Proceedings of The Institute of Acoustics

Hardware and Software Interfacing Requirements

N.W. Heap

The Open University

Introduction

This paper reviews some of the basic hardware and software requirements essential to the task of interfacing microprocessors to experimental systems. Today it is possible to buy a microcomputer package capable of performing most tasks, but inevitably it appears one reaches the position where the basic package cannot meet all the requirements. At this stage additions and modifications must be made in order to complete the task in hand.

It is at the level of tailoring a system, to meet specific demands, that the user finds that the hardware and software are entwined and he cannot view the problem from one side or the other. Full exploitation of the inherent flexibility of a micro system can only be achieved when one has learnt to handle it at the lowest level. However, there is a delicate balance to be reached, otherwise one rapidly falls into the trap of becoming a microprocessor system designer, rather than an Acoustician with a problem to solve. A worthwhile piece of advice is to buy whenever possible and only build when the available products cannot fulfill your requirements.

Hardware Basics

The microprocessor manufacturers quickly realised that if people were ever to use their devices, they must provide the essential building blocks that enable the micro to communicate with the outside world. In order to do this the micro must be able to uniquely distinguish between the various peripherals. Two methods have been adopted,

(a) each device is given a unique code, (b) each device is treated as a memory location.

The micros are said to have accumulator I/O or memory mapped I/O respectively. From the hardware point of view the choice has little effect because the hardware building blocks have been designed appropriately. The major difference will be the software.

Once a device has been selected the data transfer can commence. Of course, each end of the link must be able to determine when the transfer starts and ends, so a group of handshaking signals are provided. A second such group is required between the interface block and the peripheral itself.

Data transfers can be handled in two ways. The micro can transmit the data and then interrogate the device until it receives some form of acknowledgement. This scheme, known as polling, is wasteful of microprocessor time, so an alternative is for the micro to send the data and continue program execution. When the device requires the intervention of the micro it interrupts the current program flow and causes it to locate a special subroutine to handle

the particular device.

Both schemes can establish a priority structure amongst the devices which ensures that fast peripherals are not hindered by the slow ones.

Hardware Interface Building Blocks

The basic building blocks that are available will handle parallel or serial data word transfers, either to the micro or directly into memory.

(i) PIA - Parallel Interface Adaptor
The early examples of this device were little more than 8 bit latches with a couple of handshake signals. Second generation devices allow the user to program the direction of the data transfer and will handle at least two peripherals. Each device contains several 8 bit registers, some for data and others for data direction and control signals. Before a transfer can occur the direction and control registers must be loaded with the appropriate codewords to ensure proper response to the handshake signals.

(ii) ACIA - Asynchronous Communications Interface Adapter
UART - Universal Asynchronous Receiver Transmitter
These two devices enable data transfer in a serial fashion, that is one bit at a time along a single pair of wires. Data is sent to the device in parallel form and the conversion to serial data is completed internally. On the peripheral side data arrives serially and is converted to parallel before transfer to the micro. This is the type of device that would be required to interface a terminal to a micro system. Although called asynchronous devices, it is important to realise that bits of each data word are transmitted synchronously, only complete words are asynchronous. To meet the various international standards the number of bits in each word is programmable, 5, 7 and 8 are all used, and an additional parity check bit may be appended.

The serial bit rate is set by an external clockand may extend to 40,000 bits/sec, although for most applications is unlikely to exceed 9600 bits/sec. Some devices may also provide a group of modern control signals which will allow the micro to communicate directly over the telephone network.

(iii) A/D Analogue to Digital Converter
D/A Digital to Analogue Converter
No laboratory micro system could be called complete without at least one A/D and D/A. The importance of this device is reflected in the wide choice available in both chip form and complete boards. Price will be determined by the speed and resolution required, and there are many 12 bit devices built for 8 bit micros. The higher resolution requires additional data transfers and so will reduce the ampling rate. Successive Approximation A/D's offer the fastest conversion, but single and dual slope integration types offer greater accuracy.

Multiplexors offer increased flexibility by enabling a single A/D to convert data from several signal sources. Of course one must not forget the sample and hold or the anti-aliasing filters.

(iv) DMA Direct Memory Access
Occasionally one encounters a situation where the necessary data transfer rates

cannot be met by any of the devices that operate under the control of the micro. In such cases one must resort to a technique known as direct memory access. The high speed peripheral interface effectively shuts off the micro and generates all the control and address information required by the system.

Most manufacturers produce an appropriate device called a DMA controller. The device must be told how many transfers are to occur and what part of memory is to be used. Aimed primarily at high speed block transfer peripherals such as disks, these devices have proved very successful in high speed data aquisition applications.

Software Basics

The creation and testing of computer programs consumes more money and time than the hardware equivalent, but when a hardware fault exists all work stops, whilst a program error only delays a single project and the cost gets lost under the general heading of 'program development'.

It is important to choose the language and level appropriate to the task in hand and so various software development tools have been produced to meet most demands.

(i) Machine code - the direct translation of instructions into binary code. This must be avoided at all cost because it consumes too much time and produces too many errors.

(ii) Assembler - almost essential when trying to control user designed interfaces, or interfaces not supported by the micro operating system. Each instruction/operation is assigned a unique mnemonic which is decoded by the assembler program to produce machine executable code.

(iii) High level - primarily aimed at user programs. Various languages are available each noted as being appropriate for a particular application.

Basic - Universally available and simple to learn, usually each program statement is interpreted at run time so execution is slow

Fortran - extremely powerful equation solving language offering comprehensive library of subroutines for all arithmetic operations

Pascal - structured language easy to read and maintain with strong variable type checking

C - structured language developed by Bell Laboratories can produce code almost as efficiently as an assembler programmer

Ada - new concept in languages, develops idea of software bus, offers real-time capabilities

Software Compromise

Although high level languages are desirable in as much as they allow rapid program development and easy maintenance, they cannot provide the software support necessary for user designed interfaces. Systems such as the PET and Apple do allow the user to communicate at the binary codeword level to devices. However, since the high level language they use is interpreted, the overall transfer rate is slow (around 50-100 bytes/sec). To achieve higher transfer

rates one must mix languages.

The simplest technique is to create an assembler subroutine to perform a specific task. This is then loaded into an area of memory unused by the high level program. The main program simply passes control to the subroutine as and when required. This method is equally valid for compiled or interpreted languages but the actual implementation will depend upon the specific language. A frequently encountered problem arises when one needs to measure a variable and use the value in a calculation. If one used Fortran for the main program, then arguments to be passed to the subroutine are supplied in the form of parameter lists. The calling sequence and methods for passing arguments must be followed exactly .If the subroutine is to function properly. The simplest way to see how a subroutine call is implemented is to examine the assembler code generated by the Fortran compiler.

The new languages such as Pascal and C normally allow user written assembler modules to be included quickly and simply.

Operating Systems

An operating system provides the human interface between the user and the micro system. Some are noted for their 'friendliness' and so enable users to minimise the learning period, others are not. Operating sytems for large mainframe computers are normally supplied by the manufacturer, but the micro has followed a different route. In this case hardware and software systems have been developed independantly, so one can buy the hardware and choose an operating system designed for the task in hand. This division of labour has also caused the creation of many independant software houses each offering various application packages. To simplify their task these software houses have split the load of developing operating systems and applications packages with each specialising in one area.

It is still possible to buy a complete software and hardware package, but one may then find that there are no application packages available. Two systems worthy of note are

CP/M Digital Research Corporation	system for 8080 and Z80 based 8 bit micros
Unix Bell Laboratories	originally developed for PDP machines, but finding wide acceptance for 16 bit micros.

Conclusion

Microprocessors and microcomputer systems have changed our way of life and have enabled us to undertake experiments and solve problems that would have been left a few years ago. But perhaps it is only now that computers are cheap enough to have in every laboratory or office, that we shall learn that with a little knowledge and common sense they are tools to do our bidding