MICRO-BUS

Compiled by DJD.

Appearing every two months, Micro-Bus will present ideas, applications, and programs for the most popular microprocessors; ones that you are unlikely to find in the manufacturers' data books. The most original ideas will probably come from readers working on their own microcomputer systems, and payment will be made for any contribution featured here. This is also the place to air your views, in general, on this new technology, so let's be hearing from you!

THE main topic in this month's Micro-Bus is a design for an extremely simple SC/MP-based microprocessor system which, while using a minimum of components, makes it possible to run and debug programs. Also included are designs for a hex keyboard and a two-digit hex display which can be added to the system.

NINE PROBLEMS

But first, here are nine light-hearted problems each to do with some aspect of programming micros, and gathered from a variety of sources. Solutions to all the problems will be presented in the next Micro-Bus.

One. National Semiconductor has just developed a micro with four registers, labelled A, B, C and D, and an instruction set consisting of the following five instructions (where X and Y stand for any of the four registers, and L represents a label):

LD X, Y	Load X with the value in Y
DEC X	Subtract 1 from the value in X
JZ L	Jump to L if result of previous DEC was zero
JNZ L	Jump to L if result of previous DEC was non-zero
DIS X	Display value of X

Write a progam for this rudimentary microprocessor, using as few instructions as possible, to display the highest prime factor of a number in the A register. For example, for 91 it should give the result 13, and for 19 the result 1.

When you have reached a solution you are advised to translate it into BASIC, or the machine code of a more reasonable micro, and run it to check that it really does work.

Two. The following problem has no possible practical application, but it should nevertheless cause some head-scratching among SC/MP programmers:

On SC/MP the obvious way to load zero into the accumulator is by executing 'LDI O' (C4 OO). Without making any assumptions about the contents of any of the registers, can you find four other ways of clearing the accumulator in just two bytes?

Three. It is very easy, in BASIC, to print the larger of two numbers by using an 'IF' statement and a 'GOTO', but how can it be done in a single statement, and without using 'IF'? In other words we want the equivalent of:

PRINT MAX (A, B)

without, of course, using the functions MAX or MIN.

Four. For a certain application using a 6800 system the programmer needed to reverse the order of bits in a byte in less than 10 cycles. One attempt is shown in Fig. 1; this routine shifts bits from A to B via the carry bit, and in the process sets B to the reverse of A as required. Unfortunately the routine takes 99 cycles to execute, and at this point the programmer gave up!

```
0000 CE 0008  REVERS LDX     £8
0003 44       LOOP   LSR A
0004 59              ROL B
0005 09              DEX
0006 26 FB           BNE     LOOP
```

Fig. 1. Program for the 6800 to reverse the order of bits in the accumulator; see problem 4.

In fact the problem can be solved, although the approach is somewhat unconventional, and the solution can be extended to more general applications.

Five. There are three things that you might want to do to the carry bit of a micro, namely set it, clear it, or complement it. The Z80 provides instructions to set it (SCF) and complement it (CCF), and clearing it is no problem; you must do SCF, CCF. On the other hand the SC/MP, 6502, and 6800 micros provide the clear carry and set carry instructions, and leave you to work out how to complement the carry. Without affecting the contents of the other registers, what is the shortest way to complement the carry bit on these three micros?

Six. A very pleasing feature of the high-level language Pascal is the 'CASE' statement, illustrated by the example in Fig. 2 (a) which prints one of three values, A, B or C,

```
'CASE' N 'OF'
    1: WRITE(A);
    2: WRITE(B);            (a)
    3: WRITE(C)
'END'
10 IF N = 1 THEN PRINT A
20 IF N = 2 THEN PRINT B    (b)
30 IF N = 3 THEN PRINT C
```

Fig. 2. Two programs which print one of three values depending on the value of N, written in (a) Pascal, or (b) BASIC.

depending on whether N is equal to 1, 2 or 3 respectively. To do the same in BASIC one might use three 'IF' statements, as shown in Fig. 2 (b). Can the same effect be achieved with a single BASIC statement, and if so, how?

Seven. The effect of the SC/MP instructions 'LDI O, CAI O' is to set the accumulator to X'FF if the carry bit is clear, and to X'OO if the carry bit is set. How, without making any assumptions about the contents of any of the registers, can the same be achieved in half the number of bytes?

Eight. The 6800 micro provides two types of instructions to shift the accumulator right; a logical shift right (LSR A) which shifts a zero into the top bit of the accumulator, and an arithmetic shift right (ASR A) which preserves the sign bit, for working with signed twos-complement binary numbers. Unfortunately the 6502 micro only provides us with an LSR A instruction; what is the shortest way of implementing an ASR A using the existing 6502 instructions?

Nine. Finally, a problem for all 6800 owners who wish they had a 6809. One of the great improvements of the 6809 over its predecessor is that its instruction set makes it easy to write relocatable programs. If you did not realise that it is difficult to write relocatable programs on the 6800, try finding a set of instructions with the same effect as:

HERE LDX £HERE

but which will work correctly wherever they are loaded into memory.

LOW-COST SC/MP SYSTEM

The following SC/MP system can be built with a small number of readily available components, and it works without the need for a monitor ROM or EPROM of any kind. It was designed by *Andrew Aitken* who submitted the following details about its operation.

"The full circuit, shown in Fig. 3, includes a single-cycle facility comprising a flip-flop and a few gates. The system has 256 bytes of RAM, at addresses OOOO to OOFF, and the states of the address and data lines are shown on 18 l.e.d.s. The whole circuit needs a 5 volt supply of about $\frac{1}{2}$ amp, and two or three 0·1μF capacitors should be added across the power rails at various points for decoupling.

PROGRAMMING

"Programs and data are entered into the memory as follows: With S1 set to 'PROGRAM' and S4 set to 'SINGLE CYCLE' press 'RESET'. The MPU will then be halted while it is fetching the first word from memory, and NRDS will be low thus enabling the data buffer. Whatever is now set on the data switches will be present on the data bus, and will be read by the MPU. Set the data switches to C4 (the op-code for the Load Immediate instruction) and switch the 'CYCLE' switch S2 up and then down. The instruction is then executed, and the MPU will again set NRDS low, waiting for the data which forms the second byte of the instruction. This is likewise entered at the data switches, and the programs in any sequence, and to change the contents of any location at will. When the program has been entered set S1 to 'RUN', leave S4 on 'SINGLE CYCLE', 'RESET', and cycle through the program by toggling S2. If everything seems fine 'RESET', set S4 to 'CONTINUOUS', toggle S2 once, and the program will run. A particularly pleasing aspect of the system is the ability to stop a program in mid run, by setting S4 back to 'SINGLE CYCLE', change an instruction, and then allow the program to continue so as to see the effect of the change immediately.

"S1 is a double-pole switch to ensure that when the system is in 'RUN' mode the data switches are disconnected from the data bus. Alternatively the data buffer EN line could be corresponding to that key is presented to the inputs of the CMOS inverters by a diode matrix. The outputs of these inverters are connected to the inputs of both of the 4-bit latches. The CMOS inverters were used as buffers because the key switches could only tolerate small currents. If more robust switches are available it would be possible to connect the outputs of the diode matrix directly to the latch inputs; in this case the 12k resistors should be changed to 1k and the data should be taken from the Q outputs of the latches.

"A key-press is detected by a diode gate which charges up a 4·7μF capacitor. This causes the output of the second Schmitt trigger to go high, which clocks the flip-flop

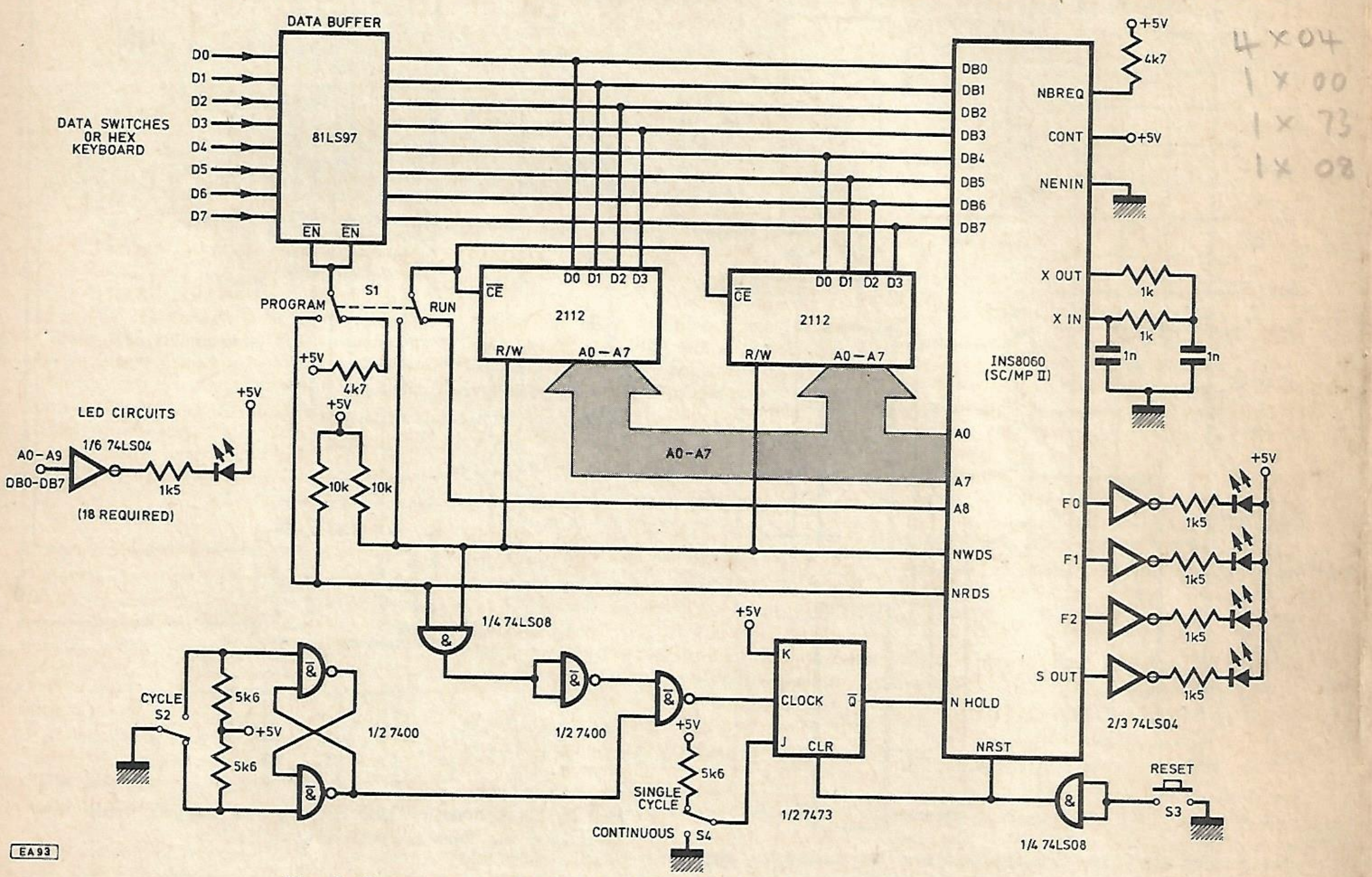


Fig. 3. Complete circuit of the simple SC/MP microprocessor system.

MPU will load this data into the accumulator.

"Now enter C9 (Store relative to pointer register P1) followed by the required memory address. Pointer P1 was set to zero on reset, so on the next cycle the MPU will store the contents of the accumulator, the required data, at this address. When the MPU writes to memory NWDS goes low which will enable the RAM.

For example, to enter 8F at location 0002 the full sequence is:

RESET, C4, CYCLE, 8F, CYCLE, C9, CYCLE, 02, CYCLE.

"The sequence is repeated to enter data at a different address and although the sequence looks quite long, in practice programs can be loaded into RAM fairly easily. The beauty of the system is that it is possible to enter connected to an inverted address line so that the data switches could be read from a program.

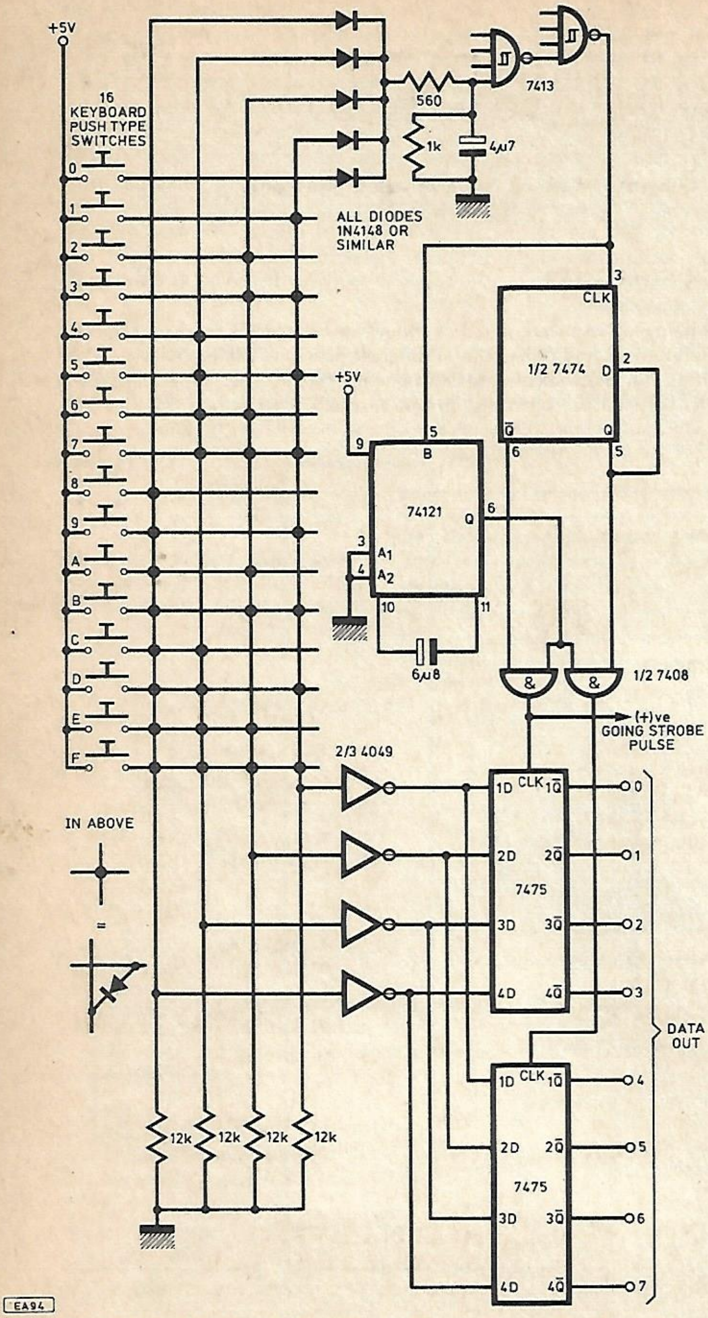
HEX KEYBOARD

"Although data for the SC/MP system can be entered by means of eight toggle switches at the input of the data buffer, a far more convenient method is to use the hex keyboard circuit shown in Fig. 4. The keyboard is based on a circuit in the *September 1978 PE* and would be useful in any application requiring hex data entry.

CIRCUIT OPERATION

"The keyboard circuit buffers two hex keypresses to give an 8-bit value at the output. When a key is pressed the binary code and triggers the monostable. The flip-flop steers the pulse from the monostable to enable the appropriate latch, and this latches the key's value.

"When the key is released the 4·7μF capacitor will discharge through the 1k resistor, and the output of the second Schmitt trigger will return low. The capacitor thus serves to debounce the keys both when they are pressed and when they are released. The next key-press will load data into the other latch, and the pulse from the monostable will be available on the strobe line to signal that a full 8-bit word is ready at the outputs of the latches. When loading a program this strobe line is not required, but it can be tied to SC/MP's Sense-B input so that programs can detect when data has been entered.



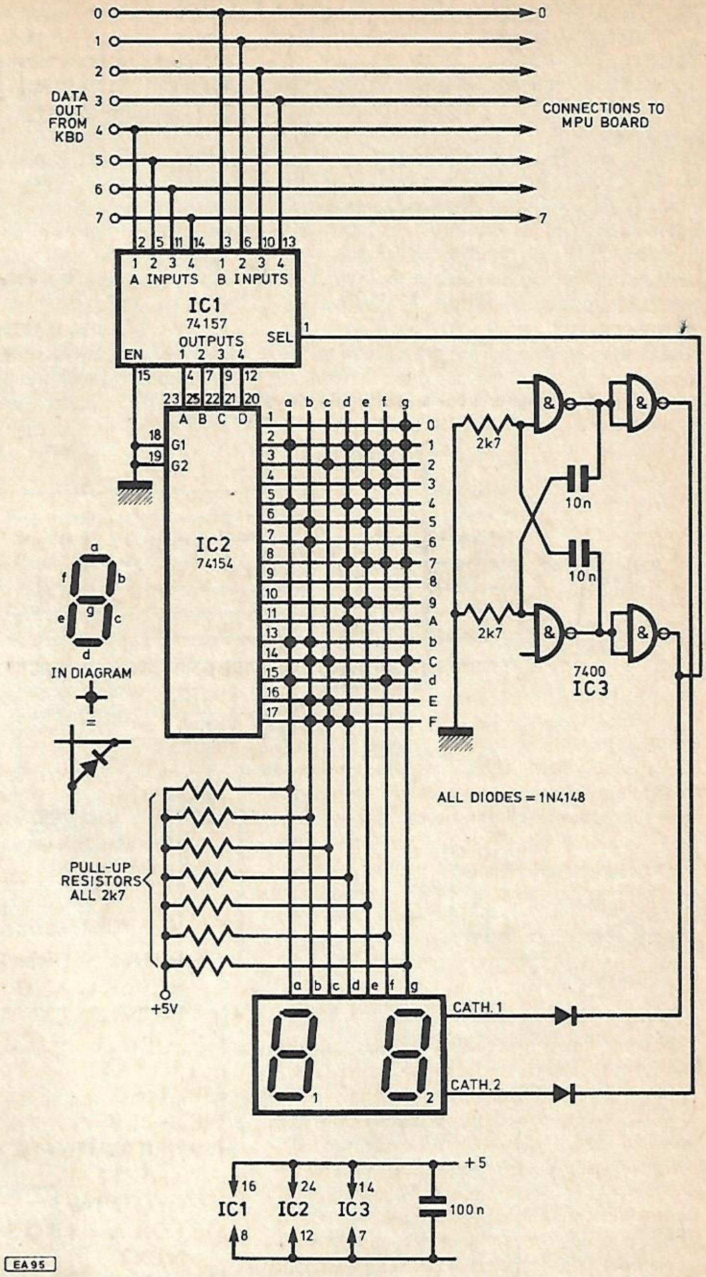


Fig. 5. Circuit which will display the output from the keyboard as two hexadecimal digits.

Fig. 4. Hex keyboard circuit which can be added to the SC/MP system to make data entry easier.

TWO-DIGIT HEX DISPLAY

"A two-digit hex display of the output from the keyboard is another useful addition to the system. The circuit of Fig. 5 achieves this with few parts, and without the need for an expensive decoder chip. The l.e.d. display is a small common-cathode multiplexed type.

"The four NAND gates form an oscillator that drives the cathodes of the displays in turn. One output of the oscillator is also taken to the select input of a 74157 quad two-input data selector which routes the appropriate 4-bit nibble from the data bus to the decoding circuitry. The 74154 decoder pulls one of its 16 outputs low depending on the code at its inputs. Each output is connected to certain segments of the displays by diodes; when the output is pulled low these segments are turned off to produce the required hex character on the display. Turning segments off is simpler than turning segments on, and results in a considerable saving in the number of diodes required. The 2k7 pull-up resistors may be reduced to 1k5 if the display is not considered bright enough.

"The oscillator thus switches the segment codes for each nibble to each display digit in turn, at high speed, giving a two-digit hex display of the data bus."

I/O PORT TESTER

The Acorn 6502-based computer provides two 8-bit I/O ports, and when these are being interfaced to external circuitry it often becomes difficult to keep track of the logic levels on the 16 lines. In such cases the routine of Fig. 6 should prove useful; it gives a continuous display of the states of the ports, in binary, as two rows of 8 dashes on the l.e.d. displays. The top row corresponds to the 8 bits of port A and the bottom row corresponds to the 8 bits of port B. The leftmost dash in each row is bit 7, and the rightmost dash is bit O. A particular dash is illuminated if the appropriate input line is high, and blank if the line is low.

The routine can also be incorporated into programs which control the I/O ports, thus providing a continuous visual indication of what they are doing. In this case modify the last instruction of the routine to an RTS instruction, and insert a call to the routine in the most frequently executed section of your program.

```
                ;       DISPLAY STATE OF I/O LINES
                ;
                DISP   =$0010              DISPLAY BUFFER
                REPEAT =$000E
                ABIT   =$0900              TEST A-SIDE BIT
                BBIT   =$0908              TEST B-SIDE BIT
                DISPLY =$FE0C              DISPLAY ROUTINE
                ;
0000                   .=$0200
0200  A9 1F     TEST   LDA    £$1F         SINGLE SWEEP OF
0202  85 0E            STA    REPEAT         DISPLAY.
0204  A2 07            LDX    £7
0206  BD 08 09  LOOP   LDA    BBIT,X       BIT 7 = STATE
0209  0A               ASL    A            INTO CARRY
020A  7D 00 09         ADC    ABIT,X
020D  95 10            STA    DISP,X       PUT IN BUFFER
020F  CA               DEX
0210  D0 F4            BNE    LOOP
0212  20 0C FE         JSR    DISPLAY      SWEEP DISPLAY
0215  10 E9            BPL    TEST         I.E. ALWAYS

                       .END
```

Fig. 6. Program for a 6502 displays the states of an Acorn's I/O lines.

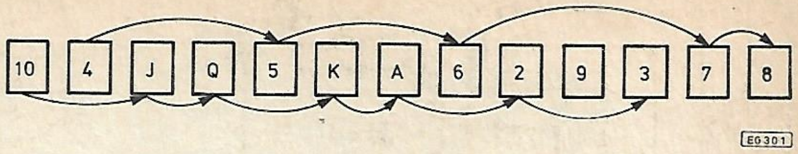


Fig. 7. Diagram to solve the card-trick problem.

CARD TRICK SOLUTION

In the last Micro-Bus you were asked to deduce which card in a series of thirteen cards had been removed, and replaced at a different position. The problem could be solved by entering the sequence of cards into one of the card-trick programs. Alternatively consider the sequence separated into two ascending series as indicated by the lines in Fig. 7. The nine is then clearly anomalous, and so this was the chosen card.

MICRO PROMPT

The hardware and software exchange point for PE computer projects

Yes, we know! This is supposed to be a bimonthly column; it appeared last month, yet here it is again! Well, the Prompt file is full of goodies, some of which we know are anxiously awaited, so we slipped this one in whilst no one was looking.

SAVE IT

Having stated in our first Prompt that the 101 has no cassette file handling firmware, we are now knee-deep in letters explaining various ways of saving raw data on tape. Below is a program which should provide the seed for some rewarding experimentation in cassette file keeping. It is an optimised combination of all the ideas sent in, some crude, some not so crude, plus our own refinements, and allows a named data file to be recorded. The data can be numbers or strings of text, since the technique utilises the SAVE and LOAD commands under program control. This program will take five words from you and record them as "FILE A"

save on tape

```
5 FOR A = 1 to 5 :INPUT
  "WORD ": W$(A) : NEXT
10 PRINT "TURN TAPE TO
  RECORD, & WAIT" : FOR A = 1
  TO 8000 : NEXT
20 PRINT : PRINT "HIT ANY KEY"
25 POKE 11, O : POKE 12, 253 : X =
  USR (X)
30 SAVE : PRINT "FILE A"
40 FOR A = 1 TO 5 : PRINT W$(A) :
  NEXT
50 POKE 517, O
60 END
```

Run the SAVE program, rewind the data tape, and then run the LOAD Program (RUN 120). All data will thus be cleared from memory, and recovery of the words will rely entirely on the tape file.

load from tape

```
120 PRINT "TURN TAPE TO
   REPLAY, AND HIT ANY" :
   PRINT "KEY IMMEDIATELY"
125 POKE 11, O :POKE 12, 253 : X =
   USR (X)
130 LOAD
140 INPUT T$
150 IF RIGHT$ (T$, 6) = "FILE A"
   THEN 170
160 GOTO 140
170 FOR A = 1 TO 5 : INPUT W$(A) :
   NEXT
180 POKE 515,O : PRINT : PRINT :
   PRINT
190 PRINT RIGHT$ (T$, 6) : PRINT :
   FOR A = 1 TO 5 : PRINT W$(A) :
   NEXT
```

Advantage is taken of the fact that any PRINT statement after a SAVE command will write to the cassette interface, and any INPUT statement after a LOAD command will take data from the cassette.

To revert to normal operation in each case, it is necessary to POKE the relevant SAVE/LOAD flag off again with a zero (lines 50 and 180).

A delay is included (dead FOR-NEXT loop) to wait for the tape leader to clear and the recorder to settle down etc.,

Lines 150 and 190 use RIGHT$ to look at only these six characters: "FILE A", which may find themselves tacked on the end of some noise characters—*all* of which will think they are T$.

101 LOCATIONS

Here are some useful UK101 scratchpad memory locations which have been discovered by *P. Goodwin* of Southampton.

hex	dec		
0200	512	Cursor position along line	
0206	518	VDU operating speed	
0213	531	Character returned by keyboard input routine	
0130	304	NMI Vector	these are in the middle of the stack
01C0	448	IRQ Vector	
0203	515	LOAD Flag POKE 515, 0 turns Load off	
0205	517	SAVE flag POKE 517, 0 turns Save off	
0218	536	Input Vector	
021A	538	Output Vector	
000F	15	Terminal Width	
0300		Program End pointer (POKE this at your peril)	

HEAT POLLUTION

We received a letter from *Mr. J. Briggs* of Malton, N. Yorks, describing a problem with his 101 concerning video stability. The machine worked fine with the family television, but when used with a portable (PYE Model 191) the picture broke up as if incorrectly tuned, after about 10 minutes. Heat from the 5V regulator seemed to be affecting the modulator capsule, and anyone experiencing the same difficulty should note that the problem was cured by mounting the regulator and heatsink separately from the p.c.b.

The 3300µF reservoir capacitor has on some 101 boards suffered from excessive heat too. We have been told this can produce video *and* keyboard problems.

Next month's Prompt will include a table of handy and unexpected characters available direct from the 101 keyboard, and a review of some software which enables line editing and programmable cursor movement in *all* directions. We shall also publish that promised CHAMP program.

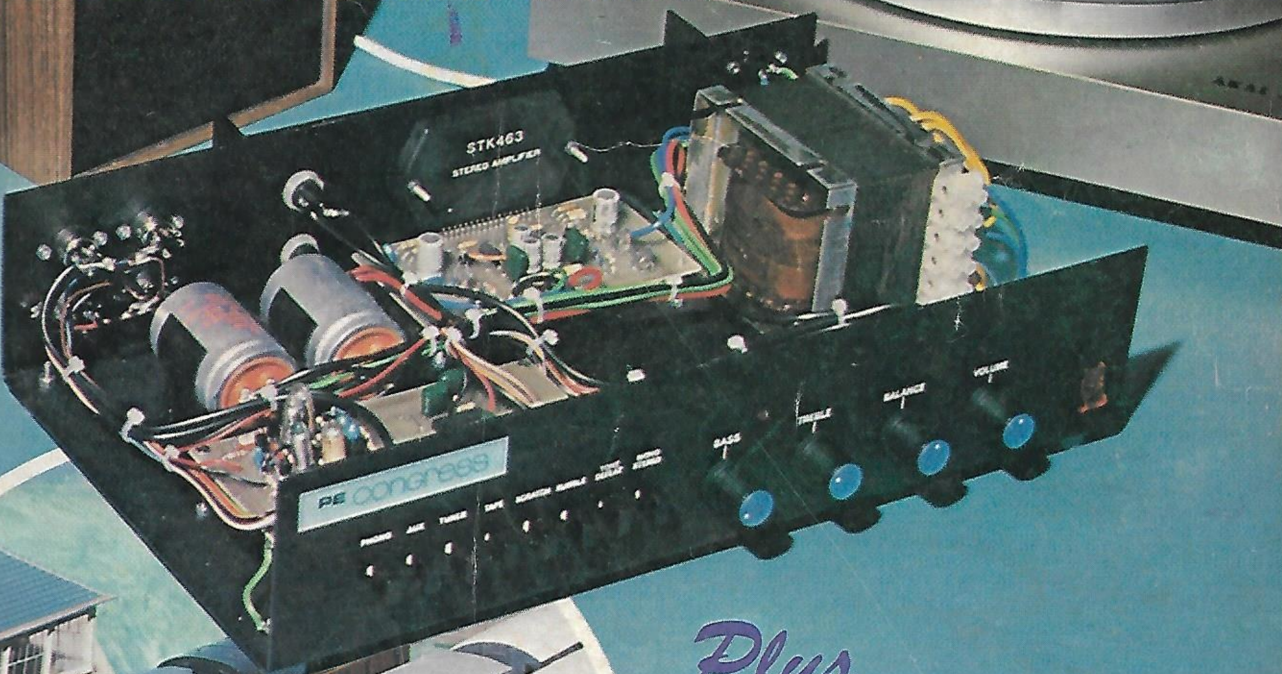
Australia 95c New Zealand $1.10 South Africa 90c Malaysia $3.50

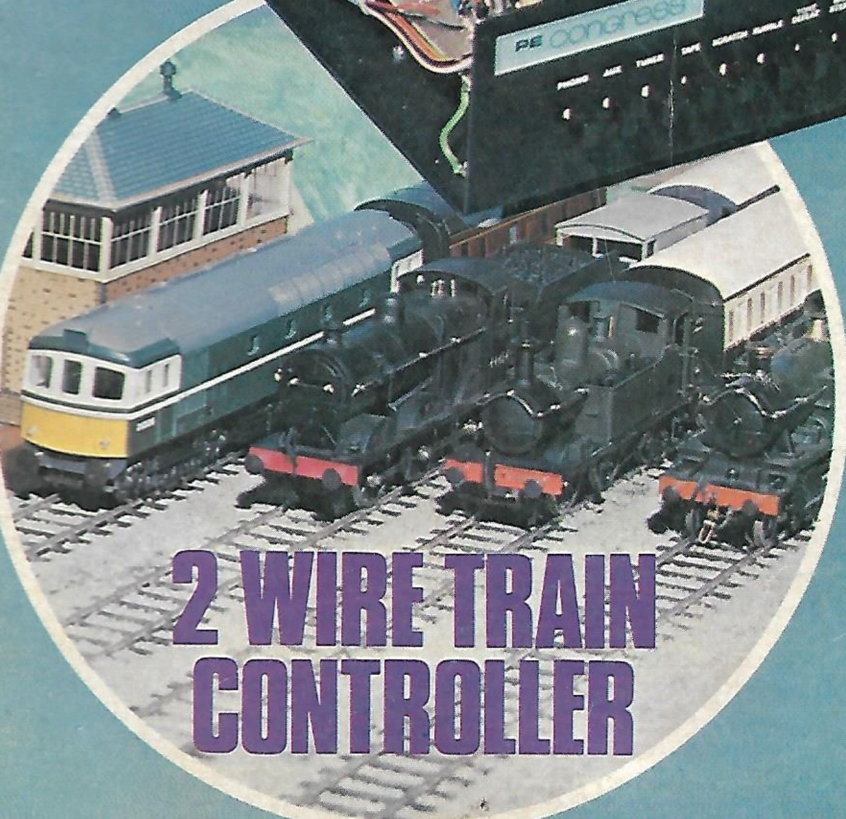
PRACTICAL ELECTRONICS

APRIL 1980

55p

30W I.C. STEREO AMPLIFIER





2 WIRE TRAIN CONTROLLER

Plus

VIDEOTONE GB3 SPEAKER OFFER

Also...

SPECIAL SUPPLEMENT VIDEO for EVERYONE!

PROGRAMMING

"Programs and data are entered into the memory as follows: With S1 set to 'PROGRAM' and S4 set to 'SINGLE CYCLE' press 'RESET'. The MPU will then be halted while it is fetching the first word from memory, and NRDS will be low thus enabling the data buffer. Whatever is now set on the data switches will be present on the data bus, and will be read by the MPU. Set the data switches to C4 (the op-code for the Load Immediate instruction) and switch the 'CYCLE' switch S2 up and then down. The instruction is then executed, and the MPU will again set NRDS low, waiting for the data which forms the second byte of the instruction. This is likewise entered at the data switches, and the programs in any sequence, and to change the contents of any location at will. When the program has been entered set S1 to 'RUN', leave S4 on 'SINGLE CYCLE', 'RESET', and cycle through the program by toggling S2. If everything seems fine 'RESET', set S4 to 'CONTINUOUS', toggle S2 once, and the program will run. A particularly pleasing aspect of the system is the ability to stop a program in mid run, by setting S4 back to 'SINGLE CYCLE', change an instruction, and then allow the program to continue so as to see the effect of the change immediately.

"S1 is a double-pole switch to ensure that when the system is in 'RUN' mode the data switches are disconnected from the data bus. Alternatively the data buffer EN line could be corresponding to that key is presented to the inputs of the CMOS inverters by a diode matrix. The outputs of these inverters are connected to the inputs of both of the 4-bit latches. The CMOS inverters were used as buffers because the key switches could only tolerate small currents. If more robust switches are available it would be possible to connect the outputs of the diode matrix directly to the latch inputs; in this case the 12k resistors should be changed to 1k and the data should be taken from the Q outputs of the latches.

"A key-press is detected by a diode gate which charges up a 4·7µF capacitor. This causes the output of the second Schmitt trigger to go high, which clocks the flip-flop

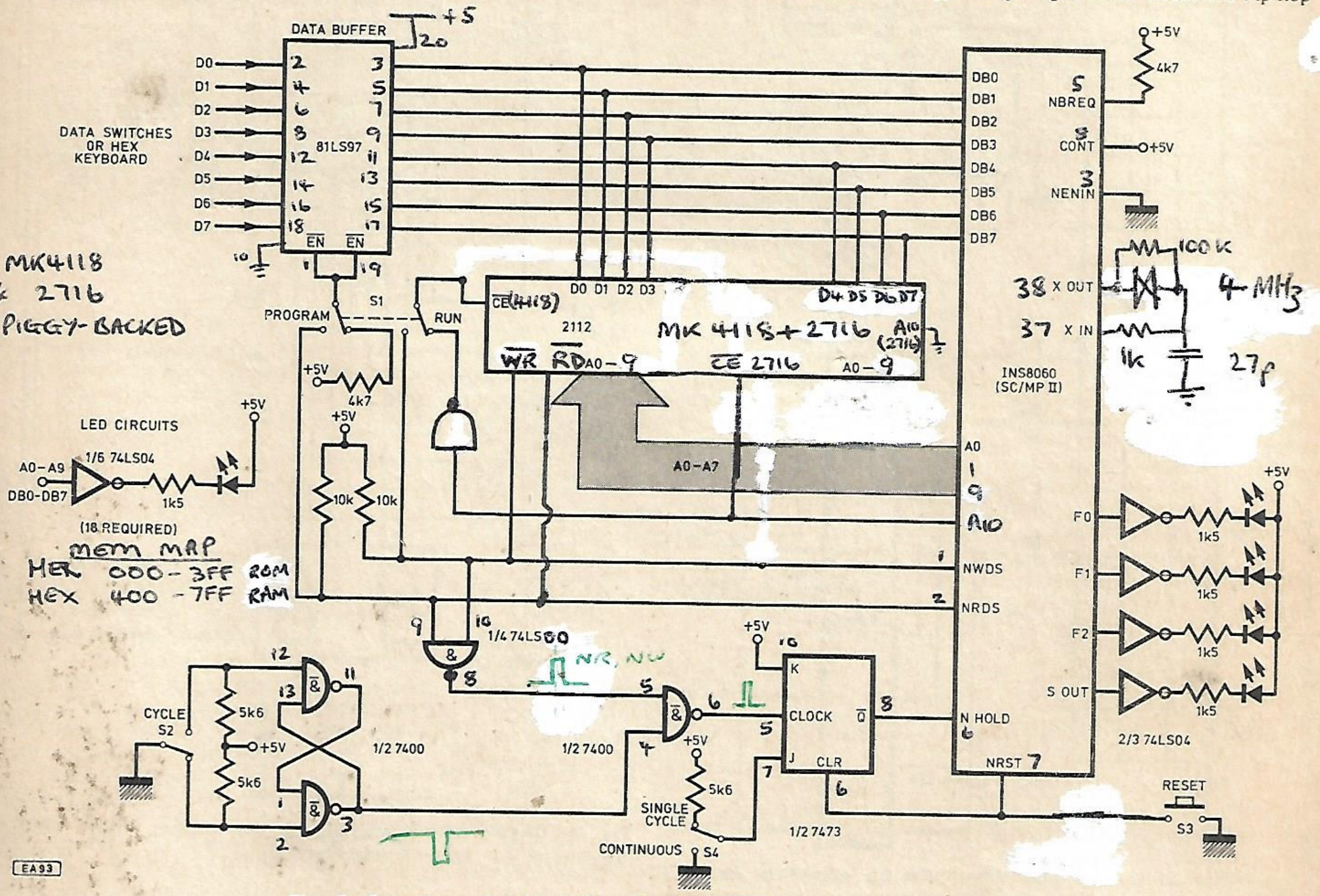


Fig. 3. Complete circuit of the simple SC/MP microprocessor system.

MPU will load this data into the accumulator.

"Now enter C9 (Store relative to pointer register P1) followed by the required memory address. Pointer P1 was set to zero on reset, so on the next cycle the MPU will store the contents of the accumulator, the required data, at this address. When the MPU writes to memory NWDS goes low which will enable the RAM.

For example, to enter 8F at location 0002 the full sequence is:

RESET, C4, CYCLE, 8F, CYCLE, C9, CYCLE, 02, CYCLE.

"The sequence is repeated to enter data at a different address and although the sequence looks quite long, in practice programs can be loaded into RAM fairly easily. The beauty of the system is that it is possible to enter connected to an inverted address line so that the data switches could be read from a program.

HEX KEYBOARD

"Although data for the SC/MP system can be entered by means of eight toggle switches at the input of the data buffer, a far more convenient method is to use the hex keyboard circuit shown in Fig. 4. The keyboard is based on a circuit in the *September 1978 PE* and would be useful in any application requiring hex data entry.

CIRCUIT OPERATION

"The keyboard circuit buffers two hex key-presses to give an 8-bit value at the output. When a key is pressed the binary code and triggers the monostable. The flip-flop steers the pulse from the monostable to enable the appropriate latch, and this latches the key's value.

"When the key is released the 4·7µF capacitor will discharge through the 1k resistor, and the output of the second Schmitt trigger will return low. The capacitor thus serves to debounce the keys both when they are pressed and when they are released. The next key-press will load data into the other latch, and the pulse from the monostable will be available on the strobe line to signal that a full 8-bit word is ready at the outputs of the latches. When loading a program this strobe line is not required, but it can be tied to SC/MP's Sense-B input so that programs can detect when data has been entered.