MDS (Microprocessor Development System)

A microprocessor development system (MDS) is an essential tool which facilitates the design, evaluation and debugging of the user's microprocessor-based circuitry.

Hardware design, by its nature, belongs to the real world. The engineer can configure concrete objects; he/she can physically touch, arrange and in general `poke about' a hardware circuit. This makes testing a relatively attainable task, especially where only hardware is involved. Hardware-oriented breadboards are essentially arrays of conductors arranged to speed up the process of interconnection of the various elements making up the circuit. Normally the system is split up into relatively independent modules, which may be constructed and tested in a self-standing mode. These are then amalgamated gradually, with their interaction being closely monitored.

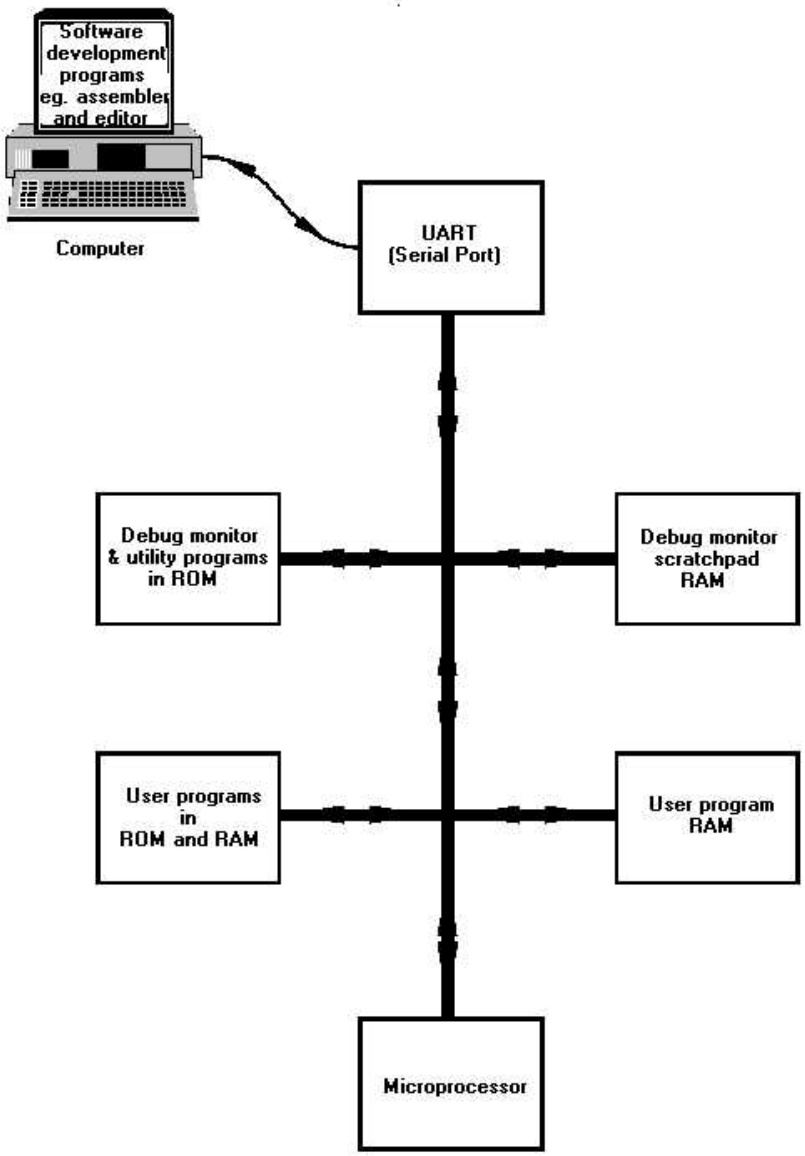


Fig. 1

Software engineering, on the other hand, is an abstract science. There is nothing tangible about a software circuit. At the physical level, software is a series of binary ones and zeros stored in various locations in ROMs and RAMs. In order to configure the software system, the designer must determine the sequence of 1s and 0s making up the program (the `software circuit'). A software breadboard then is a development tool which facilitates the `interconnection' of binary patterns in memory, and the running and testing of the interaction of these sequences with the computer hardware. A software breadboard is usually called a Microprocessor Development System (MDS).

As there is nothing physically to patch together, a software breadboard is of necessity somewhat more abstract than its hardware equivalent. One requirement, no matter how elementary a MDS, is a data terminal. This allows the designer to enter data (such as program instructions) into the system, and for the system to output information. In very simple systems, data may be input via switches on a hex keyboard (eg. the Heathkit ET3400 trainer) and output via 7-segment displays. More sophisticated systems use a printer, visual display unit (VDU) or a computing terminal (eg. a personal computer, PC).

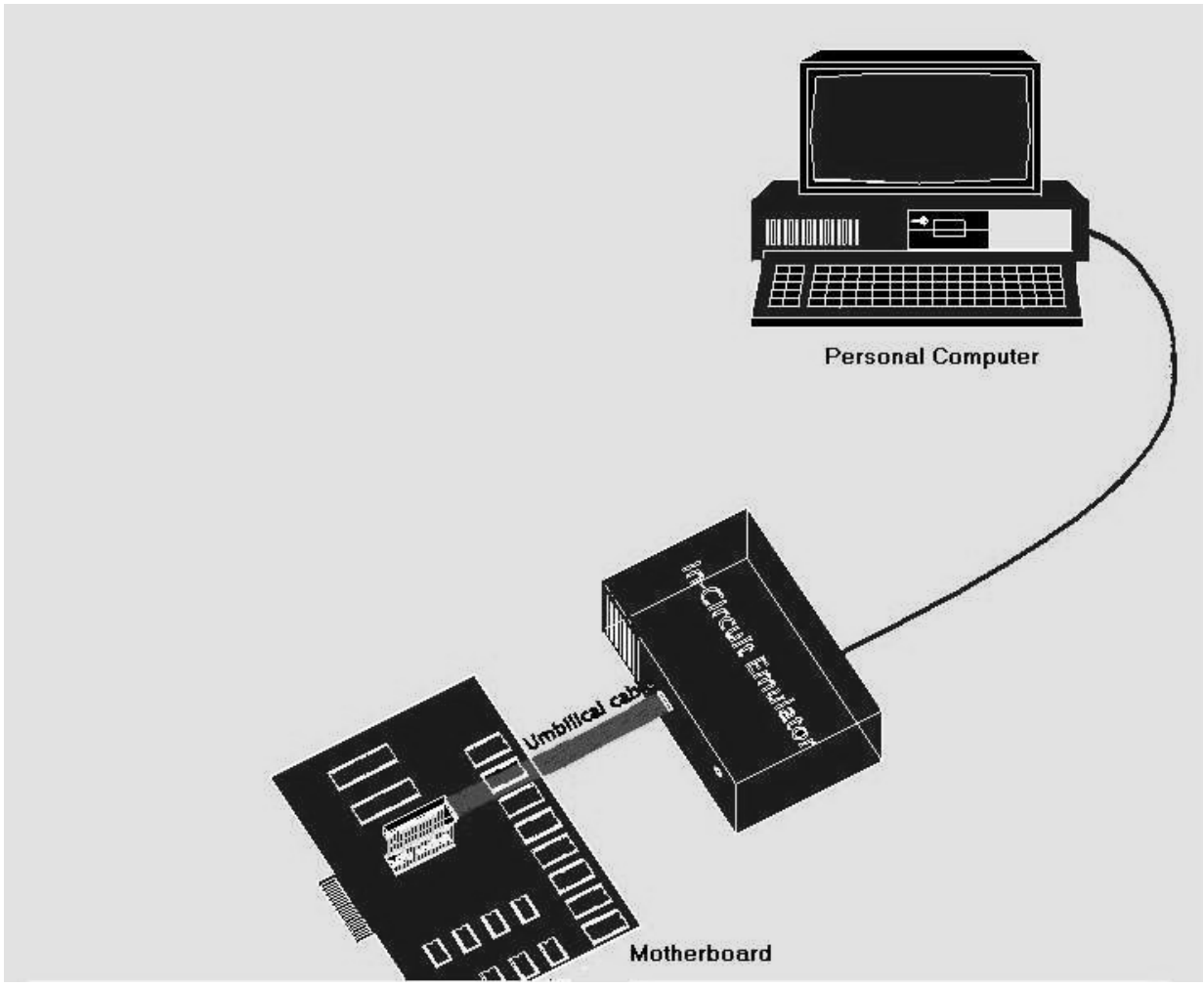
In order to handle this data flow, a MDS will include a central processor unit (CPU). This CPU is frequently the same processor as the hardware engineer is using, but not always. The CPU will of course be programmed to handle the tasks involved in the MDS. Dedicated MDSs will have at least a basic operating system programmed in ROM which is available on switch-on. Other system programs may be loaded in from disc (eg. assemblers, editors and compilers). The basic operating system is known as a **monitor**.

At the very simplest level, a monitor will allow the designer to examine the contents of given memory locations; change this data (if in RAM); run a program, save/load your user's program on disc, and use breakpoints to examine the progress of the program.

A skeleton of a basic development system is shown in Fig 1. The ROM-based monitor has an associated RAM in which it stores variables used by its routines. The monitor itself is usually a few Kbytes, whilst the scratchpad RAM is typically 1/4 Kbyte. The rest of the memory space is used by peripheral devices, other system programs and, of course, the RAM used to store the designer's program under development. RAM is used in preference to ROM, due to the ease of changing the code. However, bulk storage is necessary, as RAM is volatile, unless the user is prepared to type in the program each time the MDS is switched on.

Of course, for serious work, a more sophisticated MDS will be required. Typical enhancements for a dedicated unit are onboard EPROM programming and disc mass backup. These are normally plugged into a motherboard, forming the base of the MDS chassis. Where a PC is used as the terminal, as in this case, the computer will supply many of these facilities such as disc backup. One of the most intractable problems associated with MPU-based development, is monitoring the interaction of the hardware and software system components. To provide this facility, the MDS shown in Fig. 2 offers the technique of **In-Circuit Emulation** (**ICE**). As can be seen from the figure, the ICE enables the designer to remove the MPU in the target hardware, and replace it by the MDS itself. Thus the MDS pretends to the user's hardware that it is just an ordinary MPU. However, all the debugging facilities of the MDS are now available to monitor just what is happening. The ICE facility is very expensive, and is therefore usually only found on the more sophisticated MDS. On PC-based configurations ICEs are available as an outboard controlled via the serial port, and this is the configuration shown here.

Fig. 2


Personal Computer
In-Circuit Emulator
Umbilical cab
Motherboard