An ideal microcomputer for the beginner:

"Mini Scamp"

Forget about expensive terminals: here's a REALLY low cost and simple microcomputer. It uses front-panel bit switches and LEDs for input and output, in normal binary code, making it completely self contained. Based on the National SC/MP microprocessor, it comes with a minimum of 256 words of RAM—but this is easily expanded up to 1024 words. We think it's the ideal way of getting into the exciting world of microcomputers at low cost.

by DR. JOHN KENNEWELL Physics Dept., Newcastle University

The design of this microcomputer started around October of last year with the formation of the Newcastle Microcomputer Club. It became obvious at the inaugural meeting that there were many people who would like to play with their own microcomputer, developing programming skills, yet who were unable to afford even the lowest cost kits available on the market. The problem becomes even more acute when considering the cost of a terminal to interface with these kits.

Various solutions to the terminal problem have now been presented in this magazine. Jim Rowe has described an ASCII-Baudot translator for use with surplus Baudot teleprinter machines (EA, October 1976) and also a video data terminal (EA, January and February 1977). However, either of these alternatives means an outlay of at least about $200, not including the microcomputer itself, which brings the total cost to around $300 using a small system such as the SC/MP evaluation kit.

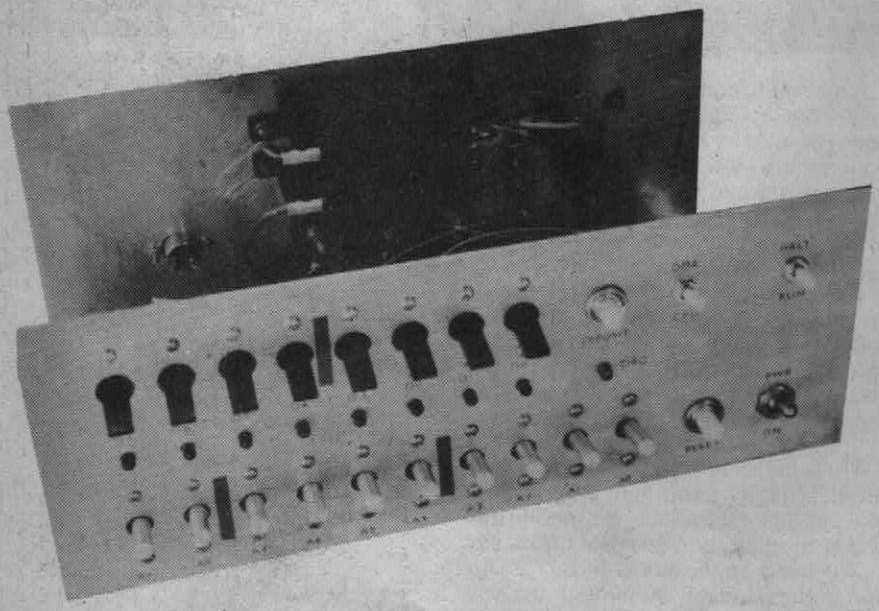
Recently Applied Technology have released a SC/MP I/O kit, which interfaces with the SC/MP evaluation kit and permits program and data entry via panel switches. While this unit undoubtedly fills a gap in available I/O hardware and seems to be enjoying great popularity, the total system cost is over $150 (including power supply). I also feel that it is not suitable for a beginner to microcomputers because it employs an operating system (in ROM) that was designed for communication with a teletype using hexadecimal characters in ASCII code. Entry of information via the panel switches thus involves a prior translation of characters into this code, and one tends to lose contact with the basic organisation of the processor (CPU). From the point of acquiring an understanding of microprocessor operation (without the complications of an intervening operating system) I feel this is undesirable.

In searching for a suitable design, and to overcome the problems mentioned above, it became apparent to me that the idea of using an available evaluation kit together with an I/O interface was not the way to go. As an operating system was not desirable, there was no need for read only memory (ROM). Building a system from scratch meant that costs could be kept down as only those features necessary were included. At the same time, I personally wished to build a much larger system than that shown here, and ease of system expansion was well to the fore in my design considerations. The TOTAL cost of the computer should fall somewhere between $50-$100 depending upon your method of construction, selection of components, and upon how many existing components you have that may be pressed into service. This is most likely to be so in the case of the power supply.

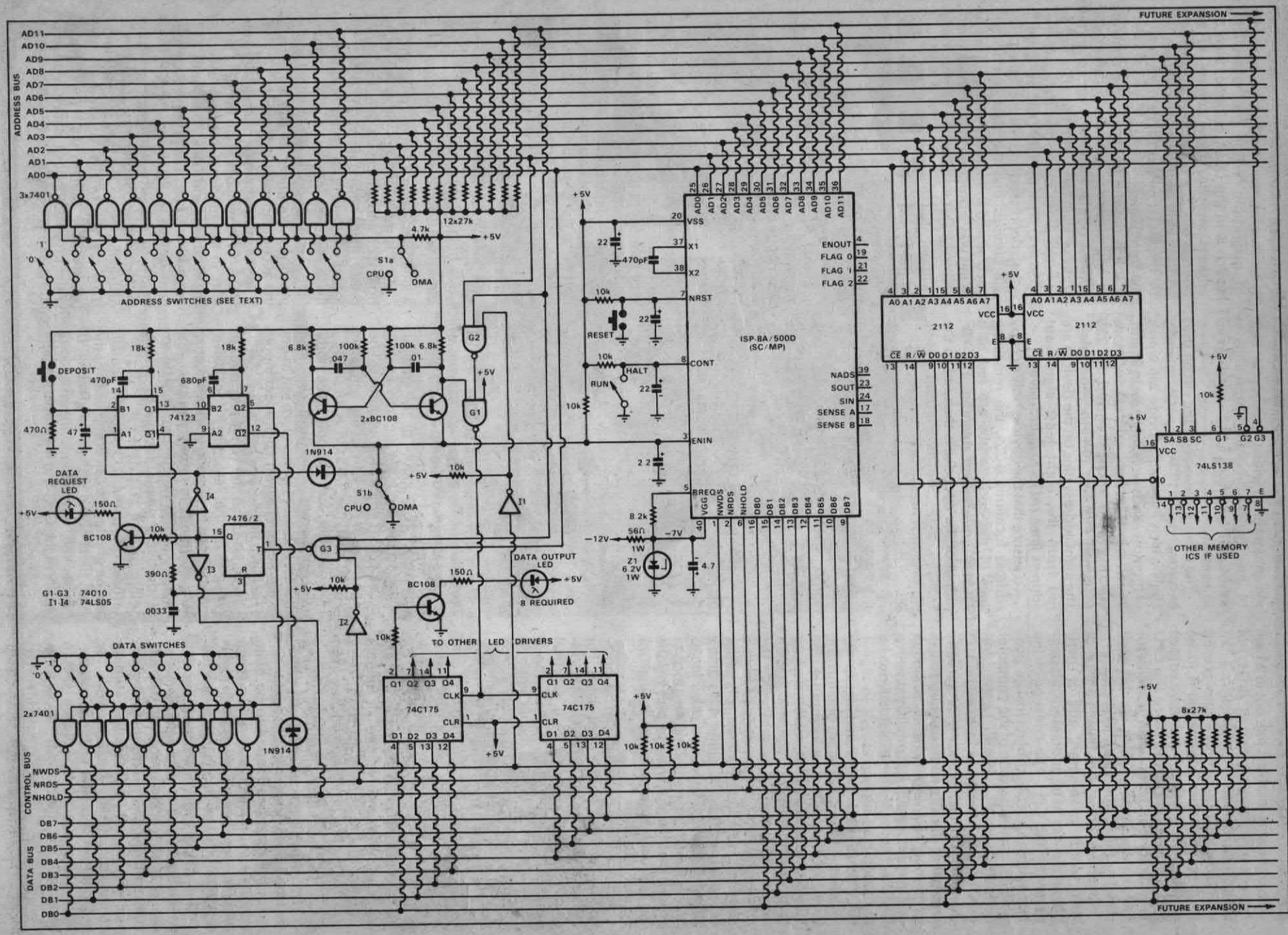
Excluding the power supply, the computer may be conveniently divided into three basic units. These are the central processing unit (CPU), the memory, and the front panel input/output circuitry. These three units communicate with one another via three system 'bus' lines: an address bus, a data bus, and a control bus.

The address bus, comprising twelve actual lines, is used by both the CPU and the front panel I/O circuit to specify which location in memory information is to be sent or received. It is also used by the CPU during program execution, to select a particular input or output device for communication with the outside world.

The data bus, of 8 lines, enables the passage of information between any two of the three units in either direction. The unit that does not participate in a given



At left is the author's prototype of his Mini Scamp, with the full circuit shown on the page opposite.


FUTURE EXPANSION
ADDRESS BUS
AD11
AD10
AD9
AD8
AD7
AD6
AD5
AD4
AD3
AD2
AD1
AD0
3x7401
12x27k
4.7k
+5V
S1a
CPU
DMA
ADDRESS SWITCHES (SEE TEXT)
ISP-8A/500D
(SC/MP)
VSS
X1
X2
NRST
CONT
ENIN
SENSE A
SENSE B
ENOUT
FLAG 0
FLAG 1
FLAG 2
SIN
SOUT
NADS
RESET
HALT
RUN
470pF
2112
74LS138
OTHER MEMORY
ICS IF USED
DEPOSIT
74123
2xBC108
1N914
DATA
REQUEST
LED
BC108
150Ω
7476/2
G1-G3 : 74010
I1-I4 : 74LS05
DATA OUTPUT
LED
8 REQUIRED
TO OTHER LED DRIVERS
DATA SWITCHES
2x7401
74C175
Z1
6.2V
1W
8x27k
CONTROL BUS
NWDS
NRDS
NHOLD
DATA BUS
DB7
DB6
DB5
DB4
DB3
DB2
DB1
DB0
FUTURE EXPANSION

```
                *BINARY COUNT AND DISPLAY
0000 08           NOP
0001 C408         LDI   8
0003 35           XPAH  1
0004 C400         LDI   0
0006 31           XPAL  1
0007 C902  LOOP   ST    2(1)
0009 8FFF         DLY   255
000B A803         ILD   COUNT
000D 90F8         JMP   LOOP
000F   00  COUNT  .BYTE 0
0010
```

```
                *MOVING LIGHTS WITH INPUT
0000 08           NOP
0001 C408         LDI   8
0003 35           XPAH  1
0004 C400         LDI   0
0006 31           XPAL  1
0007 C101  LOAD   LD    1(1)
0009 C810         ST    BITS
000B C00E  LOOP   LD    BITS
000D C902         ST    2(1)
000F 1E           RR
0010 C809         ST    BITS
0012 8FFF         DLY   255
0014 B806         DLD   COUNT
0016 9CF3         JNZ   LOOP
0018 90ED         JMP   LOAD
001A   00  BITS   .BYTE 0
001B   00  COUNT  .BYTE 0
```

Here are two sample programs to help you get going with Mini Scamp. In both cases the hexadecimal numbers in the first column are memory addresses, and those in the next column are the actual code. Each pair of hex digits is an 8-bit byte.

greatly increase the cost of the overall unit, and may not be justified, particularly if further expansion is not desired.

The LED's and their transistor drivers were soldered onto a long narrow strip of Veroboard and then the LED's were glued through holes in the front panel.

The power supply requirements are quite small, and any supply giving +5V at say 0.5A and about -12V at 150mA should prove adequate. Fig. 2 gives a typical circuit for those wishing to build the power supply using new components.

It will be found that the cost of the switches can be a considerable fraction of the total computer cost. Those used for the prototype were lever switches (DPDT) from Tandy Electronics. Similar switches at a much lower cost from Electronic Disposals in Little Lonsdale Street in Melbourne have also been tried. Although satisfactory to date, only time will allow us to determine how many repeated switchings may be made before the contact resistance becomes too large for correct operation. In this regard, it is most desirable to wire both poles of the above type of switches in parallel.

Although this microcomputer was conceived mainly as an educational instrument through which an understanding of the engineering and programming concepts involved could be learnt, there is no reason why it could not be put to work in the role of a simple controller. Detection of off/on states of various devices, and the activation of relays, etc., is most easily accomplished using the sense inputs and flag outputs available on the SC/MP chip. Anyone wishing to make good use of the computer should obtain a copy of the SC/MP Technical Description from NS Electronics Pty. Ltd., in Bayswater, Victoria, or from their distributors. This manual describes all of the program instructions available on the SC/MP, and details what each of these actually does.

On the programming side, you might like to try your hand at writing a multiplication routine, a BCD to binary conversion program, the converse, i.e. binary to BCD, or even a simple program to demonstrate the function of the logical operations, AND, OR and EXCLUSIVE OR.

Although it uses a different instruction set than does the SC/MP, the advice on programming contained in the EDUC-8 handbook provides valuable information for those with little prior knowledge in this field. I have also found the hexadecimal conversion table printed in the E.A. Yearbook (1976/77) to be of great assistance when manually assembling small programs.

In order to get you started along the road in programming your microcomputer, I will describe two short demonstration programs.

The first program simply counts in binary, displaying each number on the LED's with a fixed delay between numbers. The delay is necessary to slow the computer down sufficiently for you to observe what is happening. The program listing is shown in Fig. 3.

Ignoring the first two columns for the moment, what we have is a list of instructions to the computer in 'Assembly' language. The first instruction (NOP) does nothing, and is ignored by the CPU, as the first instruction actually executed is at address one. The next four instructions load the hex address 0800 into pointer register 1. The following instruction (ST 2(1)) outputs whatever number is presently in the accumulator to the LED's. Note that the operand 2(1) means the address stored in pointer register 1 plus 2 (i.e., 0802) which is, of course, the address of the LED's.

The next instruction (DLY 255) creates the delay, while the ILD COUNT instruction adds one to the location called COUNT (which has address 000F) and then loads this number into the accumulator, ready for display on the LED's when the CPU jumps back (via the JMP LOOP instruction) to the ST 2(1) instruction.

The information in the second column is the translation of the assembly language instructions detailed above into machine language form. These hexadecimal numbers may be loaded into the memory at their respective addresses shown alongside in column 1.

First, set the RUN/HALT switch to HALT, and the DMA/CPU switch to DMA. Then set all the address switches to zero, and set the hexadecimal number 08 on the data switches. Now press DEPOSIT, and the LED's should also display 08. Continue by setting the address switches to 01 and the data switches to C4. Depress DEPOSIT again. As the LDI 8 instruction is a double-byte instruction, the number 08 must now be set into address 02 followed by 35 into 03, and so on, using the above procedure. When all locations up to and including 0F have been loaded, the program is ready to be executed or run.

Set the DMA/CPU switch to CPU, depress and release the RESET button, and then set the RUN/HALT switch to RUN, and the LED's should then be counting. If not, return the appropriate switches to HALT and DMA, in that order, and check the contents of each address by successively incrementing the address switches from 00 to 0F hex.

The second program shown in Fig. 4 demonstrates both the data input facility of the computer and also the rotate instruction (RR). When run, the program will request (via DRQ) any number from the data switches. For example, when DRQ comes on, set hex 80 on the switches, then press DEPOSIT. The single light on the left will then be successively moved to the right and finally 'rotated' back to its initial position. After 256 rotations, the program will then request a new bit pattern to rotate.

Look what happened to the Mini Scamp!

With some help from component suppliers, we have been able to turn Dr. Kennewell's Mini Scamp microcomputer design into a complete full-scale construction project. With almost all of the circuitry on a single PC board, it is now not just the lowest-cost complete microcomputer system available, but the easiest to build as well!

by JAMIESON ROWE

Just about everyone interested in microcomputers seems to agree that Dr. John Kennewell's Mini Scamp design has great potential. By starting from scratch with a SC/MP chip, and then designing a simple RAM-orientated system around it, he has produced an ideal microcomputer for the hobbyist and student.

It is fully self-contained, needing no expensive terminal. Programs are fed in via front-panel switches and LEDs, which can also be used to communicate with the machine when it is running—in simple binary code, the actual language used by the machine itself. What better way to learn how computers work!

At the same time, it can be built for around half the cost of any other microprocessor based system, and hundreds of dollars less than broadly comparable earlier designs like our own EDUC-8.

In other words, it is a design which should appeal to a wide variety of people, especially those still looking for a way of becoming familiar with microcomputers easily and at low cost.

While we were preparing Dr. Kennewell's article for last month's issue, the conviction grew that the project deserved to become a very popular one. But we realised that one thing was lacking: a low-cost PC board, to make it really easy to build even for those with little previous experience.

We immediately resolved to design a PCB for the project, to help ensure that it wins the popularity it deserves. And we managed to fit a small "stop press" box in the April article, to let readers know that a PCB was on the way.

Because of the box no doubt quite a few readers have been waiting for the current issue, for the promised PCB design. As you can see, we have in fact gone much further than this, and have turned Mini Scamp into a full-scale project. So that your wait should not have been in vain.

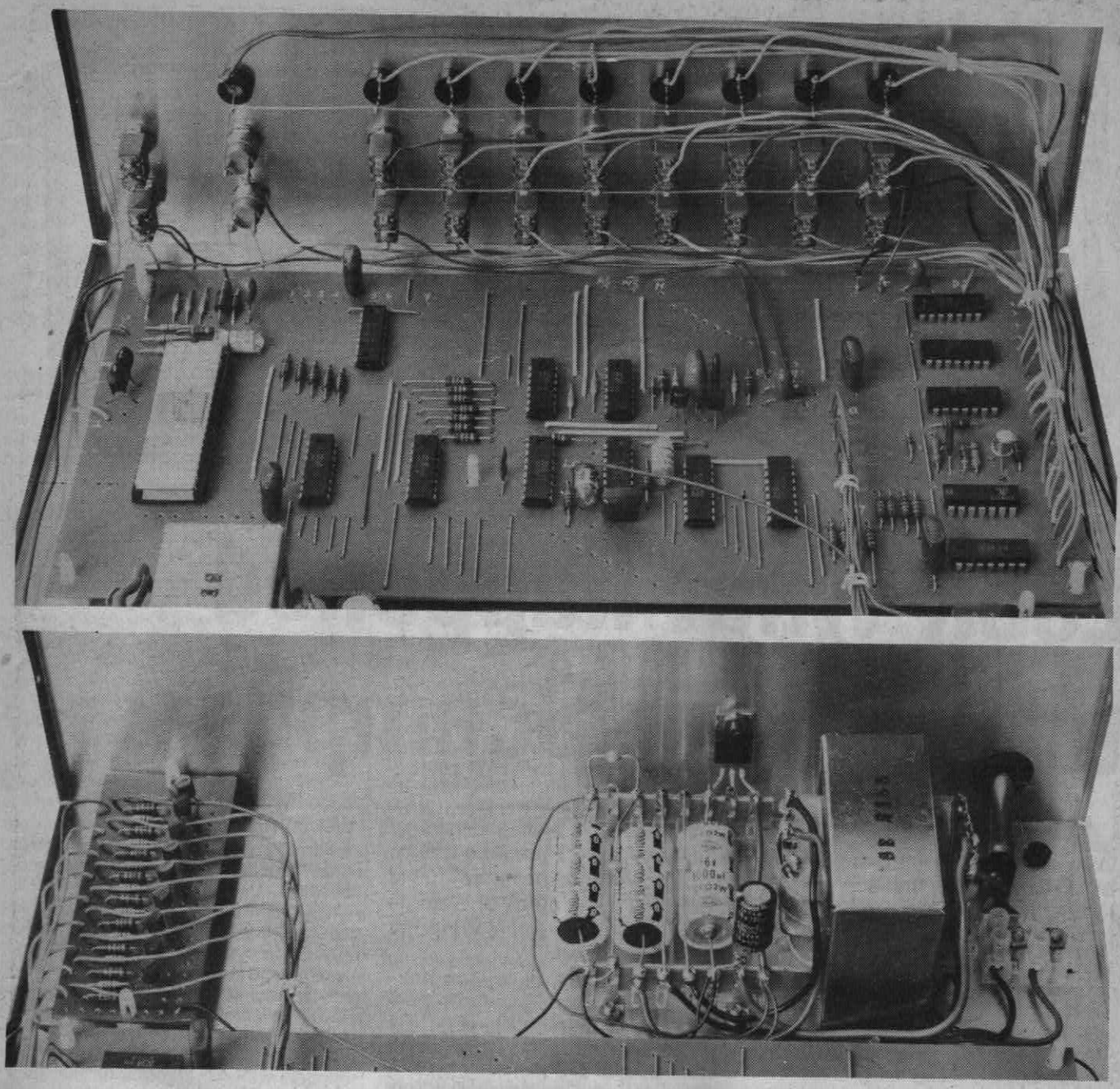
How did this happen? Well, we couldn't publish the PCB design without trying it out first, to make sure it worked. This meant getting together a set of ICs, switches and other components, and at the same time warning suppliers that the project was coming—to ensure that readers would be able to buy the parts. While we were doing this we sought reactions from suppliers also, to see if they became as keen about the project as we were.

They certainly did. In fact, some of the suppliers were so enthusiastic that they offered to make some of the components available to readers at special prices.

For example NS Electronics have offered to make available through their distributors a special deal on the SC/MP chip, all the other ICs, the transistors and LEDs. These will be available for $36.21 including tax, or $31.49 to schools and colleges who can claim a tax exemption, until the end of June.

Similarly C&K Electronics are prepared to supply the complete set of 18 toggle switches and 2 pushbuttons direct to readers for a package-deal price of $14.65 plus 50c postage, or $12.74 plus 50c postage for those who can claim a tax exemption. Their address is PO Box 101, Merrylands, NSW 2160.

* When we told the story to well known kits-n-bits entrepreneur Dick Smith, his immediate question was why we weren't



As these pictures show, Mini Scamp now really looks the part, comparing well with machines costing much more. You should be able to build it for around $105 including tax, and thanks to the new PC board it should take you only a few evenings' work!

planning to describe such an excellent design as a full-scale construction project. This would then allow his firm and, others to produce a complete kit...

Needless to say, we decided there and then to do just that. And thanks to quite a bit of help from Dick Smith, NS Electronics, C&K, RCS Radio, Radio Despatch Service and Bespoke Metalwork, we have been able to produce the full project in double-quick time.

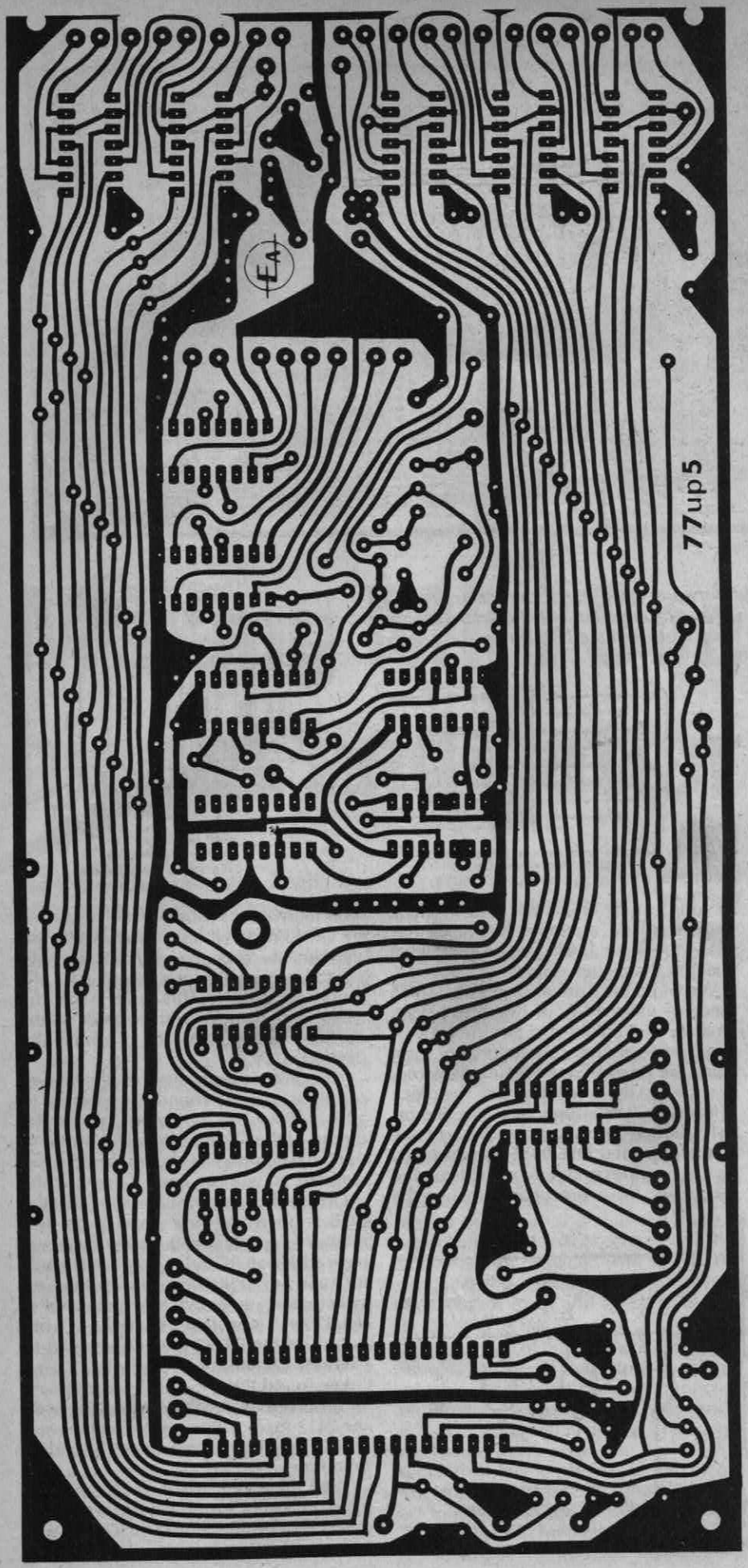
As you can see, it really looks the part, comparing very favourably with designs costing three and four times the price. Yet with most of the circuit on a single PCB measuring 254 x 117mm, you should be able to wire it up very easily in a few evenings.

The basic design has 256 bytes of RAM memory, plenty to let you cut your teeth on basic programming. However we have provided the PCB with data bus, address bus and control signal access, so that additional "outboard" memory may be added very easily. In fact all you will need to expand the memory to 1k bytes is six more 2112 memory chips, a small PCB or piece of perf-board to mount them on, and some hookup wire!

We have even allowed for a couple of additional address switches to be mounted on the front panel, if you want to expand the memory to 1k in this way.

We have also brought out all of the flag and sense pins on the SC/MP chip itself, so that it should be possible to interface Mini Scamp to a terminal later on if you wish. If there is sufficient reader interest we may be able to tell you how to add the "Kitbug" ROM into the system, with its terminal interfacing and debug routines. This should again be a relatively simple matter.

Wiring up the PCB should be quite straightforward as we have prepared an overlay diagram showing the position and orientation of all parts. There is a reasonable number of links, as the PCB is single-sided to keep the cost low, but not so many as one might have expected.



PARTS LIST FOR OUR MINI SCAMP

1 Case, 285 x 235 x 104mm
1 Printed circuit board, 254 x 117mm, coded 77up5
18 SPDT miniature toggle switches, C & K type 7101 or similar
2 Miniature pushbuttons, single pole normally open type, C & K type 8532 or similar
1 stepdown transformer, with multi-tapped 15V secondary at 1A. Type 2155 or similar

SEMICONDUCTORS

1 ISP-8A/500D microprocessor (SC/MP)
2 2112 memory chips (256 x 4)
2 74C175 quad latches
1 74C10 triple 3-input gate
1 74LS138 decoder
1 74LS05 hex inverter
5 7401 hex inverters
1 7476 dual flipflop
1 74123 dual monostable
12 BC108, BC317 or similar transistors
9 5mm diameter LEDs with panel adapters (8 red, 1 green or yellow)
1 LM340T-5, 7805 or similar 5V/1A 3-terminal regulator
4 1N914, 1N4148 or similar diodes
4 50V/1A rectifier diodes
1 6.8V/1W zener diode

RESISTORS

1W rating: 1 x 56 ohm
¼W rating: 9 x 150 ohm, 1 x 390 ohm, 1 x 470 ohm, 1 x 2.7k, 2 x 4.7k, 2 x 6.8k, 1 x 8.2k, 19 x 10k, 2 x 18k, 20 x 27k, 2 x 100k

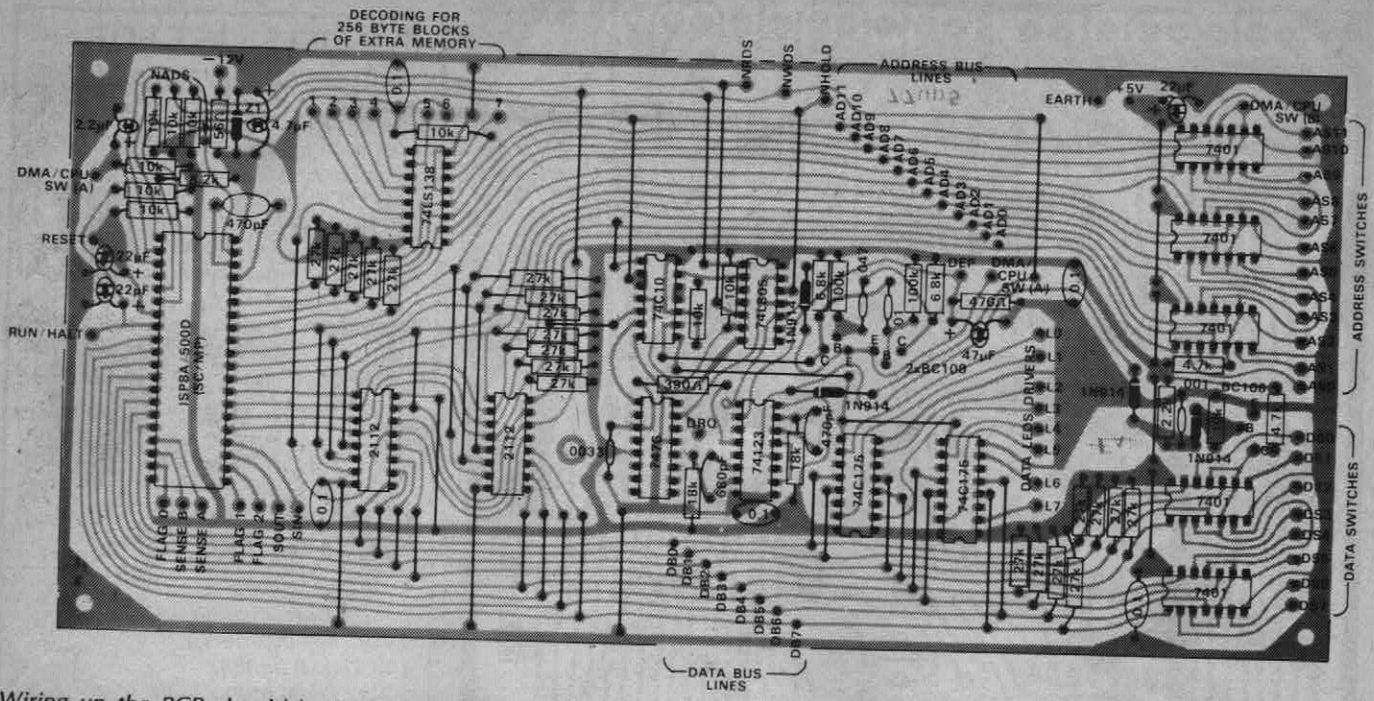
CAPACITORS

LV greencap polycarbonate: 1 x 1000pF, 1 x 3300pF, 1 x .01uF, 1 x .047uF, 5 x 0.1uF
2 470pF polystyrene or NPO ceramic
1 680pF polystyrene or ceramic
1 2.2uF 35VW tantalum
1 4.7uF 35VW tantalum
3 22uF 6VW tantalum
1 47uF 6VW tantalum
1 100uF 16VW electrolytic
3 1000uF 16VW electrolytic

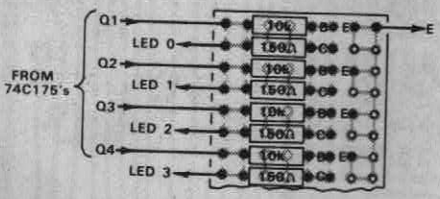
MISCELLANEOUS

Three-wire mains cord and 3-pin plug; grommet and cord clamp; 40-pin DIP socket for SC/MP (PC type); 7 x nylon PCB supports (Richco); 1 x 85 x 40mm piece of utility PCB for LED drivers; 2 x 8-lug miniature tagstrips for power supply wiring; 4 rubber feet for case; connecting wire, solder, nuts, bolts, etc.

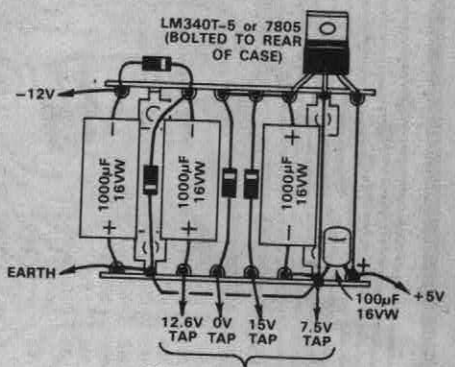
At left is the PCB pattern reproduced actual size. Etched and drilled boards should be available from the usual suppliers by the time you read this.



Wiring up the PCB should be fairly straightforward using the above diagram as a guide. Note that the address and data bus lines are brought out only for future expansion; this applies also to the flag and sense lines. Details of the LED driver and power supply wiring are shown below and at right.



HOW TO WIRE THE LED DRIVER PANEL (9 DRIVERS IN ALL)



2155 TYPE TRANSFORMER POWER SUPPLY WIRING

Incidentally the PCB hole spacing for the SC/MP clock capacitor is 12.5mm, so that readers who wish to substitute a 1MHz quartz crystal for the existing 470pF capacitor can do so easily. This would perhaps be advisable when and if you wish to interface the Mini Scamp to a terminal, to ensure a stable and predictable data rate.

Except for the SC/MP chip itself, all of the ICs are mounted directly on the PCB without sockets. A high-quality 40-pin socket was used for the SC/MP because of its higher unit cost.

We have not included the LED driver transistors and their associated resistors on the main PCB. This would have committed builders to using the binary LED scheme, and we think that some may prefer to use a pair of 7-segment displays with hexadecimal drivers instead. By leaving the drivers off the PCB, you can take your pick.

The original 9-LED scheme has been retained for our version of Mini Scamp, as you can see. This is because we think beginners find a simple binary display easier to follow. It will also be somewhat cheaper than a hex display!

We wired the 9 driver stages on a small piece of utility PCB, of the type having an array of 4 linked-pad groups. By cutting some of the conductors to form pairs, the drivers were very easily wired, as shown in the small diagram. The piece of utility PCB measures only 85 x 40mm.

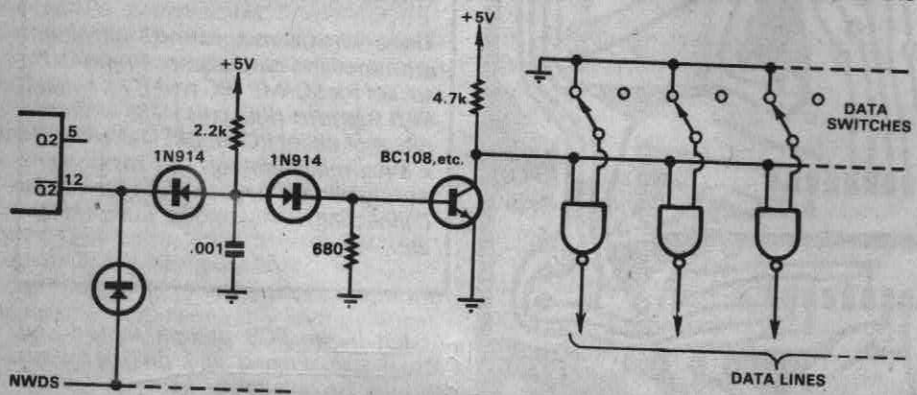
To reduce costs we elected to use one of the imported DSE 2155 transformers. As this provides only 7.5V per side on the secondary, we were unable to use the twin full-wave rectifier circuit suggested by Dr. Kennewell last month. The positive rectifier is unchanged, but we have substituted a half-wave doubler for the negative supply. This runs from the "12.6" tap (i.e., 5V with respect to the earthed centre-tap), to produce the desired –12V.

The only other change to the power supply is that we found it necessary to add a 100uF/16VW electro across the output of the 5V regulator. The wiring of the final supply is shown in a small diagram, so you can copy it if you wish.

One further point: you will find that the PCB incorporates a change to the memory deposit circuit. In particular, we have added an RC delay circuit and a buffer transistor to the drive line for the data switch gates, as shown in the diagram at left. Drive is now taken from the Q2-bar output of the 74123 (pin 12), instead of from the Q2 output.

We found this modification necessary to ensure fully reliable depositing with the 2112 memory devices, which have a critical requirement in respect to data hold time.

Well, there you have it. We think you'll agree that Mini Scamp has become quite an exciting project—and an excellent way to learn about microcomputers.



DEPOSIT TIMING MODIFICATION

Making Mini Scamp bigger and better

Here are the promised details showing you how to expand the basic Mini Scamp microcomputer design into a more powerful one. You can interface it to a teleprinter or a video terminal very easily, and you can also expand its memory to either 1,024 bytes of RAM or 1,280 bytes of mixed RAM and ROM or PROM—all at surprisingly low cost!

by JAMIESON ROWE

In its original form, Dr. Kennewell's Mini Scamp microcomputer is excellent for teaching the fundamentals of microcomputer operation. However once you have learned the fundamentals, it can be a bit frustrating to have to feed your programs in each time via the front panel switches and LEDs. Before long, you will want to add interfacing so that you and your Mini Scamp can talk to each other more conveniently, using a teleprinter or video terminal.

As it happens, you can provide your Mini Scamp with standard 20mA-loop asynchronous serial interfacing very easily, and at low cost. All that is required are two low cost transistors and a handful of other parts, connected to the "Flag-0" and "Sense-B" pins of the SC/MP chip itself (pins 19 and 18 respectively).

The circuit details are shown in Fig. 1. As you can see, the output interfacing from the SC/MP flag pin is simply a BC558 or similar PNP transistor arranged as a switch controlling a 20mA current derived from the +5V line. When the SC/MP flag is at the logic low level, the transistor is turned on and 20mA of current flows out into the teleprinter or terminal. This is the "mark" or idle condition.

If the flag is taken to logic high, the transistor is turned off. The output current to the terminal is thus interrupted, producing the "space" condition. The SC/MP is thus provided with a means of switching the terminal current between the standard mark and space levels, via the level at its flag 0 output.

The input side of the interfacing is equally simple. The keyboard output of the terminal is used to control the conduction of another transistor switch, this time a BC548 or similar NPN device. This is connected in turn between the SC/MP sense-B input and ground, so that it controls the logic level at this input (there is an internal pull-up resistor of around 7.5k, on the SC/MP chip).

The RC circuitry at the input of the transistor is used to suppress contact bounce and any hash which may be picked up by the terminal cable.

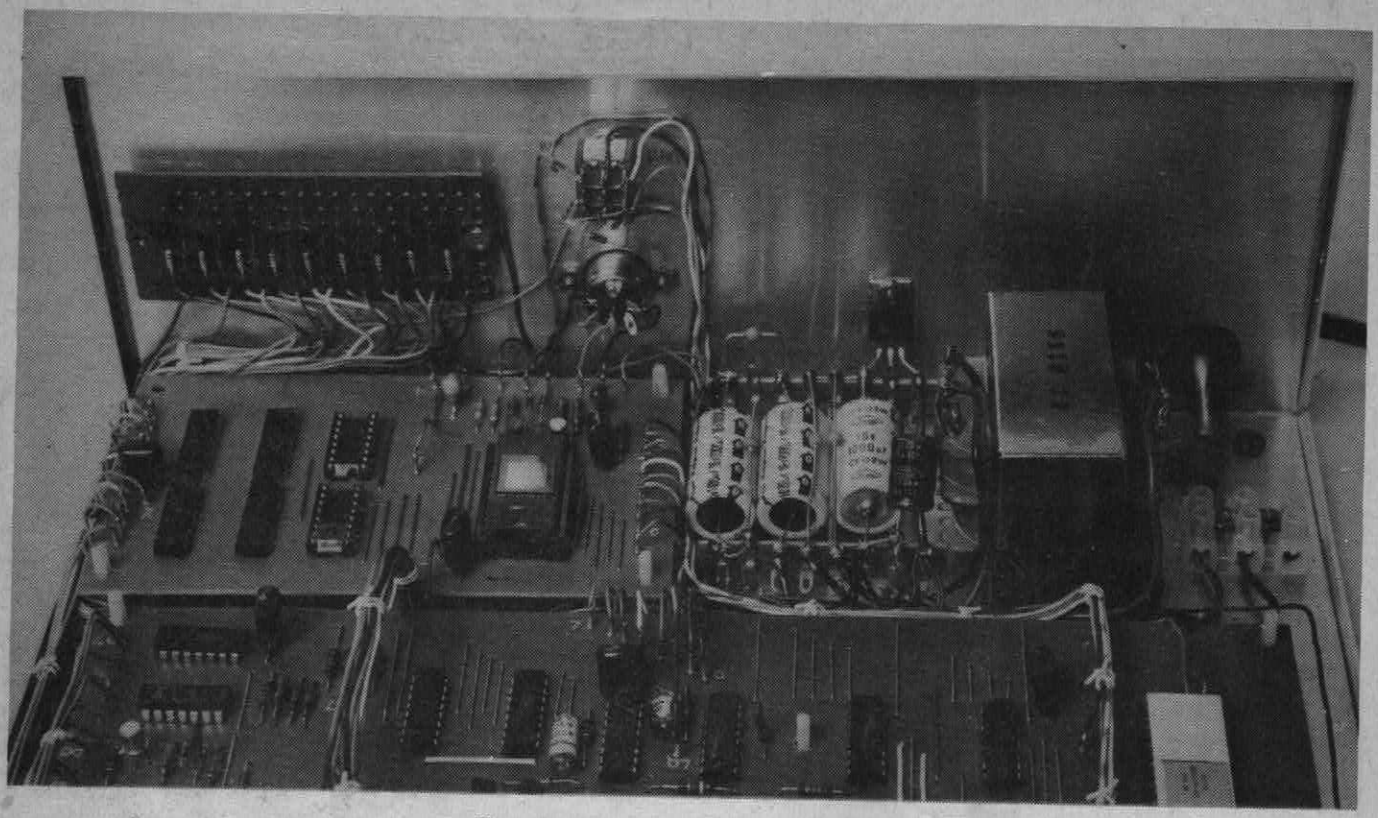
When the keyboard output of the terminal is in the low-impedance "mark" state, it holds the base of the transistor at ground potential. The transistor is thus cut off, and the pullup resistor inside the SC/MP chip pulls the sense-B input up to the logic high level.

If the keyboard output switches to the high impedance "space" state, a forward bias is applied to the transistor base via the divider formed by the 1k, 3.3k and 1k resistors. The transistor thus conducts, pulling the SC/MP sense-B input down to the logic low level.

Hence this simple interface allows a standard teleprinter keyboard or video terminal output to communicate with the SC/MP chip, by controlling the logic level at its sense-B input.

As you can see, the interfacing circuits of Fig. 1 are very basic. They merely provide electrical matching between the SC/MP flag-0 and sense-B pins and a standard 20mA current-loop terminal. There is no special hardware for handling information in asynchronous serial form, or for converting between this form and parallel format. This is all left to the SC/MP itself to take care of, via suitable software "driver" routines.

As well as being very economical, this



The pictures above and on the facing page show the prototype Mini Scamp in expanded form. It now sports a serial terminal interface, a total of 768 bytes of RAM, and a 512-byte ROM with resident monitor.

approach is also a very flexible one. It means that you can change the serial data format very easily, simply by changing the software driver routines. All you need to do to interface Mini Scamp to a terminal with a particular code and data rate is write the appropriate driver routines!

What this means, for example, is that you aren't just limited to using a relatively expensive ASCII-code teleprinter or video terminal. You can use a surplus Baudot-code teleprinter instead, merely by writing a pair of driver routines to suit its 5-bit data format and 50 (or 45) baud data rate.

So that you won't be completely on your own, we are reproducing here listings for a pair of driver routines for talking to a standard 110-baud ASCII teleprinter or video terminal. These should at least give you an idea of what is involved, so that you can write similar routines for other types of terminal.

Both routines are reproduced here by courtesy of National Semiconductor, as they are basically those which come in the "KITBUG" monitor ROM. We have modified them slightly to adapt them for general use in RAM, and deleted the original ROM addresses as these are not relevant.

GECO is the input subroutine, which is called to fetch a character from the terminal and return with it in the SC/MP accumulator. It also echoes the character automatically to the teleprinter or video terminal display screen.

GECO expects pointer register P2 to have been set up as a pointer to a stack in RAM—i.e., it expects P2 to contain the address of the uppermost of a small stack of "scratchpad" memory locations.

You call GECO using an XPPC P3 instruction (hex code 3F), having first set pointer register P3 to the address of the memory byte immediately before the start of GECO.

As you can see GECO returns to your program by performing a similar XPPC P3 instruction. Because this leaves P3 pointing to its own exit, the "JMP GECO" instruction at the end allows you to call the routine again simply by using another XPPC P3 instruction. (In effect, the start of the subroutine for repeated calling is at the end—this is one of the little tricks with SC/MP.)

The second routine PUTC is to take a character in the accumulator, and send it to the terminal. Like GECO, PUTC is called using an XPPC P3 instruction, but of course P3 must in this case have been set up to the address of the byte immediately before either the first or last instruction of PUTC, rather than those for GECO.

PUTC also expects P2 to be set up as a stack pointer, like GECO. In fact if you use the two routines with a program, it is a good idea to use P2 as the stack pointer for your own program, to avoid hassles. It is often convenient to allocate the very top of the RAM as the stack, so that P2 is set up by loading it with the

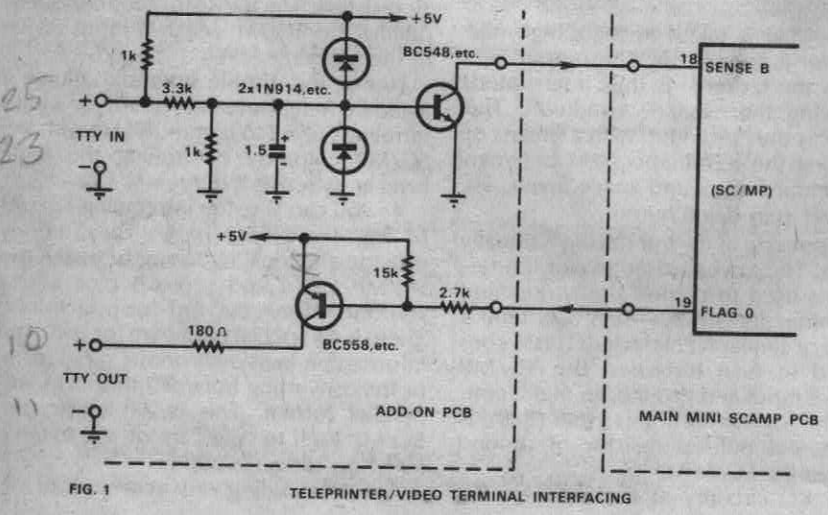


FIG. 1 TELEPRINTER/VIDEO TERMINAL INTERFACING

```
        ; GECO IS USED FOR KEYBOARD INPUT SO IT ECHOS THE
        ; CHARACTER BUT DOES NOT ENABLE THE READER RELAY.
        ;
C408  GECO:   LDI   8          ; SET COUNT = 8
CAFF          ST    -1(P2)
06    $2:     CSA              ; WAIT FOR START BIT
D420          ANI   020
9CFB          JNZ   $2         ; NOT FOUND
C457          LDI   87         ; DELAY 1/2 BIT TIME
8F04          DLY   4
06            CSA              ; IS START BIT STILL THERE?
D420          ANI   020
9CF2          JNZ   $2         ; NO
06            CSA              ; SEND START BIT (NOTE THAT
DC01          ORI   1          ;   OUTPUT IS INVERTED)
07            CAS
C47E  $LOOP:  LDI   126        ; DELAY 1 BIT TIME
8F08          DLY   8
06            CSA              ; GET BIT (SENSEB)
D420          ANI   020
9802          JZ    $3
C401          LDI   1
CAFE  $3:     ST    -2(P2)     ; SAVE BIT VALUE (0 OR 1)
1F            RRL              ; ROTATE INTO LINK
01            XAE
1D            SRL              ; SHIFT INTO CHARACTER
01            XAE              ; RETURN CHAR TO E
06            CSA              ; ECHO BIT TO OUTPUT
DC01          ORI   1
E2FE          XOR   -2(P2)
07            CAS
BAFF          DLD   -1(P2)     ; DECREMENT BIT COUNT
9CE5          JNZ   $LOOP      ; LOOP UNTIL 0
06            CSA              ; SET STOP BIT
D4FE          ANI   0FE
07            CAS
8F08          DLY   8
40            LDE              ; AC HAS INPUT CHARACTER
D47F          ANI   07F
01            XAE
40            LDE
3F            XPPC  P3         ; RETURN
90C1          JMP   GECO
```

```
        ; 'PUTC'
        ; PUT CHARACTER  IN AC TO TTY.
        ; NOTE: TTY LOGIC LEVELS ARE INVERTED FOR OUTPUT
        ;
01    PUTC:   XAE
C4FF          LDI   255
8F17          DLY   23
06            CSA              ; SET OUTPUT BIT TO LOGIC 0
DC01          ORI   1          ;   FOR START BIT. (NOTE INVERSION)
07            CAS
C409          LDI   9          ; INITIALIZE BIT COUNT
CAFF          ST    -1(P2)
C48A  $1:     LDI   138        ; DELAY 1 BIT TIME
8F08          DLY   8
BAFF          DLD   -1(P2)     ; DECREMENT BIT COUNT.
9810          JZ    $EXIT
40            LDE              ; PREPARE NEXT BIT
D401          ANI   1
CAFE          ST    -2(P2)
01            XAE              ; SHIF DATA RIGHT 1 BIT
1C            SR
01            XAE
06            CSA              ; SET UP OUTPUT BIT
DC01          ORI   1
E2FE          XOR   -2(P2)
07            CAS              ; PUT BIT TO TTY
90E8          JMP   $1
06    $EXIT:  CSA              ; SET STOP BIT
D4FE          ANI   0FE
07            CAS
3F            XPPC  P3         ; RETURN
90D4          JMP   PUTC
```

Here are two utility driver routines, designed to service a 110-baud ASCII teleprinter or video terminal. Both assume that the SC/MP clock oscillator is running at 1MHz, so that if you aren't using a crystal you may need to "tweak" the clock capacitor.

address of the top of your RAM area.

And talking about RAMs leads us on to the next area of interest when it comes to expanding Mini Scamp.

Although the 256 bytes of memory provided in the basic Mini Scamp are enough to let you work on quite respectable machine-language programs, the odds are that it won't take long before you'll be wanting to expand the memory to run larger programs. This seems to happen with almost everybody.

As we noted last month, it is quite easy to expand Mini Scamp's memory up to quite a respectable level. In fact if you want to simply provide additional RAM, it is merely a matter of connecting additional pairs of 2112 memory chips up to the address and data buslines, the write strobe line and the appropriate outputs of the 74LS138 address decoder.

You can add up to three additional pairs of RAM chips in this simple way, without exceeding the loading capability of the SC/MP chip. This will give you a total of 1,024 bytes of RAM, or "1k", which should be more than adequate for any machine language program you're likely to write for quite a while.

The basic connections required for this sort of simple RAM expansion are shown in Fig. 2. As you can see, the existing RAMs on the main Mini Scamp PCB remain as the lowest 256 bytes, gated by the 0 output of the decoder as before. The additional pairs are gated by the next three decoder outputs, so that the four pairs provide a continuous range of addresses from 000 to 3FF hexadecimal.

Note that you can't expand to beyond 1k of RAM using this approach, because any further RAM pairs would exceed the SC/MP loading capacity. More about this later . . .

Naturally enough some readers may like to provide their Mini Scamp with the ability to run a program in ROM or PROM memory. For example, you may have an MM5214 ROM with the "Kitbug" monitor program, as supplied with the National Semiconductor SC/MP kit.

Although Kitbug is a fairly basic little monitor program, it does include the terminal driver subroutines—so that your programs can simply call them as required. It also has another subroutine

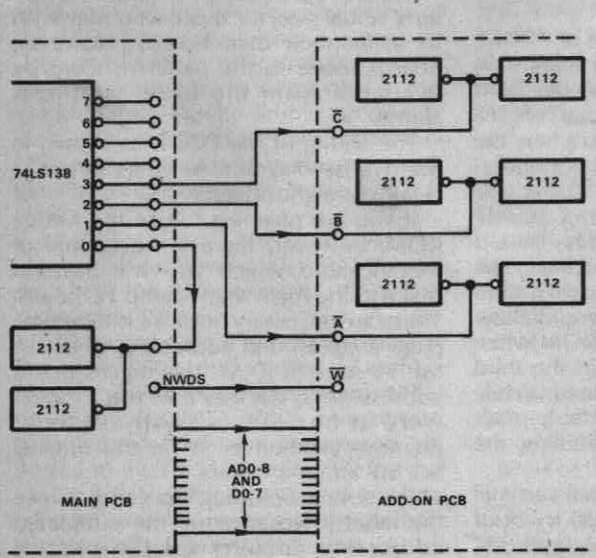


FIG. 2 DECODING FOR RAM-ONLY SYSTEM — TOTAL OF 1k (000-3FF)

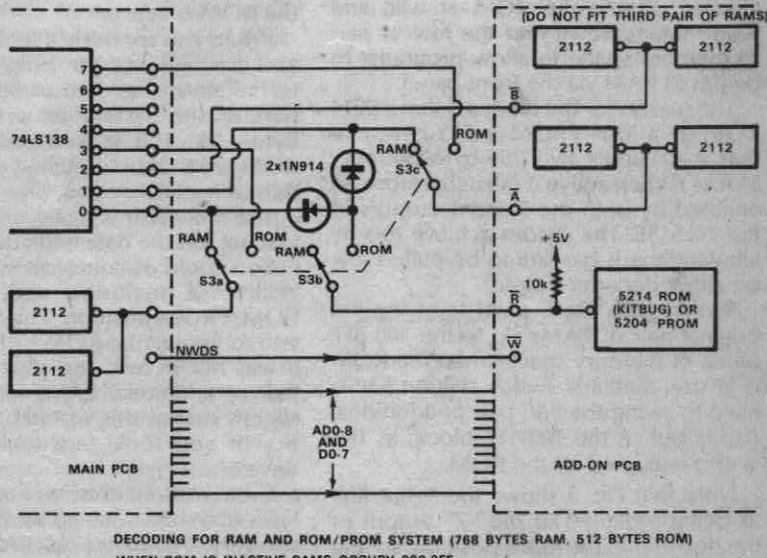
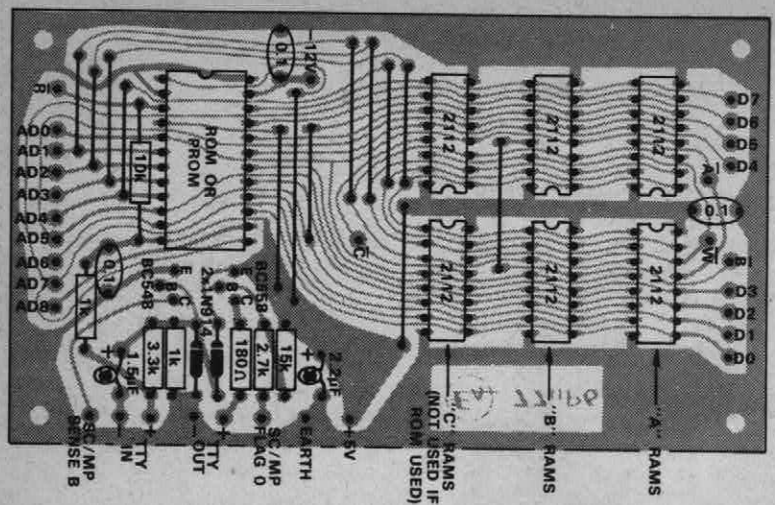
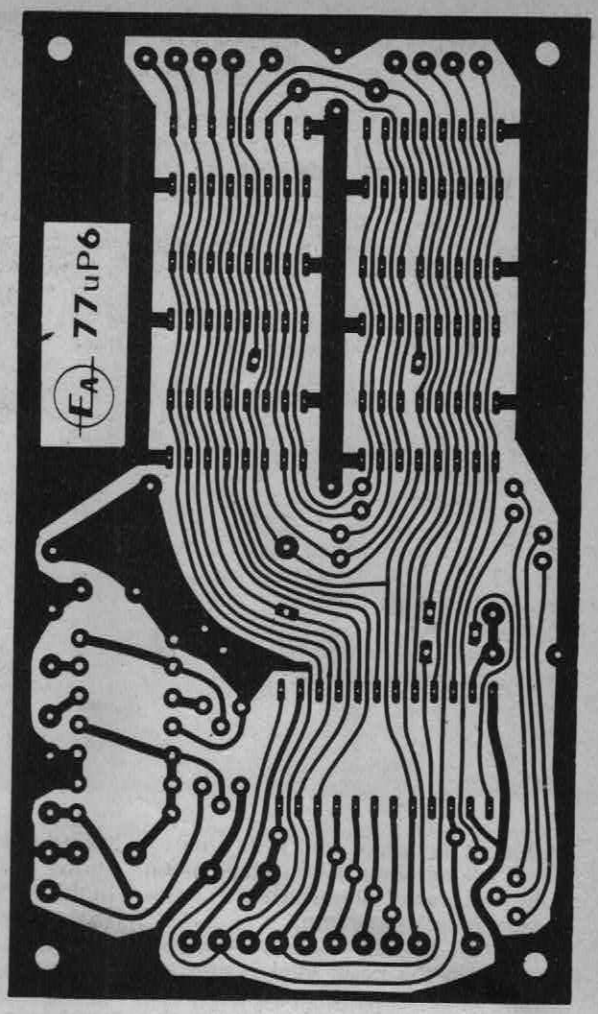


FIG. 3 DECODING FOR RAM AND ROM/PROM SYSTEM (768 BYTES RAM, 512 BYTES ROM)
WHEN ROM IS INACTIVE RAMS OCCUPY 000-2FF
WHEN ROM IS ACTIVE ROM OCCUPIES 000-1FF, RAMS OCCUPY 200-3FF, 700-7FF





At right is the PCB pattern for our add-on memory and interfacing board, shown actual size. The diagram above shows how to wire it up for both RAM and ROM, but note that only two extra pairs of RAMs can be used if you fit a ROM or PROM as well.

which can be used to print out 8-bit numbers in hexadecimal, called PHEX. This is also handy.

And in addition Kitbug provides you with some basic monitor-debug functions, so that you can feed in programs from the terminal keyboard, examine and modify them, and run them. So that it can be very convenient to have Kitbug resident in your Mini Scamp, in a ROM, so that it knows how to do these things right from switch-on.

It is fairly easy to add an MM5214 ROM or an MM5204 UV-erasable PROM to the Mini Scamp, with up to a total of 768 bytes of RAM (i.e., ¾k) as well if desired. The basic circuit is shown in Fig. 3.

As you can see some address switching is required, because SC/MP programs normally start at the bottom of memory space. Hence to run the ROM program it must be switched down in place of the existing RAM.

Although this could be done with fixed wiring, it would then not be possible to run programs in RAM under control of the front-panel switches. They could only be run under the control of the ROM monitor, using a terminal.

By using switching as shown, you are able to disable the ROM at will, and swing RAMs down into the lowest part of memory space to allow programs to be run in RAM via the front panel.

The reason for the diodes is that a 5214 ROM or a 5204 PROM is 512 bytes, so that it occupies two 256-byte memory blocks. When active it must therefore be enabled by both the 0 and 1 outputs of the 74LS138. The diodes achieve this by allowing the R-bar line to be pulled low by either decoder output.

Switch pole S3a is used to swing the original pair of RAMs up to the 300-3FF block of memory space when the ROM is in use. Similarly switch section S3c is used to swing the "A" pair of additional RAMs out of the 100-1FF block, as this is also occupied by the ROM.

Note that Fig. 3 shows the A-bar line as being switched to the "7" output of the decoder in the ROM position, so that the "A" RAMs are swung up to the 700-7FF block of memory space. This is strictly only necessary for the Kitbug ROM, which expects its RAM stack to be at the top of the current memory page.

If you are using some other ROM or PROM, you could elect to switch the "A" pair of RAMs to some other block of memory space by using one of the other decoder outputs instead. If you were to use the "4" output, for example, they would occupy the range 400-4FF when the ROM is active.

When you are using a ROM or PROM, you can only use the simple expansion technique of Fig. 3 to expand the RAM part of the system to a total of 768 bytes—¾k. This is true even when the ROM or PROM is switched out of operation using S3.

The limitation is again one of SC/MP loading, on the data and address lines.

You could of course fit sockets in the "C" RAM positions, and also in the ROM/PROM position. This would allow you to unplug the ROM or PROM when it was not in use, and plug in the third pair of additional RAMs instead. While slightly messy, this would be fairly practical if you took care in handling the devices.

Even without this trick you can still have a system with up to 1280 bytes of memory—768 bytes of RAM, and 512 bytes of ROM/PROM. This is quite a respectable memory.

To help you in expanding your Mini Scamp along the lines we have discussed so far, we have produced a small add-on PC board pattern. Coded 77uP6, the PCB measures only 130 × 75mm. However it lets you provide both the terminal interfacing of Fig. 1 and either the RAM-only expansion of Fig. 2, or the mixed RAM-ROM system of Fig. 3.

We are reproducing here the PCB pattern, actual size, for those who may wish to make their own boards. However, boards made to the pattern should be available from the usual suppliers shortly.

The wiring of the PCB is as shown in the overlay diagram. As you can see, it is fairly straightforward.

If you are planning to use the Kitbug ROM, however, there are a couple of modifications which you will have to make to the main Mini Scamp PC board. These are necessary because Kitbug uses page wrap-around addressing to establish its stack in RAM. This means that it addresses the stack as if its top location were at hex FFF—a "non-existent" memory location even on the original SC/MP kit.

The writers of Kitbug were able to take this liberty because of the simplified addressing circuitry on the original SC/MP kit. In effect, the one pair of

RAMs occupied a number of different blocks in memory space—including both the 200-2FF block and the F00-FFF block.

To allow Kitbug to operate correctly with Mini Scamp, it is necessary to pull a similar trick. In fact the easiest way is to disconnect address line AD11 from the 74LS138 decoder, so that it can no longer distinguish between the first and second 2k blocks of memory. This then allows the "A" RAMs, connected to the 7 output of the decoder, to act as if they are in memory block F00-FFF as well as block 700-7FF.

Of course the other memory blocks become duplicated too, so that the original RAMs will occupy B00-BFF as well as 300-3FF, and the "B" RAMs will occupy A00-AFF as well as 200-2FF. The ROM will also occupy 800-9FF as well as 000-1FF. But these addressing ambiguities will not cause problems, because there are no other devices at the duplicated memory blocks.

To make this modification, you will have to cut the copper track of the AD11 address line, either side of pin 4 of the 74LS138 decoder. Pin 4 of the IC can then be grounded, by linking it across to pin 5. Another insulated link should then be used to rejoin the two severed halves of the AD11 line to each other.

Even with this modification there is still a further complication if you want to use the Kitbug ROM. The simple addressing used in Mini Scamp for the front panel LEDs and switches is no longer adequate, because it clashes with that of Kitbug itself.

In other words, it is necessary to change the addressing of the LEDs and switches. The easiest way of doing this is shown in Fig. 4.

As you can see, it involves removing the original link used to gate G2 and G3 from the AD11 line, and using one of the originally "spare" 74LS05 inverters to gate them instead from the unused "4" output of the 74LS138 decoder.

This is fairly easily done, and involves only an extra 10k resistor and two short lengths of hookup wire.

Note that with this modification done, the front panel LEDs will now have the hex address 402, instead of the original address 802. Similarly the switches will now have the address 401 instead of 801.

If you don't remember this, you may find that programs written for Mini Scamp in its original form won't work! I found this out the hard way, myself. For example with both of Dr. Kennewell's sample programs given in the original Mini Scamp article, you will need to change the first real instruction to LDI 04 (hex code C404), before they will work.

Well, that's the story on how to expand your Mini Scamp easily into quite a respectable little machine. Perhaps there's only one more question to answer: is it possible to expand the system still further?

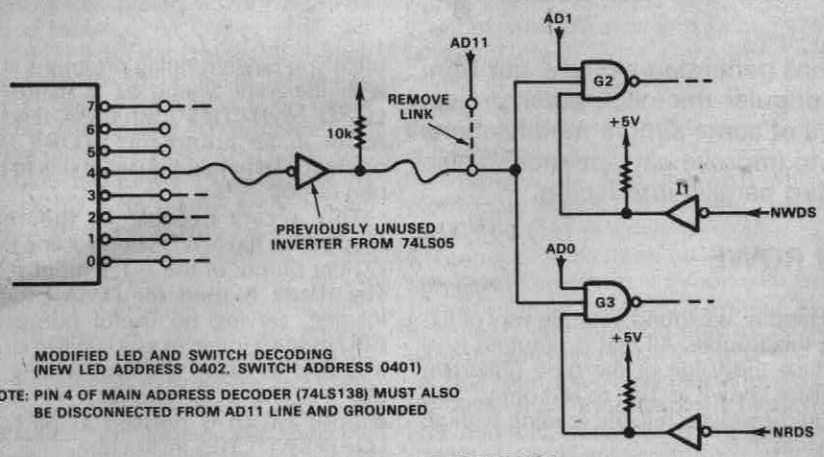
The answer to this is yes, but with qualifications. To add further memory, you will need to use a more thoroughgoing approach. The address lines will need to be buffered, while the data lines will have to be provided with bidirectional bus transceivers. Similarly the address decoding will need to be extended, to provide decoding for more blocks of memory.

It will probably also be necessary to buffer the Write strobe line (NWDS), at least. And of course you will need a suitable PC board to mount the additional memory devices, so there will be the business of mounting all of this additional circuitry in the Mini Scamp case.

While you could probably do all this, you would really be stretching the Mini Scamp concept beyond its normal bounds. The end result would start to become a mechanical monster, and rather hard to troubleshoot if anything goes wrong.

If you really do want to expand to a big system, the best plan might be to change over to a modular system rather than attempt to enlarge the Mini Scamp concept still further.

Most of the parts you have used in Mini Scamp should be suitable for use in a modular system, so that with a little care you should be able to make the change at relatively low cost.



MODIFIED LED AND SWITCH DECODING
(NEW LED ADDRESS 0402, SWITCH ADDRESS 0401)

NOTE: PIN 4 OF MAIN ADDRESS DECODER (74LS138) MUST ALSO BE DISCONNECTED FROM AD11 LINE AND GROUNDED

FIG. 4 MODIFICATIONS FOR KITBUG ROM

More on Mini Scamp

Judging by the tremendous interest it has generated already, our Mini Scamp looks like becoming the most popular microcomputer project ever described. This article gives details of some simple modifications and improvements to the basic design, to improve performance. It also discusses simple techniques for providing parallel interfacing.

by JAMIESON ROWE

One good thing about a popular project like Dr. Kennewell's Mini Scamp is that with so many people building it up quickly, any errors or bugs present tend to show up sooner than otherwise!

For example, the May issue had only been published a couple of days before a reader rang to point out that there was an error in the power supply wiring diagram.

If you didn't see our note last month about that error, it showed the wire from the 12.6V transformer tap as being connected to earth. In fact it should only connect to the "+" end of the 1000uF electrolytic capacitor, of course.

It took a little longer for any other bugs to show up, but after a couple of weeks we received reports of some builders having trouble with the deposit timing circuitry. Some units wouldn't load in programs properly in DMA mode, while others wouldn't load from the switches reliably under program control.

When we looked into this, we found a subtle timing problem associated with the deposit timing delay circuit shown in the May issue. With some 1N914 diodes and BC108 transistors, charge storage times were sufficient to slow down the leading edge of the pulse used to gate the data switches onto the data lines, and improper loading occurred. Usually, some or all of the bits loaded as "1's", regardless of the switch positions.

Happily, we found a simple way of fixing this trouble. All that is required is to reduce the value of the base pulldown resistor shown as 10k, to 680 ohms. This should produce reliable loading with all devices.

In the meantime, Mini Scamp designer Dr John Kennewell advised us that he had found it desirable to add a 180-ohm resistor in series with the deposit switch. This prevents the switch from taking the B1 input of the 74123 monostable right up to the 5V rail, and thus prevents the monostable from being spuriously triggered by supply line transients.

Incidentally you may have noticed that while Dr Kennewell's original circuit as shown in the April article shows a DPDT switch used for S1, performing the DMA/CPU switching, we called for only an SPDT switch in the parts list. This is because the SPDT switch is quite capable of doing the job, if its rotor is taken to earth. In the DMA position it earths the two wires from the PCB pads marked "DMA/CPU SW A", while in the CPU position it earths the single wire from the PCB pad marked "DMA/CPU SW B". Most builders seem to have deduced this for themselves, but I mention it in case you haven't got that far as yet.

If you make these few minor corrections to your Mini Scamp, you should find that it works quite well. However something which may become apparent once you begin running programs is that with the Mini Scamp as it stands, the LOAD SWITCHES instruction tends to result in an automatic STORE LEDS operation, whether you want this to happen or not.

This occurs because of the 1N914 diode tying the NWDS control line to the Q2-bar output of the 74123 monostable. The diode is used for DMA program loading, serving no useful purpose in CPU mode. However as it is still in circuit, it is able to cause spurious writing into the 74C175 LED latches, when the deposit switch is pressed as part of a LOAD SWITCHES instruction.

Again, there is a fairly simple way of preventing this from happening. All that is required is to replace the existing SPDT switch used for DMA/CPU selection, with a DPDT type. The second section of the new switch is then used to break the connection between the 1N914 diode and the NWDS line, in the CPU position.

You don't need to cut any PCB tracks to make this change, because the diode is normally connected to the NWDS line via a wire link. All you need do is cut the link, and connect the two cut ends to the appropriate lugs of the new DMA/CPU switch.

There is another modification to the basic Mini Scamp design which some builders may care to make. Although it requires a little more work, it is still fairly straightforward and involves only a handful of additional low-cost parts.

The purpose of this further modification is to ensure that only a single loading takes place from the switches when the deposit switch is pressed in CPU mode, in response to a LOAD SWITCHES instruction having lit the DRQ LED. With the existing Mini Scamp circuit multiple

loading can occur if the CPU executes another LOAD SWITCHES instruction while the deposit switch is still pressed, as the 7476 DRQ latch can re-trigger the 74123 via inverter I4. This is a side effect, because I4 is in circuit mainly to ensure that the 74123 can only be triggered in CPU mode when the DRQ latch has been set for a LOAD SWITCHES instruction. If the 74123 were to be triggered at other times, a program could be changed and caused to "crash".

The only way of preventing multiple loading with the circuit as it stands is to insert delay instructions into your program, so that one has time to release the deposit button before the CPU can get to the next LOAD SWITCHES instruction. This is not always convenient.

To obviate the multiple loading, I4 can be used to gate the 74123 by means of its clear input, instead of the A1 trigger input. This is done as shown in Fig. 1.

Note that the A1 input of the 74123 is now earthed directly, with the output of I4 now taken to the active-low clear input on pin 3. As I4 is an open-collector element, a 10k pullup resistor must be added as shown.

Diode D1 from the DMA/CPU switch is now taken to the input of I4, so that the 74123 can be enabled in DMA mode as before. The input of I4 is disconnected from the Q input of the 7476 DRQ latch, and connected to the Q-bar output via diode D2 to achieve the correct logic action. A further 10k resistor is used at the input of I4, to form a single OR gate with the two diodes.

Although this fixes the multiple loading problem, one can still get occasional faulty loads from the switches. This seems to be due to spurious triggering of the 74123 from the deposit switch, as a result of the low slope produced by the simple R-C bounce integrator.

Presumably the input of the 74123 can become unstable during the transition from logic low to logic high levels.

To fix this we suggest that you replace the deposit switch with a SPDT type, and use the originally redundant second half of the 7476 device as a debounce latch, as shown in Fig.1. This gives completely reliable operation.

Note that Fig. 1 also shows the optional switching between diode D3 and the NWDS line, and the 680 ohm base resistor in the deposit timing delay circuit.

To make these suggested modifications, you need to cut a small number of PCB tracks and add a few links under the PC board. We suggest you do this in the following order, to make sure that you don't lose track of anything.

1. change the 10k resistor at the base of the BC108 used to delay the 74123 Q-bar output signal fed to the data switch control gates to 680 ohms.
2. cut the track between pin 1 of the 74123 and pin 10 of the 74LS05.
3. cut the track between pin 10 of the 74LS05 and the anode of D1.
4. cut the track linking pin 11 of the 74LS05 to the track joining pin 13 of the 74LS05 and pin 15 of the 7476.
5. remove the 470 ohm resistor and 47uF tantalum electrolytic capacitor connected to PCB point "DEP".
6. unsolder the wire connecting the deposit switch to the PCB point "DEP", at the deposit switch.
7. replace the deposit switch with a momentary contact SPDT push-button (C&K type 8121 or similar).
8. connect the NO and NC contacts of the deposit switch to +5V rail with two 10k resistors (at the deposit switch).
9. connect the common contact of the deposit switch to GND.
10. drill two holes near pins 7 and 8 of the 7476, and connect these pins to the NO and NC contacts of the deposit switch respectively.
11. drill a hole near pin 11 of the 7476, and connect the loose wire attached to PCB point "DEP" to pin 11 of the 7476.
12. connect pin 3 of the 74123 to pin 10 of the 74LS05.
13. connect pin 1 of the 74123 to pin 8 of the 74123 (i.e., earth).
14. connect the anode of D1 to pin 11 of the 74LS05.
15. connect a 10k resistor from the anode of D1 to the +5V rail (pin 14 of the 74LS05).
16. connect the anode of additional diode D2 to pin 11 of the 74LS05, and the cathode to pin 14 of the 7476.
17. connect a 10k resistor from pin 10 of the 74LS05 to the +5V rail (pin 14 of the 74LS05).
18. join pins 2, 4, 6, 9, 12 and 16 of the 7476 to pin 11 of the 74123, and then connect them to the +5V rail via a 1k resistor.

With these modifications, your Mini Scamp should leave nothing to be desired in terms of its internal operation. But perhaps a few words are now in order for the benefit of those readers who are planning to use their Mini Scamp to control external devices.

For simple applications, you may be able to use direct interfacing to the flag and sense lines of the SC/MP chip itself, rather like that used for the serial interface shown last month. The SC/MP chip has three flag outputs and two sense inputs, and also two other pins designated "serial in" and "serial out". This allows for a total of seven input-output lines.

Fig. 2 shows how one of the flag lines or the SOUT line could be used to control a small relay. It also shows how a SPDT switch can be debounced and fed to one of the sense inputs, or the SIN input.

For more complex applications, you may wish to provide Mini Scamp with full 8-bit parallel interface ports. This can be done fairly simply, providing you can write your program so that it isn't worried

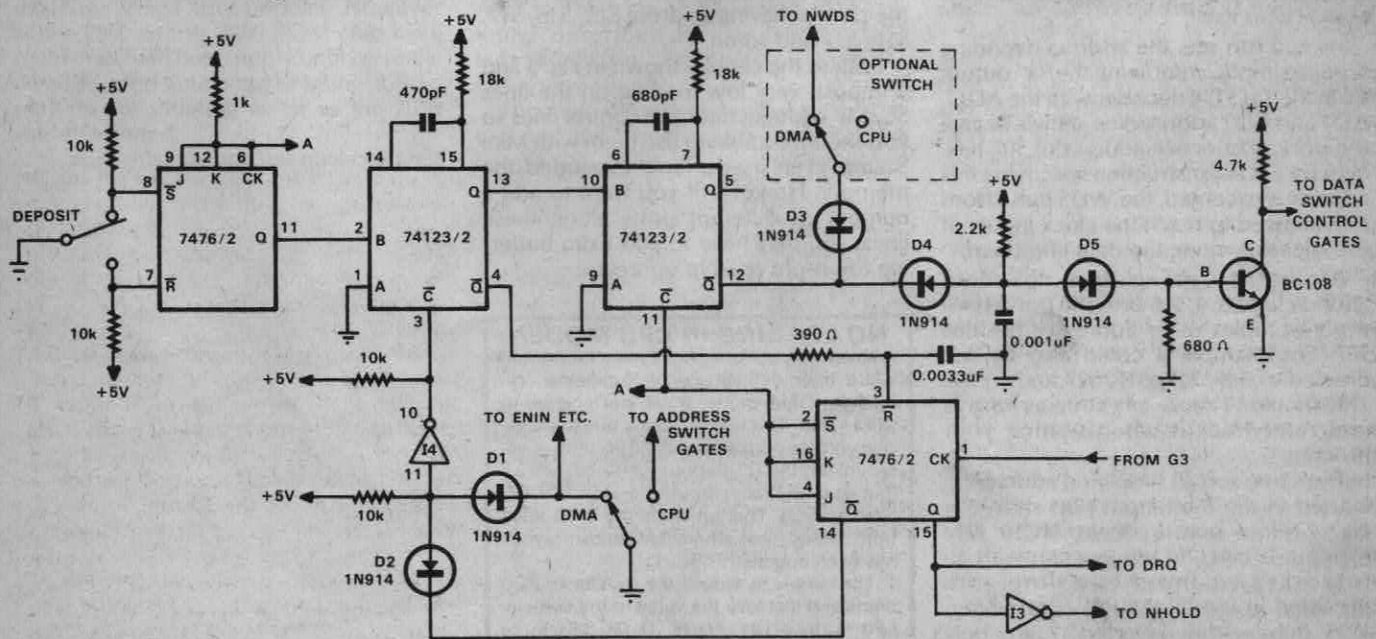


FIG. 1: MODIFIED MINISCAMP DEPOSIT SWITCH LOADING LOGIC

File No. 8/M/—

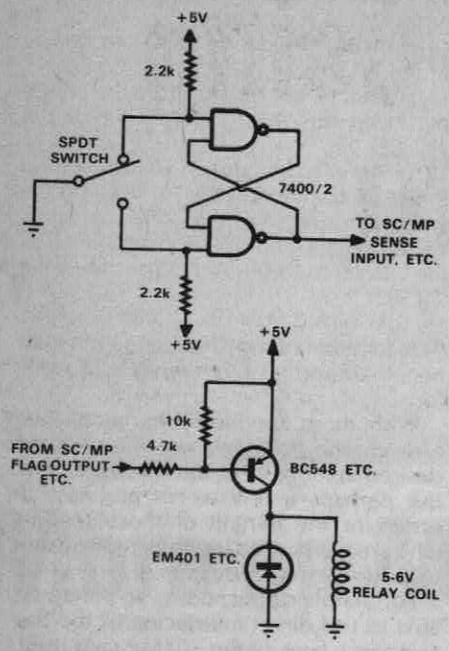


FIG. 2: SIMPLE RELAY & SWITCH INTERFACING

The three diagrams above and at right should give you some guidance on providing your Mini Scamp with interfacing to the outside world.

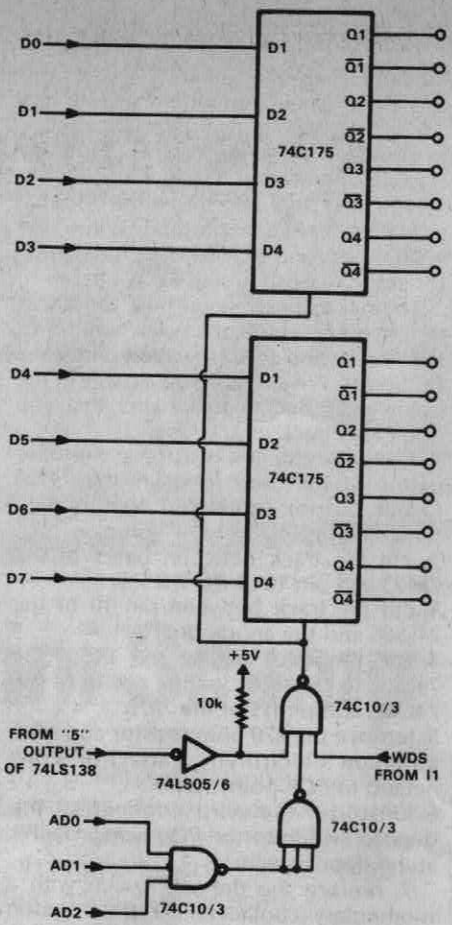


FIG. 3: SIMPLE LATCHED 8-BIT OUTPUT PORT (ADDRESS 507 HEX)

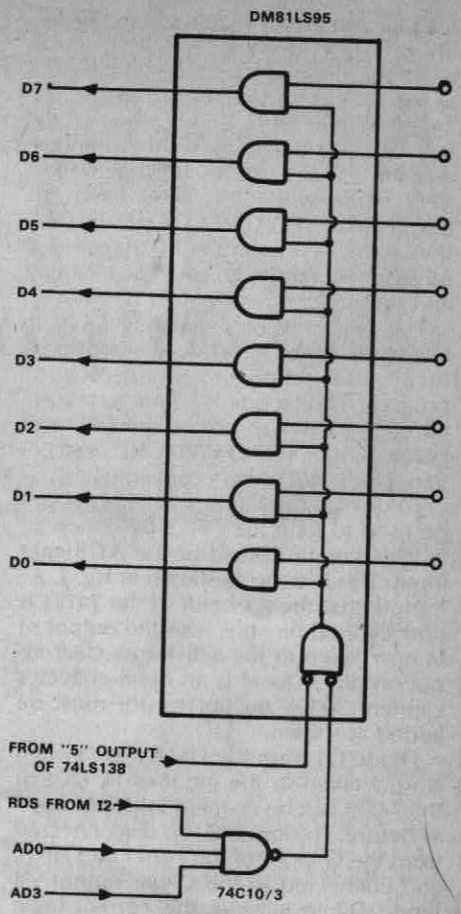


FIG. 4: SIMPLE 8-BIT INPUT PORT (ADDRESS 511 HEX)

by the ambiguous addressing which results from incomplete decoding.

A simple latched 8-bit output port is shown in Fig. 3. Two 74C175 devices are used for the latch itself, providing both active-low and active-high outputs for the eight bits. These outputs could be used to drive external circuits via buffers like those used for driving the LEDs in Mini Scamp itself.

As you can see, the address decoding is quite simple, combining the "5" output from the 74LS138 decoder with the AD0, AD1 and AD2 address line signals to give the port an effective address of 507 hex. When a STORE instruction specifying this address is executed, the WDS pulse from I1 is allowed to reach the clock inputs of the latches, writing the data into them.

Because the gating does not sense address lines 3, 4, 5, 6 or 7, the port effectively occupies other addresses besides 507. For example it could also be addressed as 50F, 517, 51F, 527 and so on. This shouldn't cause any strife as long as you remember it when writing your programs.

The same sort of simplified addressing is used in the 8-bit input port shown in Fig. 4. Here only a single 74C10 gate element is used, in conjunction with an internal two-input enabling gate provided in the DM81LS95 octal buffer. RDS pulses from inverter 12 are only effective in enabling the buffer when address lines AD0 and AD3 are high and the "5" output of the 74LS138 decoder is low, corresponding to address 511 hex.

Hence you can read the data from the port into the SC/MP accumulator by executing a LOAD instruction which references address 511. But because of the simple decoding, you can also regard the port as having address 513, 515, 517, 51B, 531 and so on.

Both of the circuits shown in Fig. 3 and 4 impose very low loading on the lines SC/MP address, data and control lines so you should be able to use them with Mini Scamp even if you have expanded the memory. However if you want to add a number of different ports along these lines, you may have to add extra buffering to ensure reliable operation.

NO LOADING IN CPU MODE?

A small number of readers have found that while their deposit switch functions correctly in DMA mode, it will not operate in CPU mode. This appears to be due to excessive voltage drop across the 390 ohm resistor linking the Q1-bar output of the 74123 (pin 4) to the reset input of the 7476 DRQ latch (pin 3). This prevents the DRQ latch from being reset when the deposit switch has been operated.

The cure is to reduce the resistor to 220 ohms and increase the value of the associated capacitor from 0.0033uF to 0.0068uF.