F400 to F407.

This particular circuit is, of course, purely a functional *suggestion* to highlight the fact that NAND gates decode 1's and NOR gates decode 0's, and that all address lines play some part in decoding the base address of F400.

Thus, small memory requirements are easily catered for using the simplest logic devices. It is usually a good idea to use CMOS or LS i.c.s to reduce bus loading.


A9 A10 A11 A12 A13 A14 A15 GND GND GND Ø2 R/W BD7 BD6 BD5 BD4 GND GND GND GND
J1
A8 A7 A6 A5 A4 A3 A0 A1 A2 SPARE GND GND GND BD3 BD2 BD1 BD0 DD NMI IRQ

DD	DATA-BUS DIRECTION
0	MPU READS FROM EXTERNAL DEVICES
1	MPU WRITES FROM EXTERNAL DEVICES

Fig. 4.2. Expansion connector

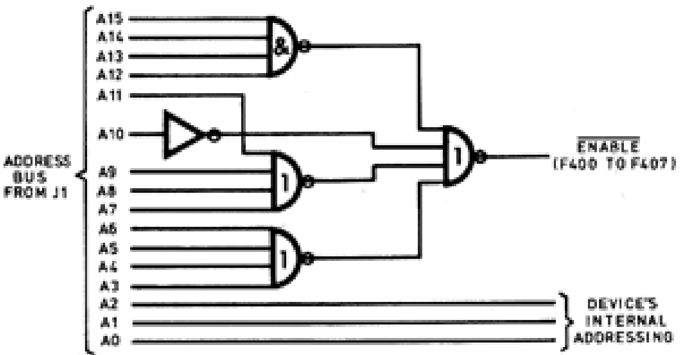

A15 A14 A13 A12 A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0
ADDRESS BUS FROM J1
&
1
ENABLE (F400 TO F407)
DEVICE'S INTERNAL ADDRESSING

Fig. 4.3. Address decoder for 8 bytes

For large memory requirements, the n to 2^n line decoders such as 74LS138,9 and 74154 are extremely useful. A 24K memory board, for instance, will need its own internal page-select logic to enable different banks of memory i.c.s, just as the Compukit itself does, via some of the above decoders.

For general hardware control purposes, the Compukit may be operated in either BASIC or machine code. The latter can be considerably faster as it deals fundamentally with *electrical* steps. From BASIC, an I/O port can be controlled by the WAIT statement or using PEEK and POKE. This has the advantage of extreme ease of programming. Imagine controlling a home-security system. The program could continually PEEK a number of I/O ports (PIA perhaps) connected to remote sensors. When a change occurs, an IF statement would decide whether to act, and a few subroutines would decide which action to take. Some other sensors could be PEEKed nearby and, in a short time, an alarm could be sounded or a stream of appropriate invective produced via a speech synthesis unit!

Extremely complex programs with feedback and analysis could be constructed using the powerful BASIC involved, which, though not as fast as machine code, would act many orders faster than any human activity involved.

The speed problem becomes important, for example, in controlling high speed machines or processes. Then, a hybrid program using the USR function could swop back and forth between BASIC and machine code for *instant* response to requirements.

As the IRQ and NMI interrupts are fully available to the user, an even more sophisticated system is possible whereby the external process takes control of the computer, when needed, via an interrupt.

The potential is exciting and to some extent already being exploited. Anyone interested could do worse than construct an "l.e.d. and Switch" I/O expansion as in Fig. 4.1 and learn to use it! The next step would be to add a D/A or A/D converter and learn to control and receive analogue data in real time.

SPECIAL KEYBOARD FUNCTIONS

Referring to the keyboard matrix circuit diagram and hardware description, it has already been stated that the keyboard is polled in software for key closures except during program execution (unless waiting for an INPUT).

There are two important routines associated with the polling sequence. One determines which key has been pressed, and the other is a routine for detecting CONTROL C. The latter is not in general disabled during program execution, and may be used to BREAK a program for examining variables, etc. The routine involved may, however, be disabled or enabled by the user via the following statements:

POKE 530,1 disables
POKE 530,0 enables

The first of these may be placed before a part of the program whose execution it is important not to be able to interrupt. If the second statement is placed at the end of the protected region, then CONTROL C will never intrude on that region if pressed.

Table 4.2.

COLUMN	ROW	CA/RA
C0	R0	254
C1	R1	253
C2	R2	251
C3	R3	247
C4	R4	239
C5	R5	223
C6	R6	191
C7	R7	127

The keyboard matrix may be used in special applications during the execution of a program, by treating it as an ordinary read/write memory location (57088 decimal or DF00 hexadecimal). To do this, it is often important to disable the CONTROL C routine to prevent it from interfering.

An example of the keyboard's use for special functions could be to allow the keys to be reprogrammed to return graphics characters. A program would be written to allow, say, all the "block" characters to be called from a section of the keyboard when SHIFT LOCK is up.

To perform any special programming of the keyboard, the following statements are used:

POKE 57088, RA
IF PEEK(57088) = CA THEN (statement)

RA is the *address* of the row being tested for key-closures according to Table 4.2. This POKE statement may be thought of as "setting" the appropriate row to the "on" condition. CA, column address, is the value which location 57088 takes on when a key in the row RA is pressed. Thus if 57088 is POKEd to have value 254, and 57088 is then read (via PEEK) and found to have value 254 then the program knows that SHIFT LOCK is down (see the keyboard matrix diagram).

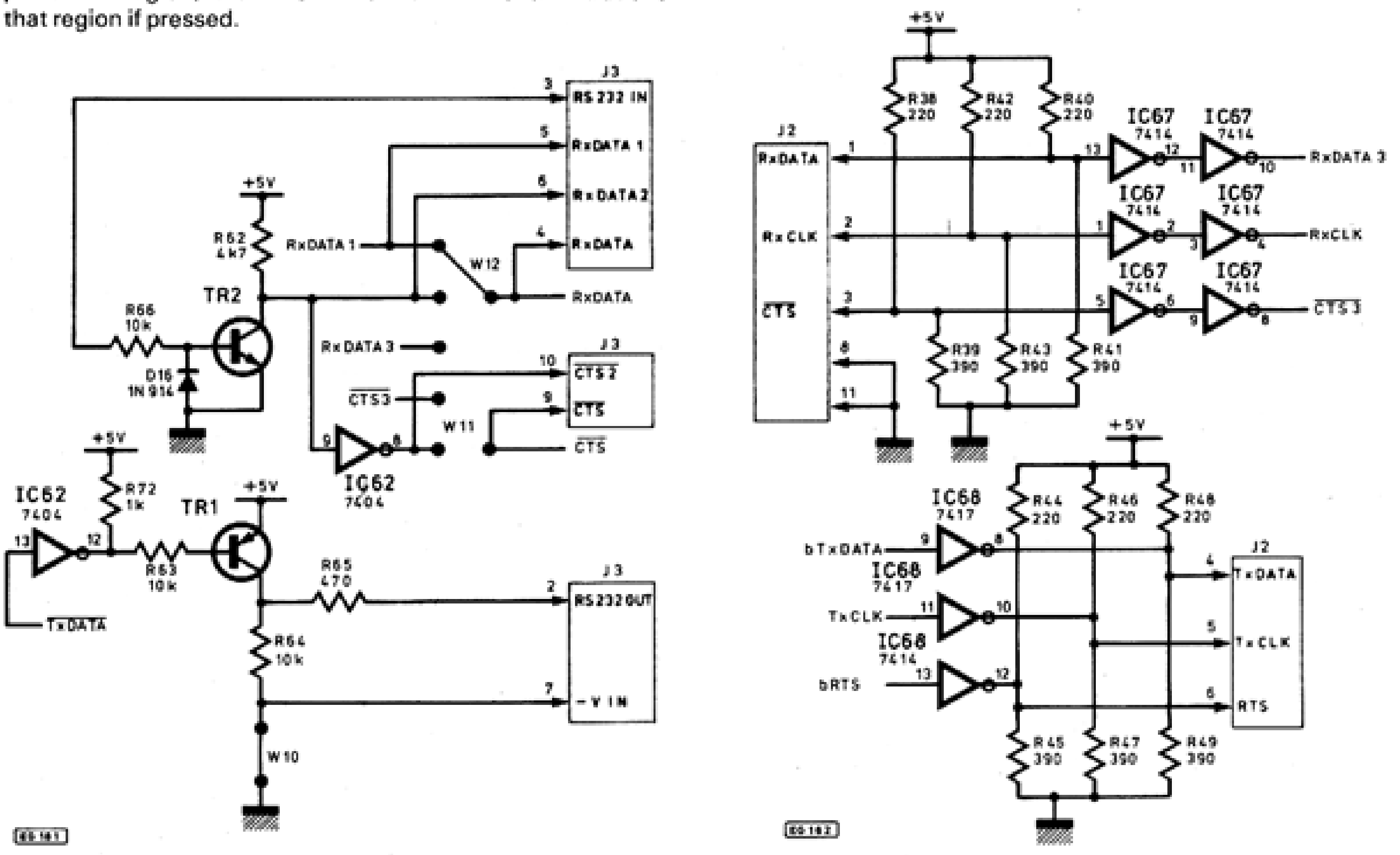


Fig. 4.4. (left) Asynchronous I/O and RS232. (right) Serial data buffers

The following program changes the keys 1 to 7 to graphic characters when SHIFT LOCK is up. When down, the words SHIFT LOCK roll up the screen until SHIFT LOCK is pressed. Then the keys 1 to 7 are active and each gives a different character until 7 is pressed when the program terminates.

```
10 POKE 530,1
20 POKE 57088,254
30 IF PEEK(57088) = 254 THEN PRINT "SHIFT-
LOCK DOWN": GOTO 20
40 POKE 57088,127
50 IF PEEK(57088)=255 THEN 40
60 PRINT CHR$( PEEK(57088) );
70 IF PEEK(57088)=253 THEN END
80 GOTO 10
```

The program, though rather simple, is meant to illustrate how information can be gathered from the keyboard and used to control execution. Note that in line 50 the keyboard location is assumed to have value 255 unless a key is pressed, as R1–R8 pull up the inputs to IC4 and IC5 and force them to "see" 1's until an active key is pressed.

The applications of the above are manifold, not least in the execution of games or simulation exercises; two areas which in many ways are very similar!

In order to use this keyboard polling easily, it is a good idea to label the keyboard matrix diagram, published in Part 1, with the CA and RA addresses corresponding to the columns and rows. For instance, C0 and R0 should be labelled 254, C1 and R1 253, etc.

This concludes the description of the Compukit UK 101. By the time this article appears, many readers will have had the opportunity of operating the basic machine. The applications are enormous and stretch across the full gamut of endeavour. It is hoped that through the pages of *Practical Electronics,* future developments can be described as they occur and thus keep readers up to date with a machine considered to be ahead of its time.

It only remains for us to wish you all good luck with the project. ★

COMPUKIT CIRCUIT DIAGRAMS

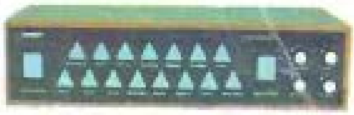
Fig. No.	
2	Block diagram
3	Keyboard matrix and interface
4	Power supply
2·1	Component layout
2·2	VDU interface
2·3	Cassette interface
2·4	The uP and expansion socket
2·5	Clocks
3·1	BASIC ROMs
3·2	Monitor ROM
3·4	8K RAM
3·5	Address decoding
4·2	J1 pin-outs
4·4	Serial interface

MORE!

Next month we will publish Part 1 of an EPROM programmer designed to plug into Compukit—it will also function with other computers. We also expect to be able to publish another exciting computer peripheral in the near future—we do not believe this has previously been published as a hobby design—more details in future issues.

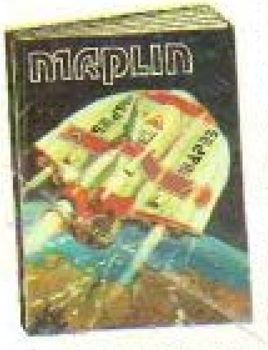
MAPLIN

This superb organ – build the first working section for just over £100. Full specification in our catalogue.


Touch operated rhythm generator, the 'Drumsette'. Construction details 25p. (Leaflet MES49). Specification in our catalogue.

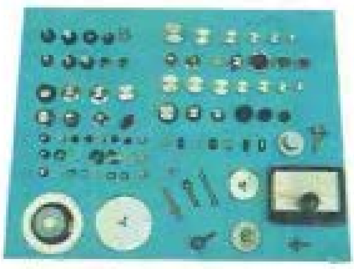
Multimeters, analogue and digital, frequency counter, oscilloscopes, and lots, lots more at excellent prices. See cat. pages 106 and 183 to 188 for details.

61-note touch-sensitive piano to build yourself. Full specification in our catalogue.



A massive new catalogue from Maplin that's even bigger and better than before. If you ever buy electronic components, this is the one catalogue you must not be without. Over 280 pages – some in full colour – it's a comprehensive guide to electronic components with hundreds of photographs and illustrations and page after page of invaluable data.

Our bi-monthly newsletter contains guaranteed prices, special offers and all the latest news from Maplin.


A range of highly attractive knobs is described in our catalogue. Our prices are very attractive too!

The 3800 synthesiser build it yourself at a fraction of the cost of one ready-made with this specification. Full details in our catalogue.

A pulse width train controller for smooth slow running plus inertia braking and acceleration. Full construction details in our catalogue.

Speakers from 1½ inch to 15 inch; megaphone. PA horns, crossovers etc. They're all in our catalogue. Send the coupon now!

Post this coupon now for your copy of our 1979-80 catalogue price 70p.

Please send me a copy of your 280 page catalogue. I enclose 70p (plus 37p p&p). If I am not completely satisfied I may return the catalogue to you and have my money refunded. If you live outside the U.K. send £1.35 or ten International Reply Coupons. I enclose £1.07

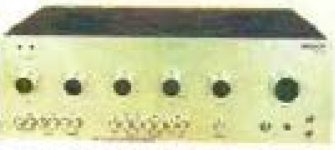
NAME ______________________

ADDRESS ______________________

____

PE 11/79

A wide range of disco accessories at marvellous prices. Our catalogue has all the details.


A very high quality 40W per channel stereo amplifier with a superb specification and lots of extras. Full construction details in our catalogue.

A genuine 150W per channel stereo disco to build yourself. Full specification in our catalogue.

MAPLIN

ELECTRONIC SUPPLIES LTD

All mail to:–
P.O. Box 3, Rayleigh, Essex SS6 8LR.
Telephone: Southend (0702) 554155.
Shop: 284 London Road, Westcliff-on-Sea, Essex. (Closed on Monday).
Telephone: Southend (0702) 554000.

P.E. 9/79