1 Introduction to Embedded Systems

1.1 Introduction

Embedded computer systems are special versions of the computers we use everyday. The word "embedded" means part of another system. We will call this the host system. Indeed, much of the intelligence of today's devices owes much to the computers which are part of them. These range from devices like our remote controls, to washing machines, automobiles and aircraft. These may have several embedded systems in them. Embedded systems are typically smaller in size as compared to desktops. A quick review of microcomputer systems is in order.

1.2 Typical Microcomputer Organisation

A basic computer system consists of the Central Processing Unit (CPU), the memory unit and the input and output units. The units are interconnected with buses.

1.2.1 Central Processing Unit

The central processing unit is made up of three parts : the arithmetic/logic unit (ALU), registers and a control unit. The ALU performs arithmetic and logical operations on binary data while the registers are used for storing data, addresses or instructions. The exact number and type of registers depends on the particular computer system. The main register is called an accumulator which usually stores data to be manipulated by the ALU at the beginning of an arithmetic or logic operation. At the end of the operation, the accumulator usually stores the result.

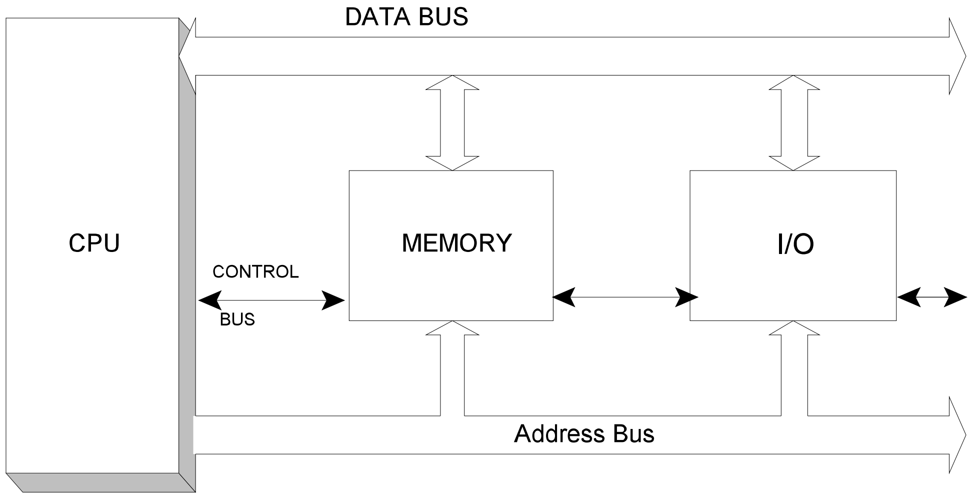


Figure 1 Block Diagram of a Basic Computer System

Control Unit
The control unit directs the operation of all other unit, by providing timing and control signals. This unit contains logic and timing circuits that generate the proper signals necessary to execute each instruction in a program. The control unit is also capable of responding to external signals; for example, interrupts. An interrupt request will cause the control circuit to temporarily interrupt the main program execution, jump to a special routine to service the interrupting device, then automatically return to the main program.

Memory Unit
The memory unit contains a large number of memory cells, usually RAM and ROM. They are used to store instruction sequences or programs which the computer will execute, the data that is to be processed by the programs, as well as data resulting from the processing. Operation of the memory is controlled by the control unit which signals for either a READ or a WRITE operation.

Input/Output Units
Input devices are used to supply the data needed for processing or to tell the program how to operate on data. Output devices allow results of the computer operations to be supplied to the user. The input/output devices are also known as peripherals. Common input devices are keyboards and mouse. Examples of output devices are printers and visual display units.

Input and output ports are simply the interface circuits which provide the appropriate electrical connections between the input/output devices and the rest of the computer system. Physically an input or output port is just a parallel set of buffers or D flip-flops which latch data when strobed by the CPU.

1.2.2 Bus Structure

The CPU is linked to other parts of the system by buses. A bus is a group of conductors that has connections to all of the system's main blocks. There are three major buses: the address bus, data bus and the control bus

Address Bus
The address bus is used by the CPU to send out the address of a memory location that is to be read from or written into; or the address of a particular input or output port. The address bus is unidirectional. Information generally moves in one direction only, originating in the CPU only.

Data Bus
The data bus is bidirectional because data can flow to or from the CPU, memory, or the I/O ports. Any device outputs connected onto the data bus must have three-state buffers, so that they can be floated except when that device is being selected. The number of conductors in the data

bus is determined by the number of binary digits the CPU can handle at a given time. However, the quantity of data stored at each location depends on the particular memory device. It may be a single bit, or 4, 8 or 16 bits per location.

Control Bus

The control bus is used to carry signals which synchronise the activities of the separate blocks of the computer system. It tells the memory and the I/O ports whether the CPU is reading or writing data. Two conductors found in the control bus are the READ and the WRITE lines. The READ line activated by the CPU when the CPU wants to retrieve data from the memory or input data from an input device. For example, to read a memory location the CPU places the address on the address bus and a READ signal is placed on the control bus. The memory then outputs the data from the addressed location to the data bus which is then transferred to the CPU data register. When the CPU is to store in memory or output data to an output device, it places the data on the data bus and then activates the WRITE line.

There are also some other control lines present in CPU to perform more complex operations.

1.3 History of microprocessors

Electronic computers were developed in the 1960's.They were physically large systems which occupied a room. In time, the size became smaller, until it was able to fit into an integrated circuit the size of a thumb. Intel Corporation designed the 4004 microprocessor in 1978, for a calculator. This was revolutionary, as previously computers were designed on separate large printed circuit boards and connected together.

The following lists the developments in terms of the various microprocessor from Intel. However, it is representative enough.

	8080	8085	8086	8088	80286	80386	80486	Pentium
Address	16	16	20	20	24	32	32	32
Data	8	8	16	8	16	32	32	32
Maths	N	N	N	N	N	Y	Y	Y
MHz	1	3	5	5	16	25	133	2000

As processors progress in terms of performance, older models get obsoleted. However, they find new life as processors in embedded systems. Thus we can expect embedded systems to become more powerful as time goes on.

A major challenge computer manufacturers face is the ever increasing demand for more computing power as users find more uses for computers. But users should not be expected to discard their old programs immediately to take advantage of the changes. Program development is a signifcant investment of effort.

1.3.1 The need for more computing power

As more uses are found for computing power, greater demand is found for its use. One area is the use of image and video data. An image can easily occupy one megabyte of memory. This requires a large addressing space. Also, manipulating these data require the use of fractional or floating point numbers, rounding off is not an option here.

Furthermore, fixed and floating point numbers need a large number of bits to represent data. The IEEE format specifies 80 bits. Many processors have incorporate hardware mathematical coprocessors to take advantage of this.

Sophisticated control algorithms like fuzzy logic, neural networks and statistical signal processing use a large amount of floating point numbers. Often they use matrix operations as well.

So, while embedded systems may control simple devices, the mathematical background required to do this *optimally* and *accurately* requires a lot of computing power, not just for games on desktops!

1.3.2 Increases in performance

This can come about in several ways:

Processing speed

Changing the processor speed brings about immediate benefits in terms of faster processing. The benefits are that older programs can be used. But this is not enough as computer programs deal with more complex data.

Address

Manipulating data like sound, images and video need large storage and high speed. These data need to be placed in memory for efficient processing. A 16 bit address only allows 2^{16} or 65536 bytes of data to be accessed. Today's processors use 32 addresses, with 64 bits are becoming available. It is a major challenge to modify programs to take care of this change.

Data

Most of the time, a unit of data manipulated is 8 bits or a byte. More sophisticated mathematical manipulations require many more bits to prevent loss of precision. Depending on whether one is using fixed or floating point, using 80 bits for mathematical data is common.

Also, instead of fetching 8 bits at a time, it is more efficient to fetch *and* manipulate them in units of 32 bits at a time, hence the reference to 32 bit data. Thus newer processors perform additions, etc on 32 bit data!

This can also cause problems in high level languages as the definition of an integer (the basic unit of data manipulation) will change when moving to a new processor.

Architecture

This refers to changing how data flows within the processor, making it more efficient. For example, having two or more processing units allow simultaneous fetching of data and actual processing of data.

Also we may pre-fetch data or code that is not required yet, anticipating when it is used and place it in memory that is faster so it executes faster.

The hardware may also be optimized so that certain commonly instructions can be executed much faster than others.

Architectural changes may introduce subtle errors in programs in they are time dependent. Otherwise, it should not change the running of the program significantly.

1.3.3 Progression of Intel microprocessors

The Intel series of microprocessors became popular because of the IBM PC which was introduced in 1982. It is remarkable that programs used then can still be run on today's desktop systems. This is not true for the 8080 and 8085 which are compatible with each other, being 8/16 data/address bit processors. As a historical note, the 8088 was designed to help software developers bring their programs over from the 8085. This was done using a segmented address scheme, allowing a 16 bit "logical" address within a 20 bit address space.The 20 bit address quickly became inadequate, prompting new address schemes involving 24 and 32 bits. As mentioned earlier, it was important that old programs could still be run.

In order to provide compatibility, Intel allowed the later processors to run in Real, Protected and virtual 86 modes to run programs written for earlier processors. See the chapter appendix for more information.

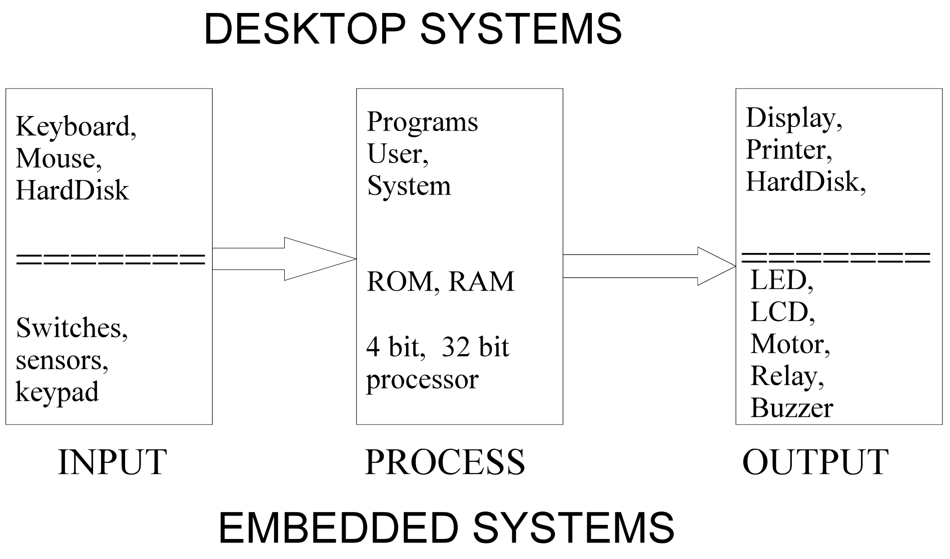
1.3.4 Advantages of the Intel PC architecture

While this section is not an advertisement, it serves to highlight what benefits processor hardware developments have on the industry.

The use of the PC over the decades have brought about many thousands of hardware and software applications. Not to mention the books and courses that build on this architecture. This translates into quick application development for sophisticated applications. Also, the processing power of the Intel processors continue to increase.

1.4 Embedded Systems Characteristics

It is useful to compare the characteristics of the commonly used desktops with embedded systems.


DESKTOP SYSTEMS
Keyboard, Mouse, HardDisk
Switches, sensors, keypad
Programs User, System
ROM, RAM
4 bit, 32 bit processor
Display, Printer, HardDisk,
LED, LCD, Motor, Relay, Buzzer
INPUT
PROCESS
OUTPUT
EMBEDDED SYSTEMS

As can be seen, the inputs to a desktop system differ from those of an embedded system. The processors however, may be similar. The amount of memory will differ, desktops being able to physically access more amounts of memory. Again, the outputs are different.

Fixed use

These systems are used for a fixed purpose. This is normally to control another device or process. Desktop computers can be used for a variety of purposes - games, word processing and so on.

Small size

The system must be relatively smaller than the system it is part of. This places constraints on the physical size of the embedded system.

Limited resources

This is related to the point about physical size. The amount of memory is limited and most of the time, this is in terms of RAM. Embedded systems mostly never use hard disks and programs are stored in ROM.

Failure tolerance
Because of the unpredictable nature of the physical environment, the systems should be capable of failing "gracefully" if it meets unpredictable circumstances. This is much preferable to the system "hanging".

Real time control
The speed of the embedded system is fast enough to affect the operation of the environment. For example, a remote control device operates in terms of seconds. But an aircraft control system works in terms of thousandths of a second. Of course the complexity of the calculations required for such a system plays a major part in determining what kind of system to use.

In many cases, especially involving human interactions which take place in fractions of a second, it does not make sense for a computer to work in millionths of a second here.

Guaranteed response time
Closely related to real time control, is the need for the embedded system to respond to a device or user request in a minimum time. This is especially true when systems are running several tasks at the same time. Important processes have stringent conditions on this.

Reliability
An unusually high emphasis is placed on this, especially when human lives are concerned. To complicate this is the fact that embedded systems often work in environmentally hostile environments. Aircraft are subjected to extremes of heat and cold and mechanical vibration and shock. Other factors are electromagnetic interference, radiation and chemical pollutants. Whatever environment the host system is operating in, will subject the embedded system to the same.

Being part of a larger system, the embedded system may be installed in physically inaccessible places. It may be very difficult to perform software updates and have to run continuously for years on end.

Simple input and output
Since embedded systems help control a larger system, it does not need elaborate input devices like a keyboard or a mouse or a display screen or a sound system for output. Most, it will read from switches, sensors, or small size keypads. It may output to motors, relays, buzzers, light emitted diodes, liquid crystal displays. Sometimes, a touch sensitive screen is used for input and output.

Power
Being part of a larger system, an embedded system should not consume too much power relative to the host system.

Low cost
This is needed, so it does not add appreciably to the total cost of the host system.

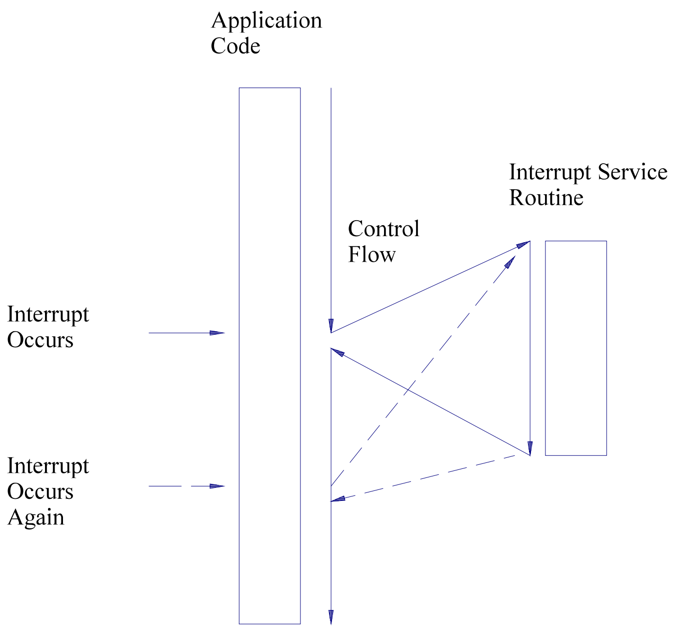
1.5 Some hardware features of embedded systems

Embedded systems frequently have to work with devices external to itself. For example, it has to receive input, from a keyboard, move an object and display the results of its processing to a display device like an LED display.
Mechancial control has to be in a precise manner. As mentioned above, the system often has to service several devices at the same time. Some of these devices have a large amount of data to be transferred in short time. Various hardware techniques have been built into the processor to help perform these tasks. These are: i) interrupts ii) hardware timers iii) direct memory access.

1.5.1 Interrupts

Embedded systems work at a high speed, in terms of millions of instructions per second, but often it interfaces to devices which operates much slower than it. Data input from humans for example, occur at millisecond rates, and the eye cannot detect events occurring at speeds faster than this. For the processor to wait for input and output is time consuming and inefficient. One way to improve this situation is through the use of interrupts.

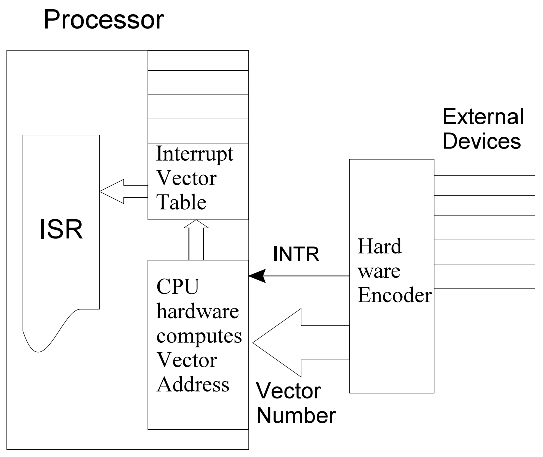
For efficient handling of several devices, such as in a real-time system, the software to handle the inputs and outputs is often handled by using interrupts. A characteristic of real-time control software is that it must read input changes when they occur and change outputs in a timely fashion, usually within a time limit. For example, a processor controlling the motion of a robot arm must stop it as it detects that the arm has reached the target position.



Most of the time, the processor will execute its main program. At any time, an external event may cause an interruption which causes the processor to suspend its current activity. The processor handles the interrupt by jumping to a special program called an interrupt service routine (ISR). It then resumes what it was doing. This differs from a subroutine call, because the locations where the routine resides have been preassigned.

Alternatively, if there is a critical section of a program which cannot have any interruptions. The main processor can set certain register flags to keep interrupts from occurring while it executes this code.

When an interrupt occurs, we have to determine which device activated it. The appropriate ISR is then chosen. A simple way is to let the processor poll each device using software. This is time consuming. So there are many schemes for efficiently determining the ISR. We briefly look at the vector technique.



1. An external device which needs to be serviced will pull the line of the encoder low.
2. The encoder will convert the line into a unique number, normally 8 bits in length. This is known as the vector number.
3. The encoder will also send an interrupt signal to the processor.
4. The internal hardware of the processor will calculate the address of the ISR.
5. It fetches the address from a preassigned area of memory known as the Interrupt Vector Table.
6 The processor will jump to that ISR and execute it.

This process is called "vectoring". The *addresses* of the ISRs (not the ISR itself) are known as vectors.

1.5.2 Timers

In order to control external devices in a precise manner, it is important to have a source of precise time events. While software delay loops may work and are simple, there are problems:

i) These normally involve loops with a high count. This does not allow the processor to perform any other tasks.

ii) In modern processors, the execution time of an instruction can vary, depending on various instruction pre-fetch schemes, like caching.

iii) A proper multitasking operating system cannot work on software loops as this makes task management very difficult.

Thus timers, external to the CPU, but which may be *on* or *off*-chip are essential. Timers may be used in many ways, but the most common is to:

i) Set a value to the timer
ii) Allow it to count down, and then cause an interrupt.
iii) The frequency of interrupts can be set so that a user can call up a delay using convenient units like milliseconds.

It is possible to directly read the values of the timer register for short duration events. As mentioned before, timer interrupts are the basis of multitasking operating systems.

1.5.3 Direct Memory Access

Direct Memory Access (DMA) is a method where peripheral devices transfer data to and from memory *without* data going through the processor. This is faster than using software. This is described in the chapter appendix.

Appendix

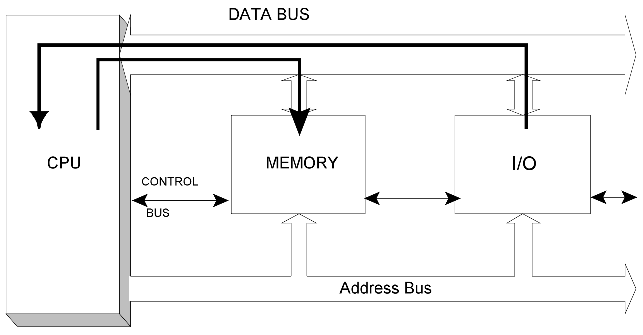
Real, Protected and virtual 86 modes

In Intel terminology, this involved starting processors in *real* mode, compatible with the 8088 to run old programs. If desired, they could switch the processor so it operated in *protected* or full 32/32 data address bits. In addition there was also a "virtual 86" mode which allowed *multiple* 8088 programs to run in protected mode.

DIRECT MEMORY ADDRESS

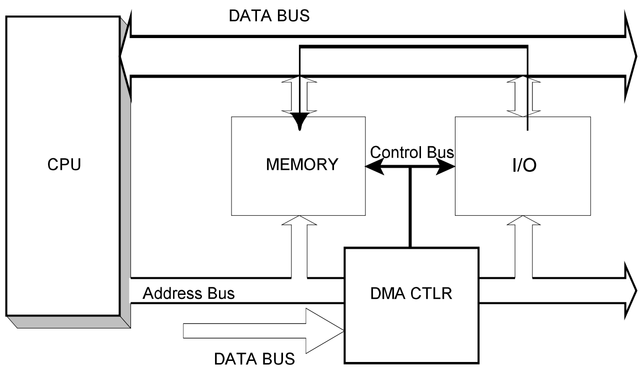
Software operations just to do a read and write can take many processor cycles. However, a DMA controller circuit generates the necessary address and control signals and can transfer data on every clock cycle. The main processor goes into a *HOLD* state where all its address and data lines are in high impedance.

Let us consider the case where a device is transferring data to memory, that is, we are doing a read. The case for writing to a device is the same, except for the direction of data transfer and of course, the control signals.



In this instance, the CPU reads from the I/O, and then writes to memory. All data passes through the CPU. Furthermore, there is also a software loop with a counter to keep track of the number of bytes transferred.So the instructions required are:

i) Read from I/O
ii) Write to memory
iii) Increment counter
iv) Check for end of loop
v) Loop



A DMA controller will generate the addresses needed and keep track of the bytes transferred, using hardware, which is much faster. The CPU is effectively inactive, its address and data bus is taken over by the DMA controller. Note that the DMA controller is treated as an I/O device as well and connected to the data bus for initialization and status purposes only.