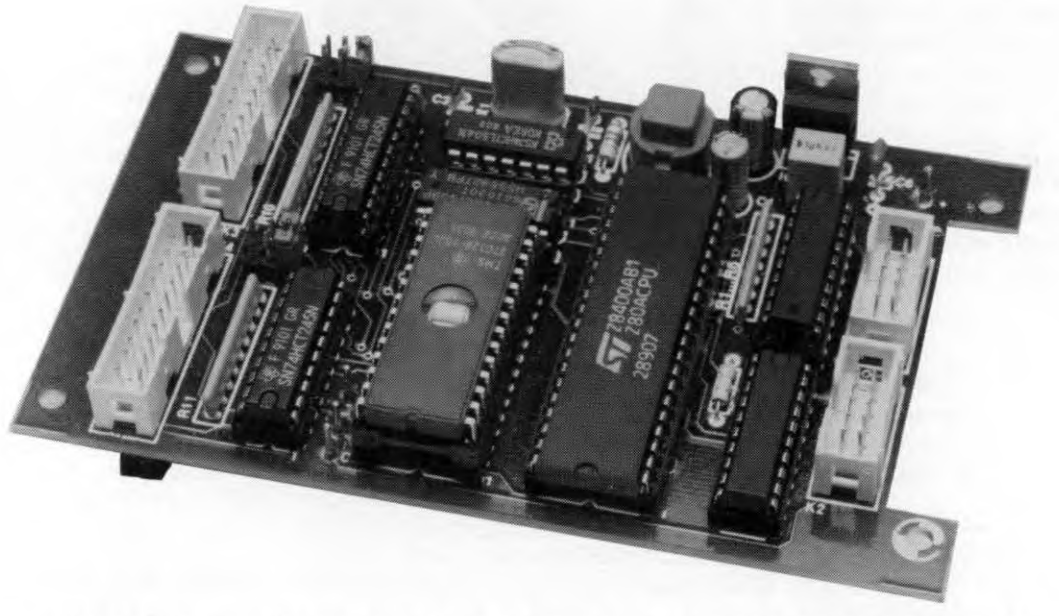
MINI Z80 SYSTEM



Call it what you like: an all-time favourite, an evergreen in computer land, or just a die-hard electronic component: the Z80 8-bit microprocessor enjoys tremendous popularity because it is inexpensive, widely available and easy to program. Furthermore, a massive amount of software and paperware is available for this powerful CPU. Here, we present a no-frills miniature computer system based on the Z80 CPU, with I/O and ROM. No RAM, no, because that is not strictly necessary for small applications if your programming is up to scratch (but we have a RAM extension up our sleeve).

by A. Rigby

PROBABLY the most remarkable feature of the present computer board is the absence of RAM (random access memory). This is unusual, but in many cases the internal registers of the Z80 can function as RAM equally well. Omitting a RAM IC then allows us to cut down on components (cost), and save board space.

The block diagram of the Z80 system is shown in Fig. 1. Remarkably, the arrangement of the functions corresponds closely to that of the associated ICs on the circuit board. In fact, Fig. 1 shows the classic setup of a microprocessor system. The Z80 CPU (central processing unit) uses I/O-mapped input-output operations, which means that the CPU works with different addresses for the memory and the I/O blocks. The present system has four I/O addresses, although two further blocks of four addresses may be selected via the two I/O ports.

The I/O ports available in the system are compatible with the universal I/O interface for IBM PCs (Ref. 1), which allows the extensions originally developed for this to be connected without problems (for instance, the relay card discussed in Ref. 2).

From the block diagram it may appear that the memory address decoder is a superfluous luxury: there is only one EPROM, and that could have been connected direct to the CPU without a decoder. The decoder, however, divides the memory range into four blocks of 16 Kbyte each, and so allows RAM

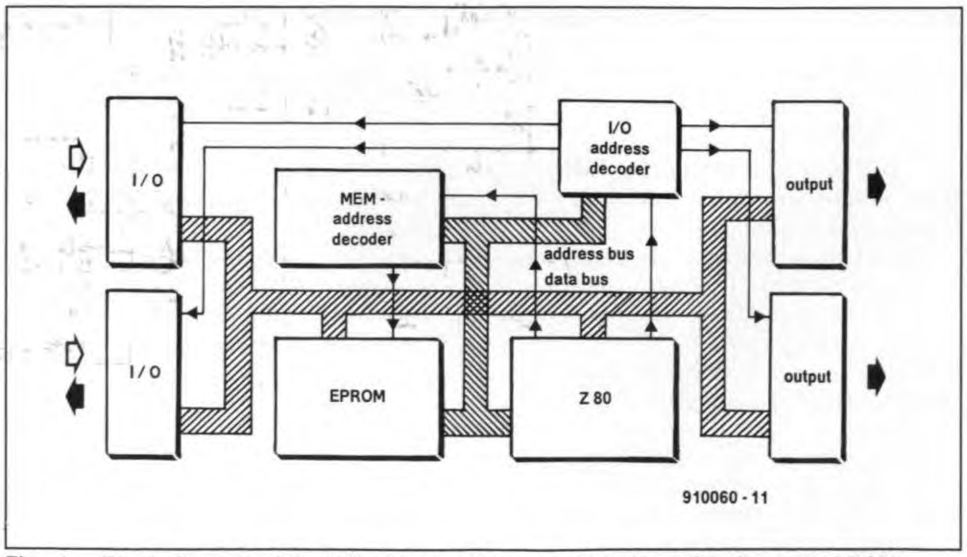


Fig. 1. Block diagram of the Z80 microprocessor system. Note the absence of RAM.

to be added (more about this next month), or more EPROMs. Moreover, the memory address decoder is formed by the remaining two 1-of-4 decoders contained in the IC used for the I/O address decoder. This means that although it may not be used in many cases, the memory decoder does not require additional hardware anyway.

Circuit description

The circuit diagram shown in Fig. 2 closely resembles the block diagram. Only two ICs have been added: a voltage regulator and a hex inverter. Three inverters in the latter IC (IC8), are used to implement the 2-MHz clock oscillator. The unused control inputs of the Z80 (IC1) are held at +5 V via a pull-up resistor array. The reset input of the CPU is connected to an *R-C* network and a switch to ground (S1) that allows the system to be reset. The control, address and data lines of the Z80 are connected to the I/O and memory sections in the usual way.

Address decoder IC7a divides the 64-KByte memory range into four sections of

MAIN SPECIFICATIONS

CPU:	Z80
Clock:	2 MHz
Memory:	8 K ROM (EPROM 2764) or 16 K ROM (EPROM 27128)
I/O:	2×8 bit input/output; 2×8 bit output
Option:	8 K RAM extension

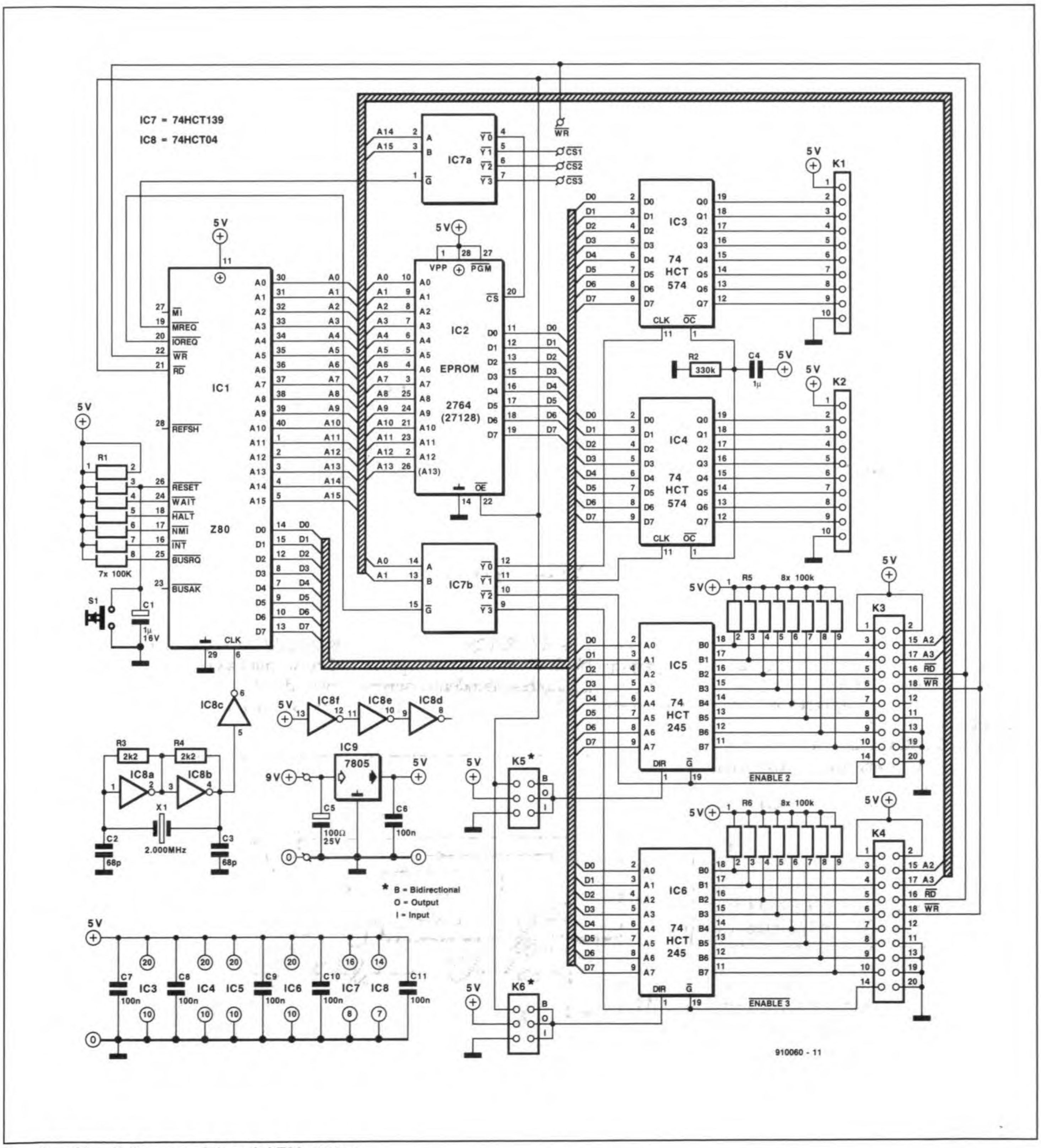


Fig. 2. Circuit diagram of the mini Z80 system.

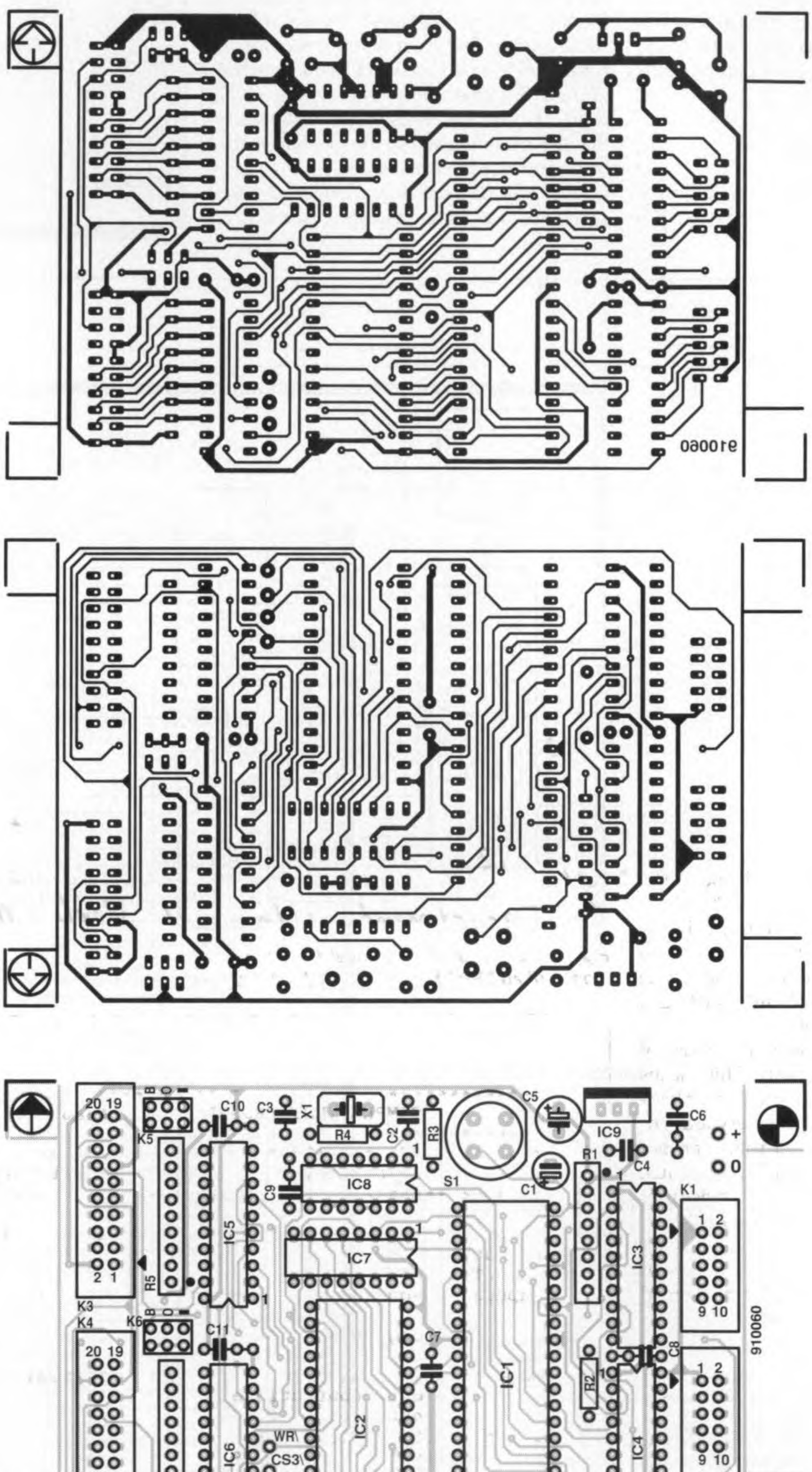


Fig. 3. Track layouts (mirror images) and component mounting plan of the PCB for the mini Z80 system.

COMPONENTS LIST

Resistors:

1	7-way SIL array 100kΩ	R1
1	330kΩ	R2
2	2kΩ2	R3;R4
2	8-way SIL array 100kΩ	R5;R6

Capacitors:

1	1µF 16V radial	C1
2	68pF	C2;C3
1	1µF solid MKT	C4
1	100µF 25V radial	C5
6	100nF	C6-C11

Semiconductors:

1	Z80 CPU	IC1
1	2764 or 27128	IC2
2	74HCT574	IC3;IC4
2	74HCT245	IC5;IC6
1	74HCT139	IC7
1	74HCT04	IC8
1	7805	IC9

Miscellaneous:

2	10-way box header	K1;K2
2	20-way box header	K3;K4
2	6-way box header	K5;K6
1	push-button	S1
1	quartz crystal 2.00MHz	X1
1	Enclosure 150×80×55mm, e.g. Bopla E440VL	
1	printed circuit board	910060

16 kByte, so that so-called mirror areas are avoided. Only if an 8-KByte EPROM is used, its contents are duplicated in the upper half of the 16-KByte area reserved for it. The EPROM on the mini Z80 card is located in the range that starts at address 0000, where the CPU starts after a reset. The other signals supplied by the MEM address decoder, and the write signal ($\overline{WR}$), are available on the PCB for use by external circuits, such as a RAM or EPROM extension.

I/O address decoder IC7b makes use of address lines A0 and A1 only. This means that the I/O addresses do have 'mirror locations': the same IC is selected every four addresses. Since the read and write lines are not used in the I/O address decoder, you must take care not to write to input devices, or read from output devices, on penalty of destroying the CPU. Note, however, that addresses 0 and 1, at which the two 8-bit outputs, IC3 and IC4, reside, can be read without problems.

The next two addresses, 2 and 3, are occupied by two bidirectional ports, IC5 and IC6, and must be used more carefully. These ports can be set to function as an input, an output, or a bidirectional device with the aid of jumpers. When IC5 or IC6 are used as input devices, a write operation to them may damage the CPU or the addressed IC, since in that case two outputs are interconnected, which results in a virtual short-circuit. When used as output devices, a read operation to IC5 or IC6 is not harmful. In bidirectional mode, the CPU is protected reasonably well against output conflicts, and it then depends

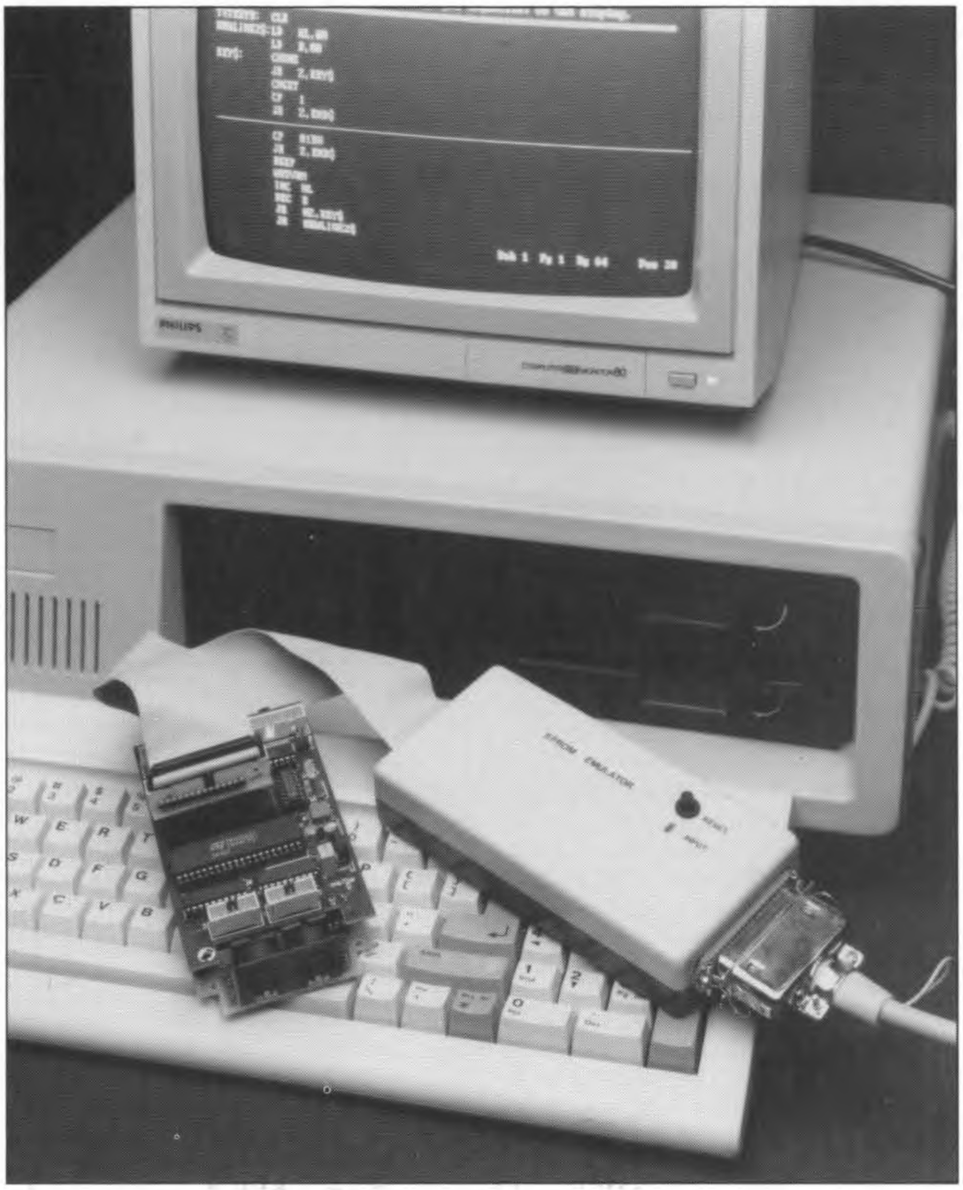


Fig. 4. Undoubtedly the most flexible way of developing software for the Z80 board is by means of an EPROM emulator and a PC running a Z80 assembler.

```
;*********************************************************
;          SIMPLE TEST PROGRAM FOR MINIZ80 BOARD
;*********************************************************
;          IC5 and IC6 jumpers set for input only
;          Be careful, a write can be destructive
;*********************************************************

output1:   equ   0                  ;address for K1
output2:   equ   1                  ;address for K2
input1:    equ   2                  ;address for K3
input2:    equ   3                  ;address for K4

           org   00h                ;program start address

begin:     ld    a,0                ;initialize outputs
           out   (output1),a
           out   (output2),a
loop:      in    a,(input1)         ;read input from K3
           cpl                      ;invert each bit
           out   (output1),a        ;output to K1
           in    a,(input2)         ;read input from K4
           cpl                      ;invert each bit
           out   (output2),a        ;output to K2
           jr    loop               ;loop to check inputs
                                              910060-13
```

Fig. 5. A simple test program that presents input data in inverted form at the output.

Table 1. I/O address overview

Port	I/O address	Connector
IC3	0	K1
IC4	1	K3
IC5	2+4n	K3
IC6	3+4n	K4

(n = 0 – 3)

on the level supplied by the circuit connected to connector K3 or K4 whether or not a dangerous situation can arise from reading from, or writing to, IC5 or IC6. In addition to the eight datalines, connectors K3 and K4 therefore also carry the $\overline{RD}$ and $\overline{WR}$ signals, which indicate read and write operations respectively, and so enable an external circuit to disable its inputs or outputs accordingly. There are more signals on K3 and K4: one enable signal, and two address lines, A2 and A3. These allow the Z80 card to work with extensions originally developed for the universal I/O interface for PCs.

The enable signal on the extension connectors indicates that the I/O lines are addressed, so that the circuit connected can start reading or writing data. Address lines A2 and A3 give us access to a total of eight external addresses: four each via each extension connector, as shown in Table 1.

To prevent the output ports being left in an undesired state after switching on the system, the outputs of IC3 and IC4 are briefly switched to high impedance with the aid of network R2-C4, whose *R-C* constant is about three times greater than that of the CPU reset network. Provided the relevant instructions are placed right at the start of the program (i.e., from 0000 onwards), network R2-C4 affords sufficient time for the CPU to initialize the outputs properly and prevent output-to-output conflicts.

The system is completed by a voltage regulator, IC9. This allows a mains adaptor with an output rating of 9 V to 15 V d.c. at 300 mA to be used, which is a safe as well as inexpensive way of powering the computer.

Construction

The design of the double-sided through-plated board used for building the Z80 system is shown in Fig. 3. This board is available ready-made through our Readers Services.

Construction is straightforward work, and merits no further discussion. Application circuits can be connected to the Z80 board either via short lengths of flatcable, or direct via the connectors. In the latter case, the Z80 card is best fitted on top of the application circuit. This ensures that the EPROM socket remains accessible for a new EPROM, an EPROM emulator, or a RAM extension. As shown in one of the photographs, such an assembly is obtained by fitting sockets to the underside of the Z80 card (see Fig. 8).

The fixing holes in the PCB are located such that the completed board is easily fitted

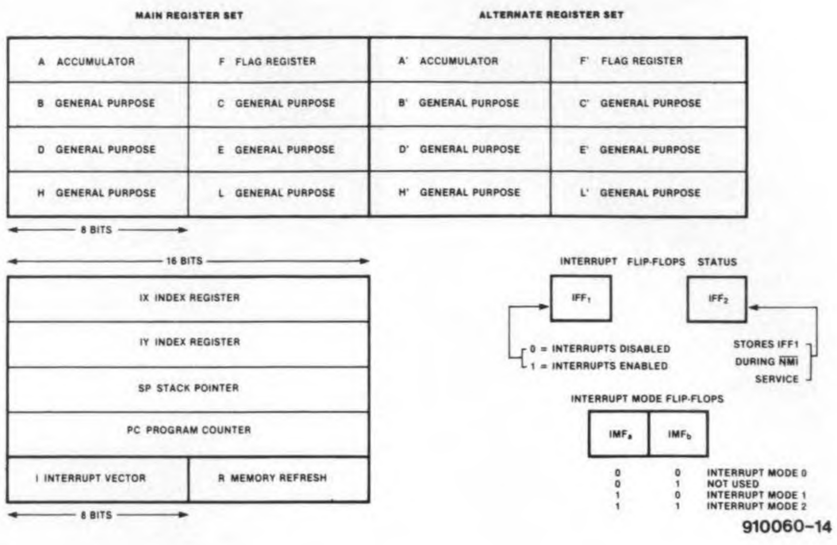


Fig. 6. Z80 register overview.

```
;*****************************************************
;        SIMPLE TEST PROGRAM FOR MINIZ80 BOARD
;        USING MACRO'S
;*****************************************************
;        IC5 and IC6 jumpers set for input only
;*****************************************************

output1:  equ  0                    ;address for K1
output2:  equ  1                    ;address for K2
input1:   equ  2                    ;address for K3
input2:   equ  3                    ;address for K4

;*****************************************************
;        MACRO DEFINITION
;*****************************************************

init_to_zero:
          .MACRO
          ld   a,0                  ;load initial value
          out  (output1),a          ;set output to zero
          out  (output2),a          ;set output to zero
          .MACEND

A_to_output:
          .MACRO output             ;define variable: output
          cpl                       ;invert each bit
          out  (output),a           ;output contents of A
          .MACEND

input_to_A:
          .MACRO input              ;define variable: input
          in   a,(input)            ;read input into A
          .MACEND

;*****************************************************
;        MAIN PROGRAM
;*****************************************************

          org  0h                   ;program start address

begin:    init_to_zero              ;initialize outputs

loop:     input_to_A input1
          A_to_output output1
          input_to_A input2
          A_to_output output2
          jr   loop                 ;loop to check inputs
```

910060-15

Fig. 7. Because of the absence of RAM, subroutines are best replaced by macros during the assembly phase.

Table 2. Non-executable instructions (without RAM)

CALL	cc,nn	;conditional subroutine ;call
CALL	nn	;direct subroutine call
IM	1	;interrupt mode
IM	2	;interrupt mode
POP	IX	;get data from stack
POP	IY	;get data from stack
POP	qq	;get data from stack
PUSH	IX	;put data onto stack
PUSH	IY	;put data onto stack
PUSH	qq	;put data onto stack
RET		;return from subroutine
RETI		;return from interrupt
RETN		;return from non- ;maskable interrupt

qq = AF, BC, DE or HL
cc = condition to execute instruction
nn = address

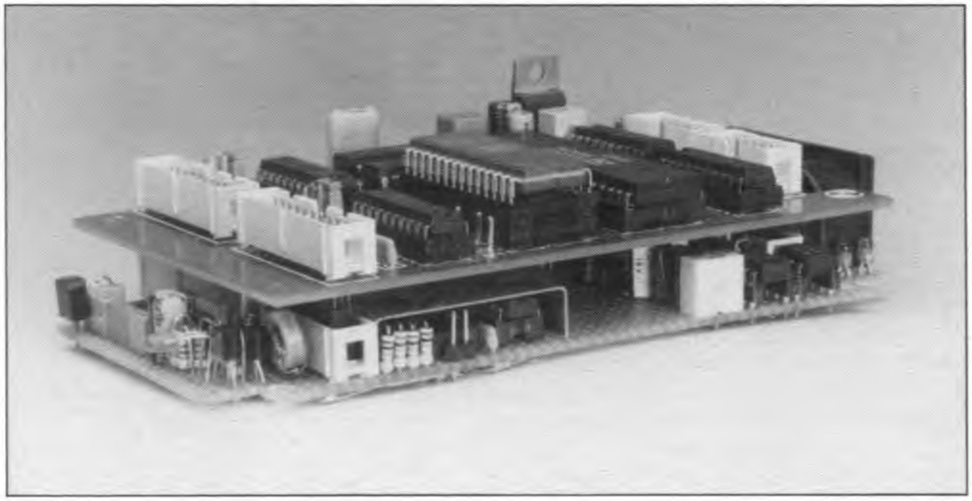


Fig. 8. Z80 board fitted 'piggy back' on to a prototype of the RC5-code infra-red receiver described elsewhere in this issue.

into the enclosure mentioned in the parts list. Where the board is used as a controller for one application only (as discussed above), the sections with the fixing holes in them may be cut off to reduce the board size even further.

Writing programs

As already mentioned, a massive amount of literature exists on programming the Z80. In addition, assemblers and cross-assemblers for the Z80 are widely available. Lacking these, you may still program the Z80 purely by hand, i.e., by writing a machine code program, looking up the opcodes, and loading them into an EPROM. A far more flexible way of developing software is afforded by an EPROM emulator (Fig. 4), which goes round the problem of having to erase and re-program an EPROM every time a change is required (and debugging, as you probably know, almost invariably involves a lot of changes).

For now, you probably want to know if the card works. Well, that can be found out quite easily with the aid of the test program listed in Fig. 5. Set the two jumpers on the board to input before running the program (from EPROM). The program turns the Z80 card essentially into an input data operator. As you can see, this can be achieved without RAM, since the (many) registers of the Z80 can be used for the 'scratch' functions. Figure 6 shows a register overview of the Z80.

The absence of RAM means that a number of Z80 instructions can not be used. These instructions, listed in Table 2, are essentially those related to stack operations. As you can see, it is not possible to call interrupt and subroutines if you do not have RAM. Fortunately, this need not result in 'spaghetti software', because most assemblers support the use of macros. Macros are small pieces of machine code that are used frequently in a program, and which need to be written in source code only once. In most cases, it is possible to use variables in macros, for instance, to indicate the register or address the macro is to make use of. An example of a piece of source code containing macros is given in Fig. 7. ■

References:
1. "Universal I/O interface for IBM PCs". *Elektor Electronics* May 1991.
2. "Relay card for universal I/O interface". *Elektor Electronics* November 1991.

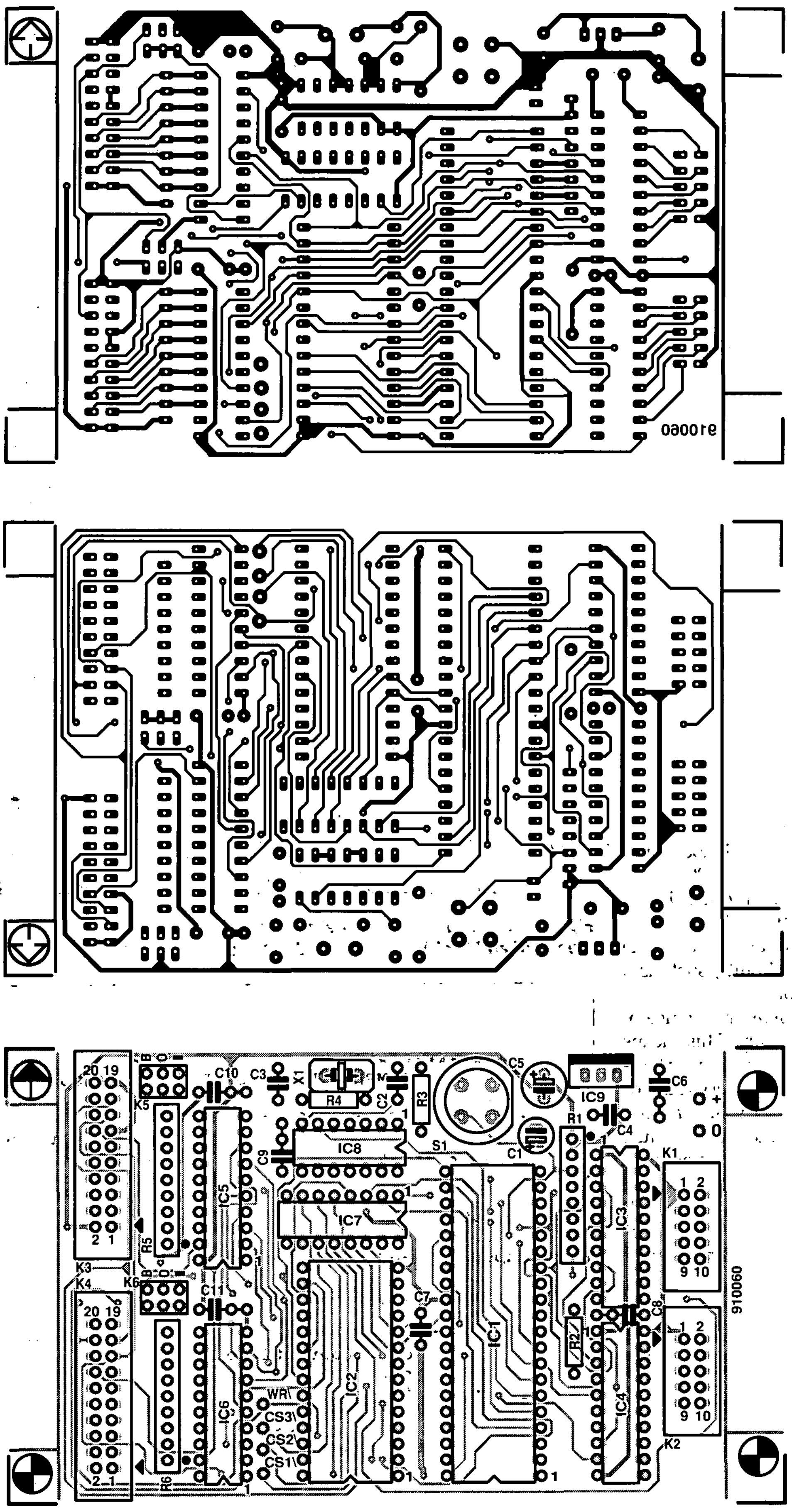


Fig. 3. Track layouts (mirror images) and component mounting plan of the PCB for the mini Z80 system.