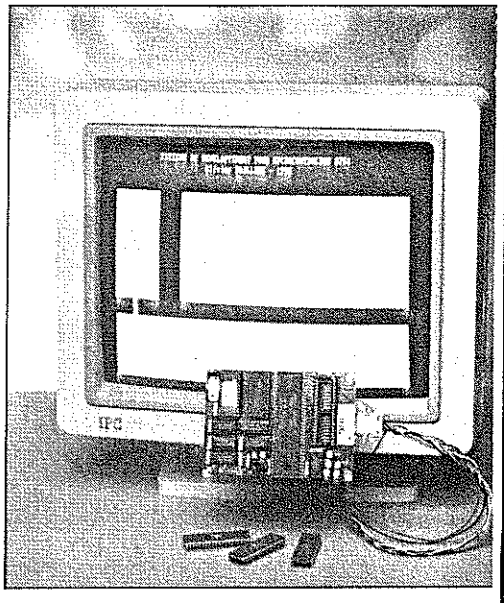
8751 EMULATOR

This article describes hardware and software that together form a powerful development system for the popular 8751 microcontroller from Intel.

Design by O. Bailleux



THE microcontroller emulator described here allows you to develop and debug 8751 application circuits with a minimum of effort. The emulator is has two main connections: one to the serial port of a PC, and another to the IC socket reserved for the 8751 in the target system. The combination of the PC, the software that runs on it, and the emulator allows you to

- download, modify, and upload 8751 programs;
- erase and program a 8751;
- put breakpoints in programs;
- display register and memory contents;
- run programs in single step mode;
- modify the contents of certain registers.

Clearly, this makes the emulator a powerful and indispensable tool for all of you who, at a certain stage, are 'confronted' with a 8751 application.

General remarks

This article does not aim at discussing or even introducing all the hardware and software aspects of the 8751 microcontroller, since this field is covered adequately by the *Microcontroller Handbook* from Intel. As regards practical programming of the 8751, the '8051/8032 assembler course' published in this magazine will be very useful to follow (the 8032, 8051 and 8751 are all devices from Intel's MCS52 family of microcontrollers).

None the less, the file READ.ME on the diskette supplied in relation to the present emulator contains some basic information on the 8751.

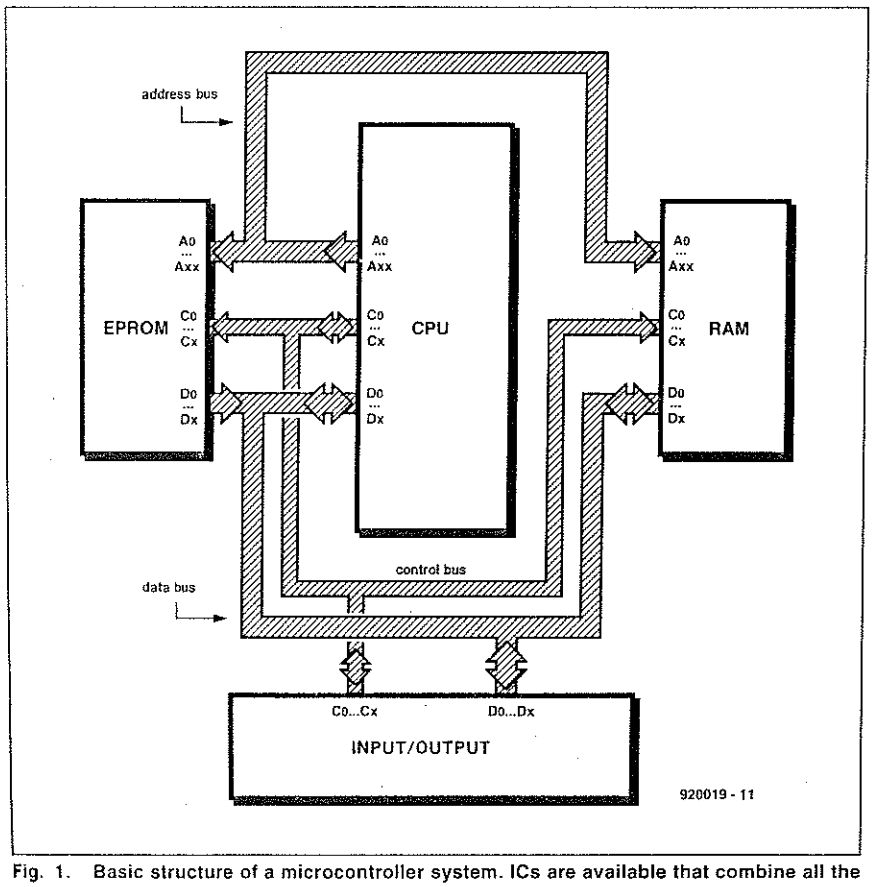


Fig. 1. Basic structure of a microcontroller system. ICs are available that combine all the functions shown here.

MAIN CHARACTERISTICS

- **Real-time 8751 emulator**
- **Clock frequency: 8 MHz**
- **All I/O ports available**
- **All internal interrupt sources available**
- **Powered by target circuit**
- **9,600 baud serial link to PC**
- **Breakpoint analysis and single-step mode**
- **Internal register and internal RAM contents displayed and available for editing**
- **Symbolic assembler for 8751**
- **Full-screen editor with error location facility**
- **Binary or Intel-hex output files**
- **Hard copy of formatted source program**

Limitations:

- **Monochip mode only (internal program memory, no external data memory)**
- **I/O bits P3.6 and P3.7 reserved for system**
- **Register 0 banks only**
- **System software uses 12 stack locations**
- **Masked interrupts during breakpoint processing**

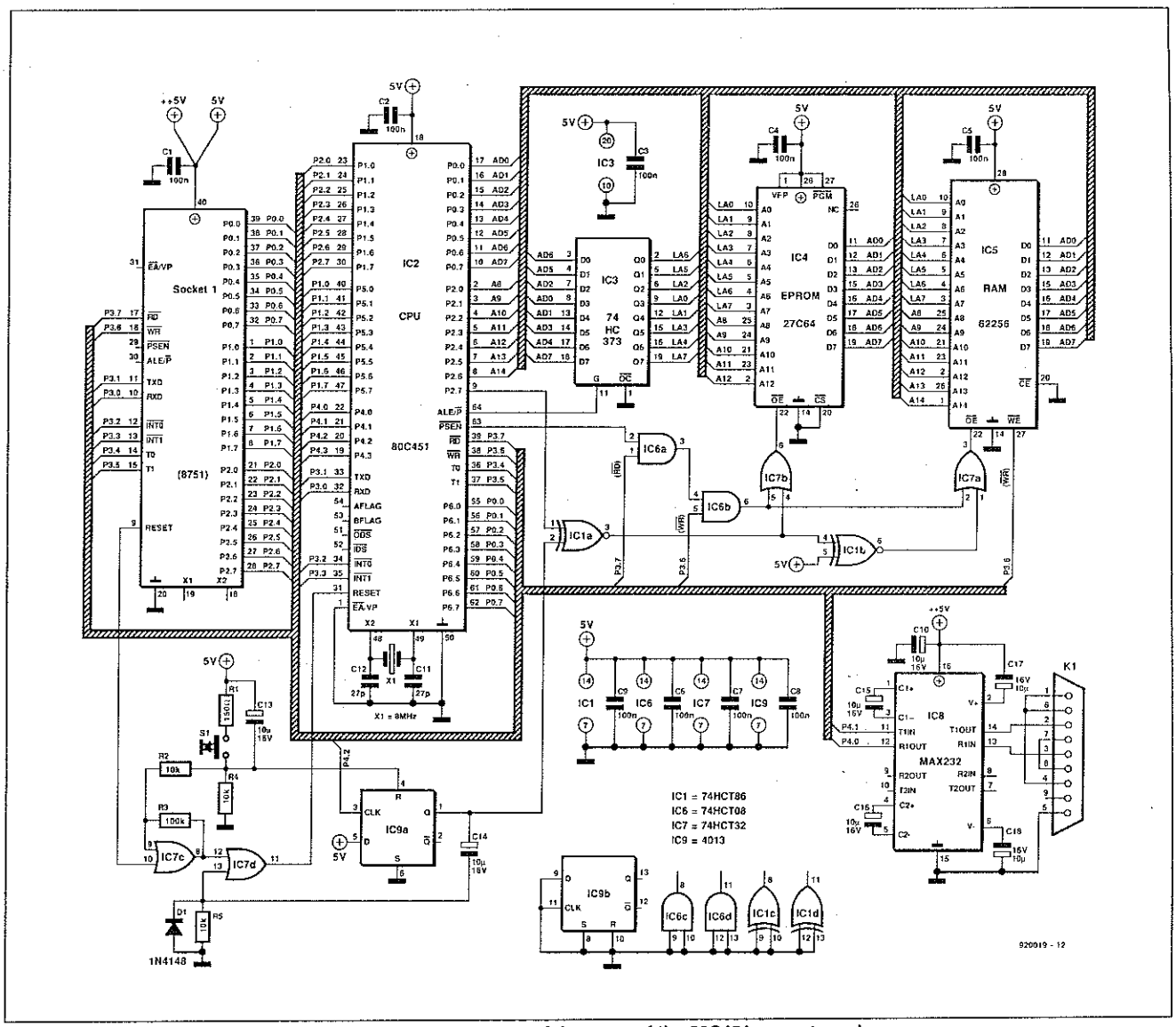


Fig. 2. Circuit diagram of the 8751 emulator. Note that some of the ports of the 80C451 are not used.

The 8751 is a microcontroller that integrates:

- an 8-bit microprocessor;
- a RAM of 128 bytes;
- 4 8-bit parallel I/O ports;
- 1 serial communication port;
- 2 programmable timers;
- 2 external interrupt lines.

The 8-bit (256-byte) addressable range of the 8751 is divided into two parts:

- user RAM between 00H and 7FH: this contains registers R0 to R7 (00H to 07H), and the system stack. A subrange of 16 addresses is bit-addressable.
- control register range between 80H and FFH: this range can be accessed by direct addressing only, and contains the I/O ports, the timer control registers, and a number of other special functions.

Principle of operation

As illustrated by Fig. 1, a microprocessor system generally consists of microprocessor, a program memory, a data memory, and input/output devices. The elements that form the system are interconnected by an address bus, a data bus and a control bus. Since the 8751 microcontroller works in 'monochip' mode, it has all of the above elements internally. Only the four 8-bit ports can be accessed by external devices. Emulating a 8751 thus calls for a device capable of:

- running object code from RAM instead of ROM—this is necessary to allow changes to the program to be made rapidly;
- providing four input/output ports as well as the basic hardware environment of the 8751;
- providing an instruction set that is compatible with that of the 8751.

The Type 80C451 meets all these requirements beautifully, and is used here to emulate a 8751. The pinning of this device is given in Fig. 3.

The emulator based on the 80C451 consists basically of:

- an EPROM with a start-up program;
- a 32-KByte RAM to hold the system routines and the user program being developed;
- a memory switching device with two states:
  start state: the EPROM is located at address 000H, and the RAM at address 8000H
  execute state: the RAM is at 0000H and the EPROM at 8000H;
- a TTL compatible (pseudo-RS232) serial interface for the connection the PC.

The initialization procedure consists of the following steps:

1. Initialization (reset), and start phase. EPROM-based program executed from address 0000H.

2. A routine in the start-up EPROM that sends the system program code from the PC to the emulator RAM. This is done via the serial link, and with the aid a simplified

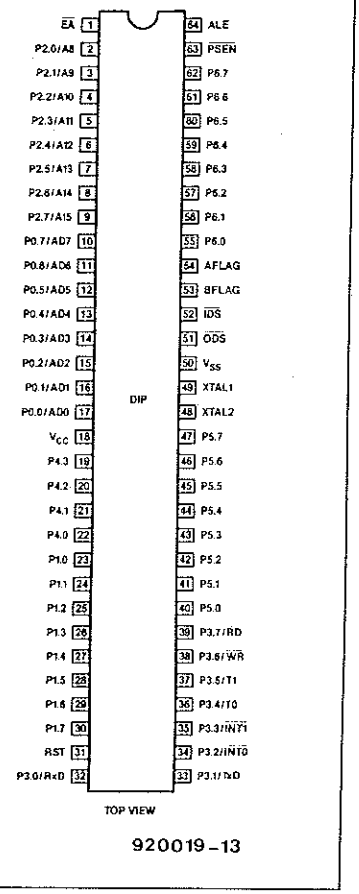


Fig. 3. Pinning of the 80C451 microcontroller (courtesy Signetics).

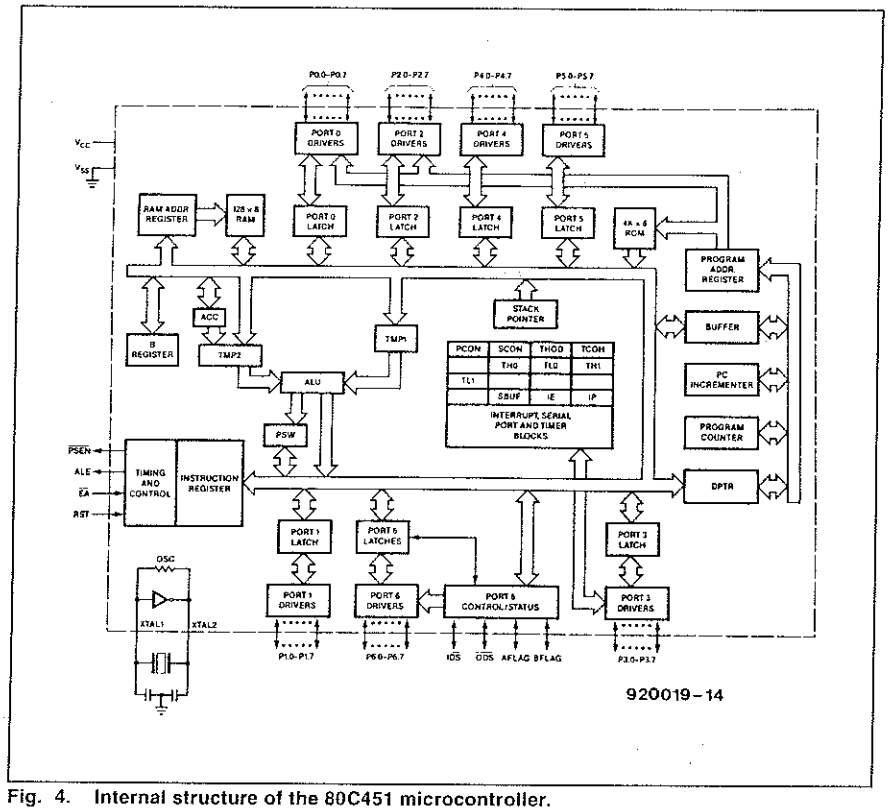


Fig. 4. Internal structure of the 80C451 microcontroller.

transfer protocol.

3. A routine in the start-up EPROM actuates a (reserved) I/O bit, which causes the system to switch to switch to the 'run' (program execution) stage, and in addition generates a reset.

4. The system program starts at address 0000H, it duplicates itself, and runs again from the upper part of the RAM at 7800H. From that location, it arranges the communication with the PC with the aid of a data transfer routine that allows the PC to

- issue a software reset to the microcontroller;
- read the emulator RAM;
- modify the emulator RAM;
- modify the stack pointer;
- upload a new user program;
- start or continue the execution of a user program.

5. The program that runs on the PC arranges the control of the emulator. Initially, it sends the user program object code to the emulator for the purpose of testing. This program will contain at least one breakpoint, which calls up a subroutine in the system software that stops the program execution.

6. The user program is executed until a breakpoint is encountered. At that location, the system software copies the current contents of the registers and the internal RAM of the 8751 into a buffer formed by the external RAM of the emulator, at 7000H. Next, the remote control subroutine is called.

7. The system software switches to the external emulator RAM, and causes the execution of the user program to continue, until the next breakpoint is encountered. At this stage, it is not possible to re-initialize the 8751 via the PC, or run a new user program.

8. The contents of the buffer starting at 7000H (which may have been modified in the mean time by the control program) are reloaded into the internal RAM and the registers, and the breakpoint routine is ended. The user program is continued up to the next breakpoint.

The key of the system is formed by the remote control routine, which enables the control software to modify the user program and the contents of the registers in the 8751. This function makes it possible to set breakpoints (temporarily), which are particularly useful in single-step mode.

Software

The software for the emulator is supplied on a 5¼-inch 360 KByte MSDOS formatted diskette, and consists of the following files:

DEV.EXE	programming environment
CONFIG.EMU	configuration file
SYS8751.C51	initialization file
SYS8751.A51	source code of initialization file
READ.ME	additional information

The user or the control program can access the emulator only while a static or dynamic breakpoint is being processed. The location of a breakpoint is determined by the user by inserting it in the source code. The control program calculates the corresponding address, saves the three bytes of opcode found there, and replaces them with an instruction that calls up the breakpoint handler. After a number of manipulations, the initial conditions are restored, and we can progress to the next breakpoint.

In single-step mode, the control program, while executing the breakpoint routine, automatically places a dynamic breakpoint after the next instruction. In single-step mode you can see exactly what happens to the register contents as the program evolves. A powerful debugging aid!

Hardware

At the heart of the emulator circuit (Fig. 2) is the Type SC80C451 microcontroller from Signetics, the North-American branch of Philips Components. Its internal structure is given in Fig. 4. By taking a close look at the block diagram, it becomes clear that the 80C451 is an extended version of the 80C51, with three additional I/O ports (which gives a total of six), and four additional I/O control lines.

The 80C451 has no internal ROM, and fetches its instructions from an external ROM or EPROM. The microcontroller offers 128 bytes of RAM, and the DIP version used here has no fewer than six 8-bit ports and one 4-bit port. The 80C451 is capable of address-

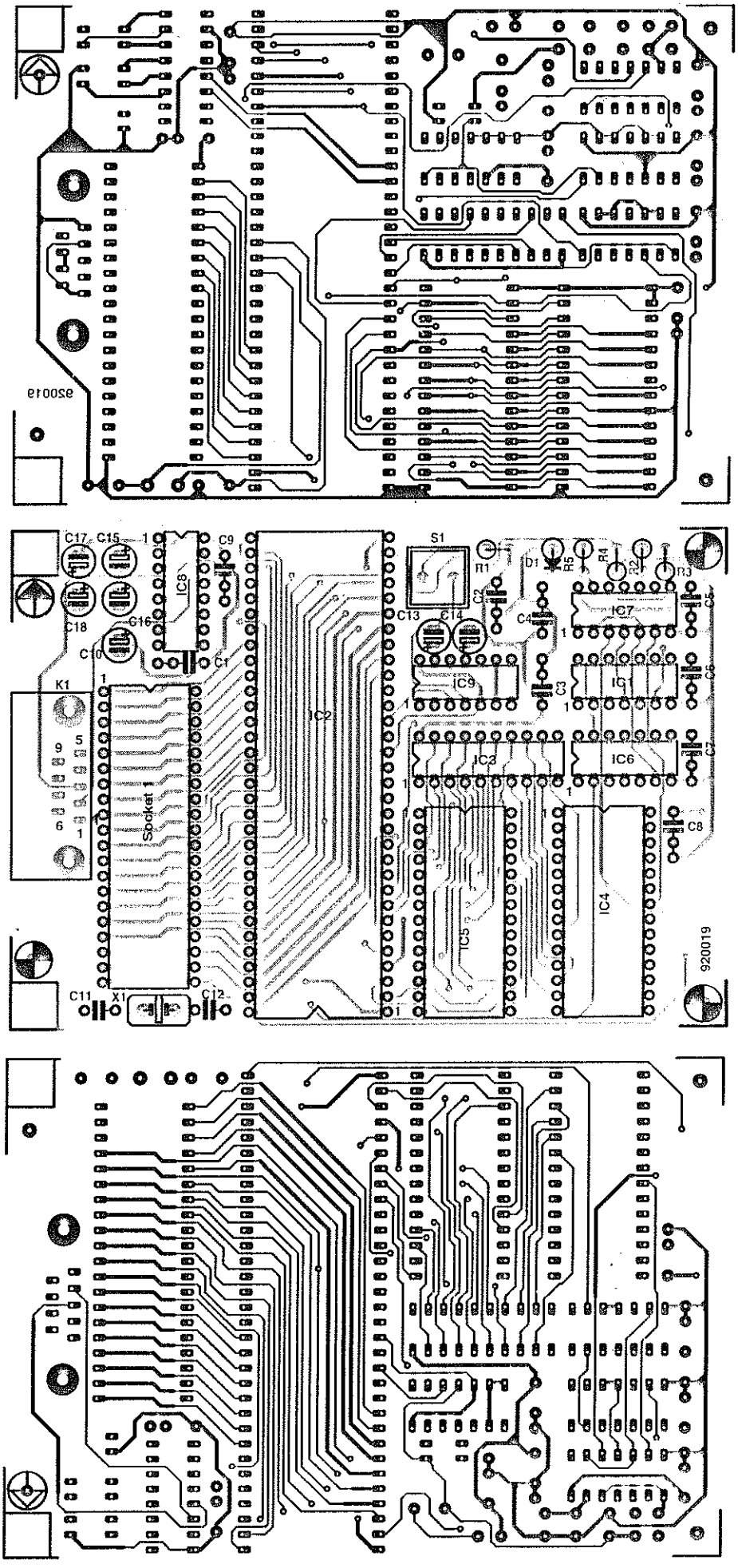


Fig. 5. Track layouts (component side and solder side) and component overlay of the double-sided, through-plated printed circuit board.

COMPONENTS LIST

Resistors:

1	150Ω	R1
3	10kΩ	R2;R4;R5
1	100kΩ	R3

Capacitors:

9	100nF	C1-C9
7	10µF 16V radial	C10;C13-C18
2	27pF	C11;C12

Semiconductors:

1	1N4148	D1
1	74HCT86	IC1
1	SC80C451CCN64 (Signetics)	IC2
1	74HC373	IC3
1	27C64 (ESS6051)	IC4
1	62256	IC5
1	74HCT08	IC6
1	74HCT32	IC7
1	MAX232 (Maxim)	IC8
1	4013	IC9

Miscellaneous:

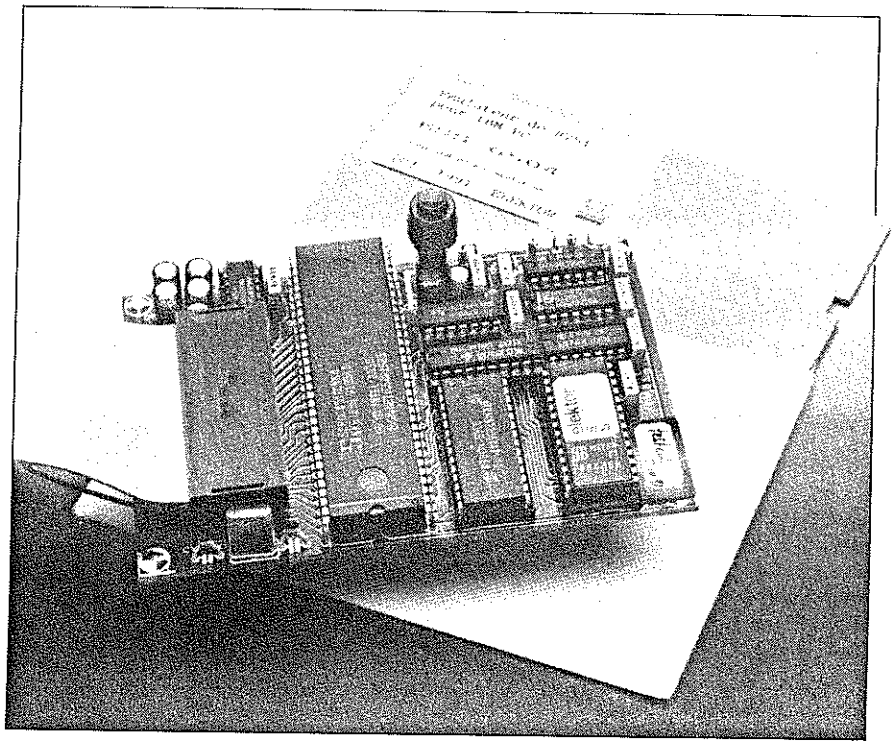
1	9-way female sub-D connector of PCB mounting	K1
1	push-button n.o.	S1
1	8MHz quartz crystal	X1
1	64-pin strip to make IC socket	
1	40-way IC socket with turned pins	H8
2	40-way IDC style DIP header	
1	length of 40-way flatcable	
1	enclosure; approx. dimensions: 92×146×28mm.	
1	printed circuit board	920019
1	control program on disk	ESS6051

ing 64 Kbytes of ROM, and an equal amount of RAM. The current consumption of the device is only about 24 mA at a supply voltage of 5 V and a clock frequency of 12 MHz, 3 mA in stand-by mode, and about 50 µA in the power-down ('sleep') mode.

The mask-programmable ROM version of the 80C451, the SC83C451, has 4 KBytes of ROM. Both the 80C451 and the SC83C451 have two 16-bit timers/counters. Their interrupt structure allows two priority levels to be implemented.

The presence of a serial I/O port in the 80C451 allows a UART (universal asynchronous receiver/transmitter) with true duplex operation to be realized quite easily. Alternatively, the serial I/O port may be used to extend the functions of the I/O lines, or to set up an inter-processor communication system.

The stand-by and the power-down modes can be entered via software. In stand-by mode, the CPU proper is halted, while the RAM, the timers, the serial port and the interrupt system continue to function. In the power-down mode, the clock oscillator is disabled, which causes all functions to be switched off, but the RAM contents to be retained.



Construction

The availability of a ready-made, through-plated printed circuit board (Fig. 5) makes the construction of the 8751 emulator relatively simple. A number of passive components are fitted upright. Due attention should be paid to the orientation of the ICs on the board, since they are not all mounted with the same direction. Naturally, the same goes for the orientation of the electrolytic capacitors and the diode.

The push-button, S1, may be fitted on to the enclosure, and is connected to the appropriate pins on the board.

The SC80C451 is best fitted into a socket made from two 32-pin strips. The RAM and the EPROM are also fitted in IC sockets.

SOCKET1 is best made from a good quality 40-pin IC socket. The connection to the 8751 socket in the target system is then readily made via a short length of flatcable fitted with a 40-way IDC-style DIP header at either end. Unfortunately, the pins of these DIP headers are pretty fragile, so take care not to break one, or the whole header is useless.

The completed printed circuit board is built into a suitable enclosure. A slot is cut in one of the sides to allow the 40-way flatcable to pass.

Practical use

Initially, the application (target) circuit and the PC are switched off. Note that the emulator is normally powered by the target circuit. Connect COM1: or COM2: of the PC to the emulator via an RS232 cable (the serial port selection is made in the file CONFIG.EMU as discussed below).

Do not connect the application circuit as yet. Connect a 5-V supply to the emulator. The + goes to pin 40 (+), and the – to pins 9 and pin 20 of the DIL socket on the emulator board. Next, run the program DEV.EXE on the PC, and check that the error message "Emulator not connected -- PRESS ANY KEY TO CONTINUE" does not appear.

At this stage, it should be possible to emulate the example program (see the syntax requirements mentioned in READ.ME), without the need of inserting the DIP plug into the 8751 socket in the target system. If this works, remove the temporary 5-V supply connections, and plug the 40-way DIP header into the socket on the emulator board. Insert the DIP header at the other end of the cable into the 8751 socket on your target system board. Power up the application before switching on the PC.

Programs may be edited and assembled even when the emulator is not connected or powered. The configuration file, CONFIG.EMU, contains only two characters. The first is an M (for the monochrome Hercules video adapter) or a C (for the colour video adapters CGA, EGA and VGA). The second character selects the serial port, and is either a 1 for COM1:, or a 2 for COM2:. If necessary, edit the configuration file using any ASCII compatible word processor. On the disk supplied through the Readers Services, the configuration is set to colour and COM1:.

The main program, DEV.EXE, is menu-driven, and uses the arrow keys to make selections. If you have the emulator powered up and connected to the target system at this stage, press switch S1 before running DEV.EXE.

Finally, READ.ME contains more information on the operation of the serial data link between the PC and the emulator. ■