1

Introduction to Real-time Systems

The purpose of this chapter is to:

- Provide a general introduction to computer control and to embedded computer systems.
- Define what is meant by real-time systems.
- Show how real-time systems can be classified.
- Illustrate the difficulties of writing real-time software.

You may not be familiar with some of the terms used in this chapter; you do not need to be concerned at this stage as they will be explained in detail in the later chapters.

1.1 HISTORICAL BACKGROUND

The earliest proposal to use a computer operating in *real time* as part of a control system was made in a paper by Brown and Campbell in 1950. The paper contains a diagram (see Figure 1.1) which shows a computer in both the feedback and feedforward loops. Brown and Campbell assumed that analog computing elements would be used but they did not exclude the use of digital computing elements. The first digital computers developed specifically for real-time control were for airborne operation, and in 1954 a Digitrac digital computer was successfully used to provide an automatic flight and weapons control system.

The application of digital computers to industrial control began in the late 1950s. The initiative came, not from the process and manufacturing industries, but from the computer and electronic systems manufacturers who were looking to extend their markets and to find outlets for equipment which had failed to be adopted by the military (Williams, 1977). The first industrial installation of a computer system was in September 1958 when the Louisiana Power and Light Company installed a Daystrom computer system for plant monitoring at their power station in Sterling, Louisiana. However, this was not a control system: the first industrial computer control installation was made by the Texaco Company who installed an RW-300 (Ramo-Wooldridge Company) system at their Port Arthur refinery in Texas. The refinery was run under *closed-loop control* on 15 March 1959 (Anon, 1959).

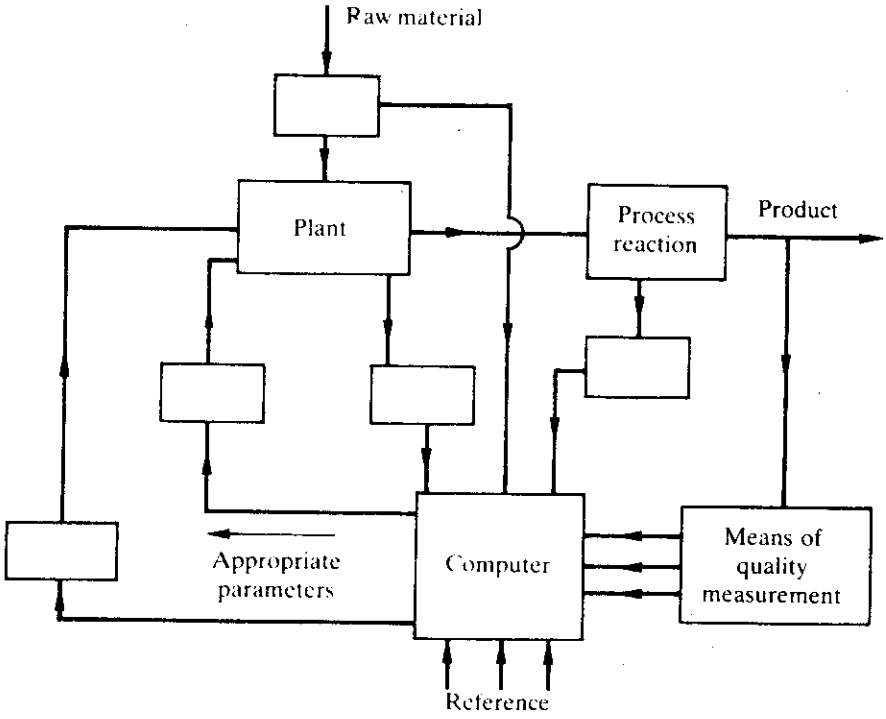

Figure 1.1 Computer used in control of plant (redrawn from Brown and Campbell, *Mechanical Engineering*, 72: 124 (1950)).

During 1957–8 the Monsanto Chemical Company, in co-operation with the Ramo-Wooldridge Company, studied the possibility of using computer control and in October 1958 decided to implement a scheme on the ammonia plant at Luling, Louisiana. Commissioning of this plant began on 20 January 1960 and closed-loop control was achieved on 4 April 1960 after an almost complete rewrite of the control algorithm part of the program. They also experienced considerable problems with *noise* on the measurement signals. This scheme, like the system installed by the B. F. Goodrich Company on their acrylanite plant at Calvert City, Kentucky, in 1959–60, and some 40 other systems based on the RW-300 were *supervisory control* systems used for *steady-state optimisation* calculations to determine the *set points* for standard analog controllers; that is, the computer did not control directly the movement of the valves or other *plant actuators*.

The first *direct digital control* (DDC) computer system was the Ferranti Argus 200 system installed in November 1962 at the ICI ammonia-soda plant at Fleetwood, Lancashire, UK, the planning for which had begun in 1959 (Burkitt, 1965). It was a large system with provision for 120 control loops and 256 measurements, of which 98 and 224 respectively were actually used on the Fleetwood system. In 1961 the

Monsanto Company also began a DDC project for a plant in Texas City and a *hierarchical* control scheme for the petrochemical complex at Chocolate Bayou.

The Ferranti Argus 200 used a ferrite core store to hold the control program rather than a rotating drum store as used by the RW-300 computer. The program was held in a programmable read-only memory (PROM). It was loaded by physically inserting pegs into a plug board, each peg representing one bit in the memory word. Although laborious to set up initially, the system proved to be very reliable in that destruction of the memory contents could only be brought about by the physical dislodgement of the pegs. In addition, security was enhanced by using special power supplies and switch-over mechanisms to protect information held in the main core store. This information was classified as follows:

Set points	Loss most undesirable
Valve demand	Presence after controlled stoppage allows computer to gain control of plant immediately and without disturbing the plant (referred to as *bumpless transfer*)
Memory calculations	Loss is tolerable, soon will be updated and only slight disturbance to plant
Future development	Extension to allow for optimisation may require information to be maintained for long periods of time

In addition to improved reliability the Argus system provided more rapid memory access than the drum stores of the RW-300 and similar machines and as such represented the beginning of the second phase of application of computers to real-time control.

The computers used in the early 1960s combined magnetic core memories and drum stores, the drums eventually giving way to hard disk drives. They included the General Electric 4000 series, IBM 1800, CDC 1700, Foxboro FOX 1 and 1A, the SDS and Xerox SIGMA series, Ferranti Argus series and Elliot Automation 900 series. The attempt to resolve some of the problems of the early machines led to an increase in the cost of systems: the increase was such that frequently their use could be justified only if both DDC and supervisory control were performed by the one computer. A consequence of this was the generation of further problems particularly in the development of the software. The programs for the early computers had been written by specialist programmers using machine code; this was manageable because the tasks were clearly defined and the quantity of code relatively small. In combining DDC and supervisory control, not only did the quantity of code for a given application increase, but the complexity of the programming also increased. The two tasks have very different time-scales and the DDC control programs have to be able to interrupt the supervisory control

programs. The increase in the size of the programs meant that not all the code could be stored in core memory: provision had to be made for the swapping of code between the drum memory and core.

The solution appeared to lie in the development of general purpose *real-time operating systems* and high-level languages. In the late 1960s real-time operating systems were developed and various PROCESS FORTRAN compilers made their appearance. The problems and the costs involved in attempting to do everything in one computer led users to retreat to smaller systems for which the newly developing minicomputer (DEC PDP-8, PDP-11, Data General Nova, Honeywell 316, etc.) was to prove ideally suited. The cost of the minicomputer was small enough to avoid the need to load a large number of tasks onto one machine: indeed by 1970 it was becoming possible to consider having two computers on the system, one simply acting as a stand-by in the event of failure. Throughout the 1970s the technological developments in integrated circuits and construction techniques for circuit boards led to an increase in reliability of computer systems, a large reduction in cost and major increases in processor power and in the amount of fast memory that could be provided. These changes began to force attention more and more on the problems of writing correct and dependable software. The advent of the microprocessor in 1974 led to a further reappraisal of approaches and made economically possible the use of *distributed computer control systems*. These developments are considered in more detail in Chapter 2.

1.2 ELEMENTS OF A COMPUTER CONTROL SYSTEM

As a simple example which illustrates the various operations of a computer control system, let us consider the hot-air blower shown in Figure 1.2. A centrifugal fan blows air over a heating element and into a tube. A thermistor bead is placed at the outlet end of the tube and forms one arm of a bridge circuit. The amplified output of the bridge circuit is available at B and provides a voltage, in the range 0 to 10 volts, proportional to temperature. The current supplied to the heating element can be varied by supplying a dc voltage in the range 0 to 10 volts to point A.

The position of the air-inlet cover to the fan is adjusted by means of a reversible motor. The motor operates at constant speed and is turned on or off by a logic signal applied to its controller; a second logic signal determines the direction of rotation. A potentiometer wiper is attached to the air-inlet cover and the voltage output is proportional to the position of the cover. Microswitches are used to detect when the cover is fully open and fully closed.

The operator is provided with a panel from which the control system can be switched from automatic to manual control. In manual mode the heat output and fan cover position can be adjusted using potentiometers. Switches are provided to operate the fan and heater. Panel lights indicate *fan on*, *heater on*, *cover fully open*, *cover fully closed* and *auto/manual* status. The desired output temperature (this is

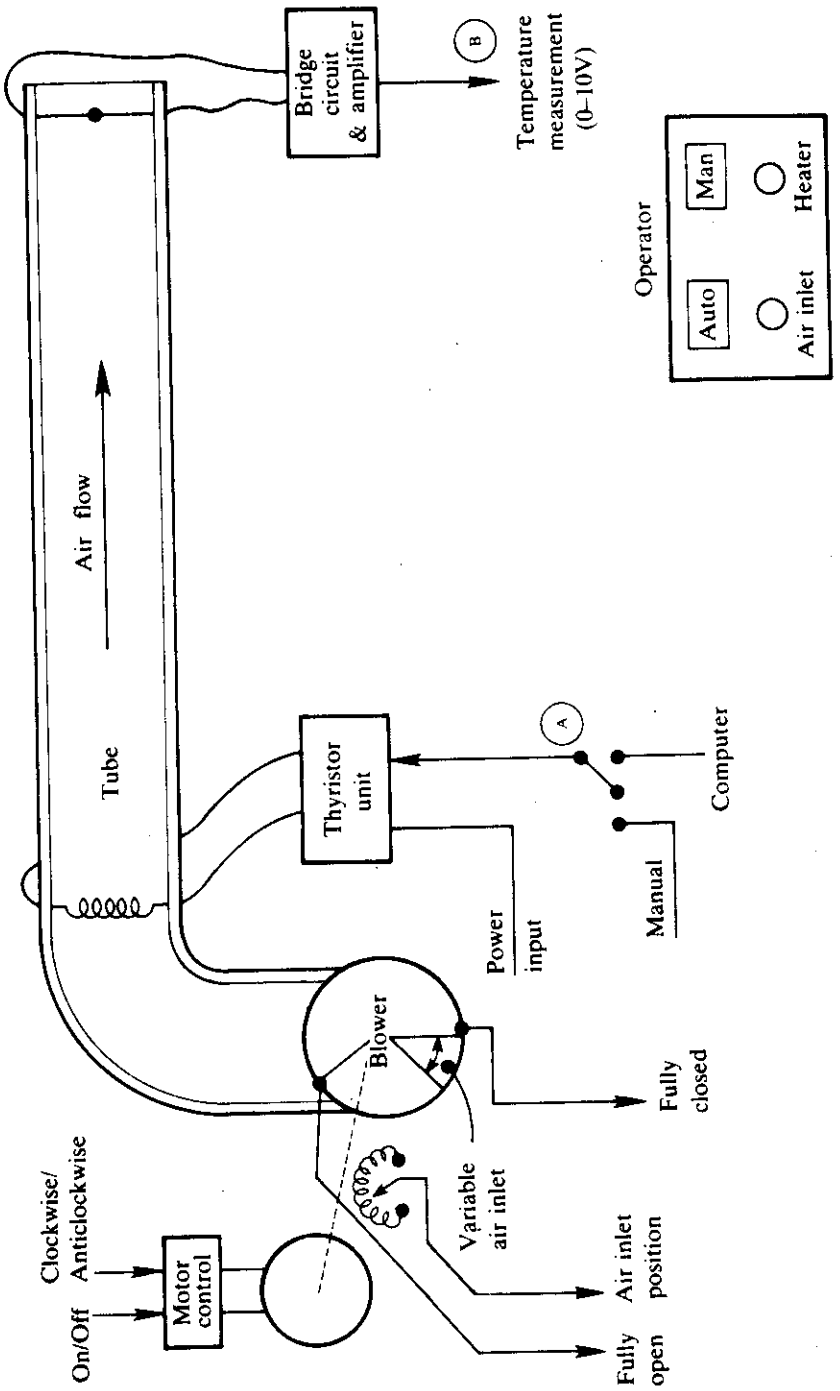

Figure 1.2 A simple plant – a hot-air blower.

known as the *set point* for the control system) is set by the operator using a slider potentiometer, the setting of which can be read by the computer. The operator can also adjust the fan cover position. The operation of this simple plant using a computer requires that software be provided to support *monitoring*, *control* and *actuation* of the plant. A general schematic of the system is shown in Figure 1.3.

Monitoring involves obtaining information about the current state of the plant.

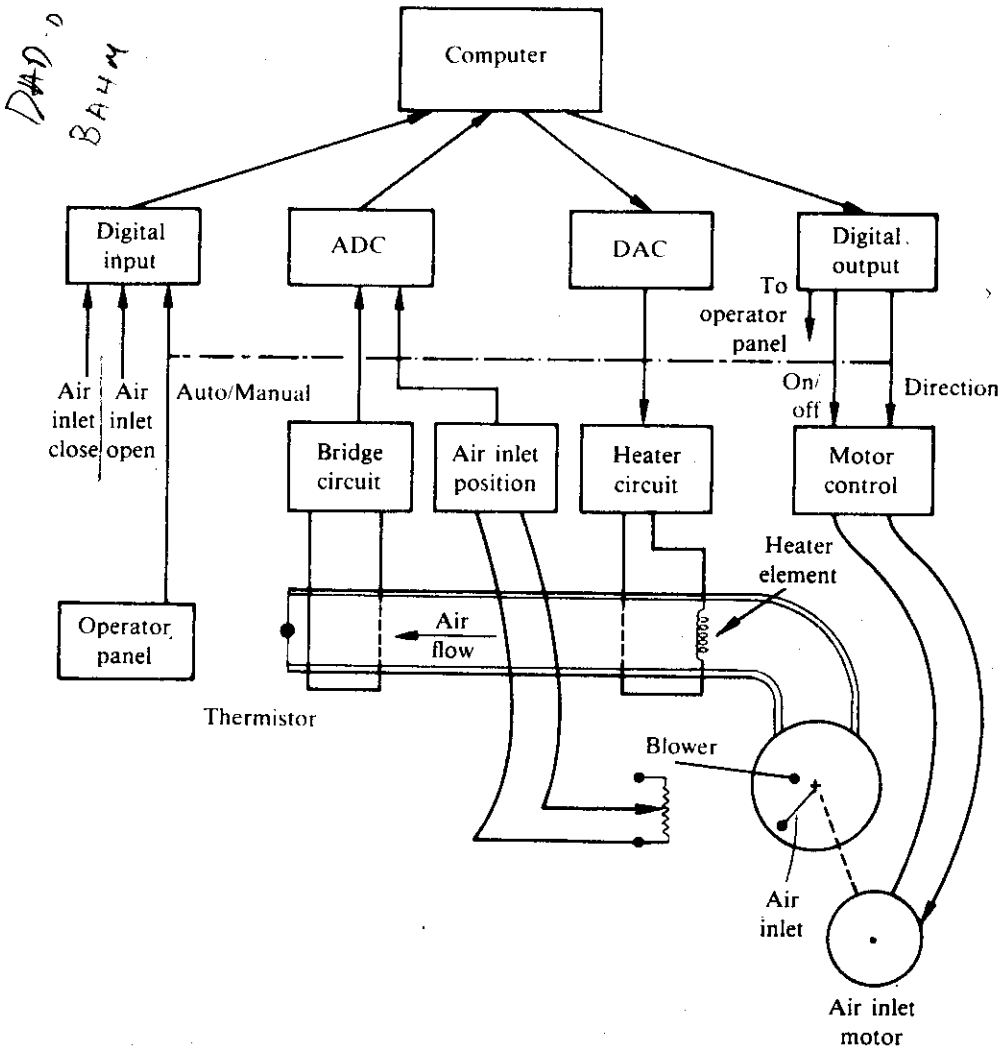

Figure 1.3 Computer control of a hot-air blower.

In the above example the information is available from the plant instruments in the following two forms:

Analog signals	Air temperature Fan-inlet cover position
Digital (logic) signals	Fan-inlet cover position (fully open, fully closed); status signals: auto/manual, fan motor on, heater on

Control involves the digital equivalent of continuous feedback control for the control of temperature (direct digital control, DDC) and for position control of the fan-inlet cover. Sequence and interlock control operations are also required since, for example, the heater should not be on if the fan is not running. The computer has also to handle automatic change-over from simply tracking (monitoring) the manual control operations to controlling the system when the operator requests a change from manual to automatic control. This change-over should be carried out

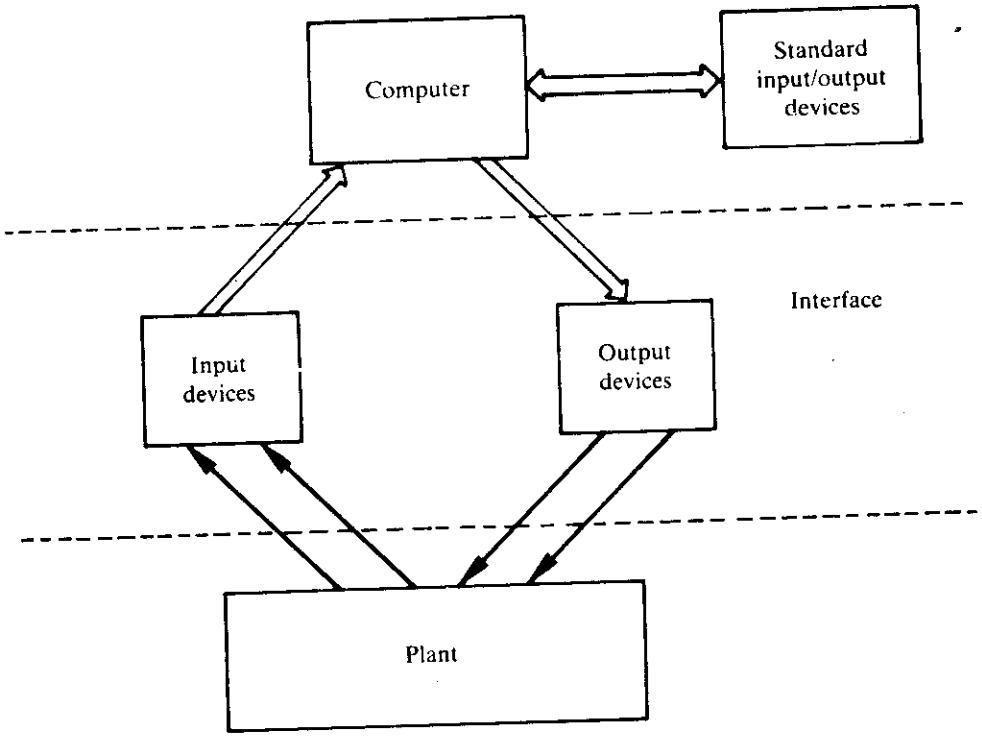

Figure 1.4 Generalised computer system.

without disturbing the temperature of the air at the output of the tube (a change-over which does not cause a plant disturbance is referred to as a *bumpless transfer*). These actions may involve parallel logic operations, time-sequential control and timing of operations.

Actuation requires the provision of a voltage proportional to the demanded heat output to drive the heater control; logic signals indicating on/off and the direction in which the fan-inlet cover is to be moved; and logic signals for the operator display.

The monitoring and actuation tasks thus involve a range of interface devices including analog-to-digital converters (ADCs), digital-to-analog converters (DACs), digital input and digital output lines and pulse generators. Details of the different types of interfaces are given in Chapter 3. For the present we will represent them simply as input devices and output devices as shown in Figure 1.4. Note that it is normal practice to use the terms 'input' and 'output' with reference to the computer;

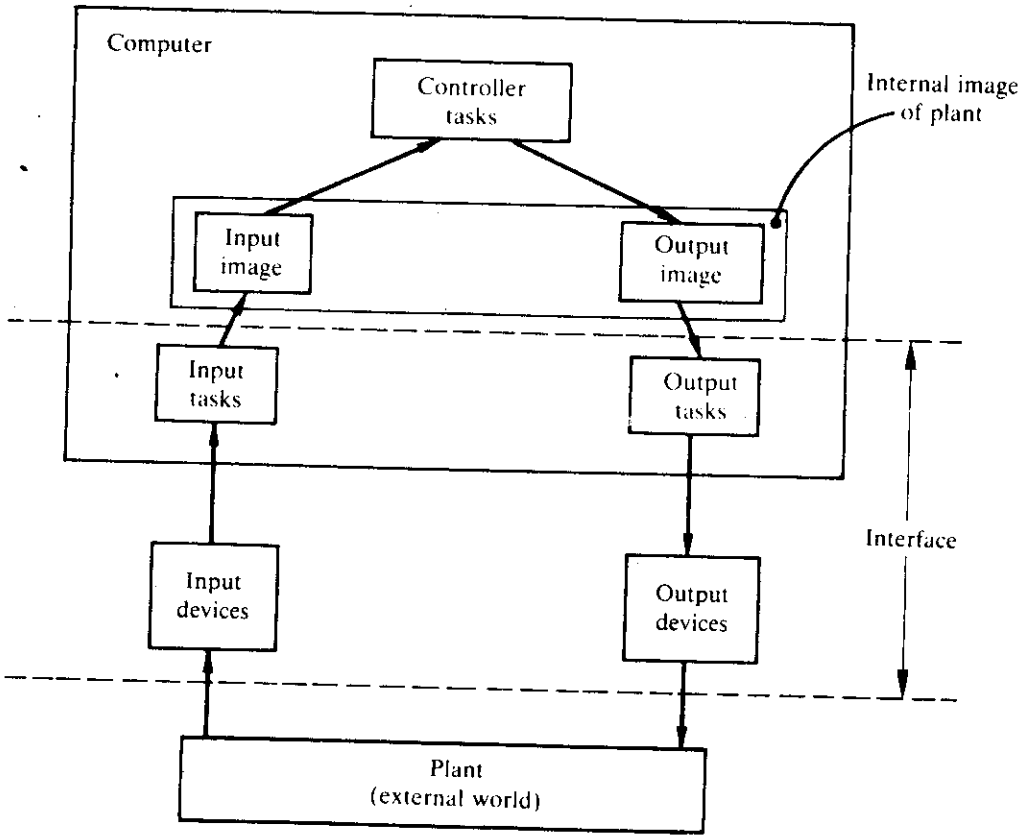

Figure 1.5 Generalised computer control system showing hardware and software interface.

thus input devices are devices which transmit information to the computer. Each of the various types of device will require software to operate it and we will represent this software as input tasks and output tasks. This generalised picture of a computer control system is shown in Figure 1.5. You should note that the interface boundary in Figure 1.5 is shown inside the computer block; this is to indicate that software (program code) is required to operate the interface devices.

The input devices plus the input software gather the information needed to create an *input image* of the plant. The input image is a snapshot of the status of the plant and this snapshot is renewed at specified intervals. (All of it may be renewed at the same time or some parts may be renewed less frequently than others.) Where the external information is in analog form the process of obtaining the snapshot will involve *digitisation* (conversion of a continuous measure into a discrete measure) as well as *sampling*.

The *output image* represents the current set of outputs generated by the control calculations. The output image will be updated periodically by the control tasks. It is the job of the output tasks to convey data contained in the output image to the plant. The control software within the computer can thus be considered as operating on an input image (or model) of the plant to produce the output image.

Figure 1.6 shows a simplified block diagram of the feedback control part of the system. In this diagram some of the symbols and terms that we shall use are defined. The diagram represents a *continuous* control system. The equivalent *sampled* (computer-controlled) version is shown in Figure 1.7.

In the simple computer control model we have described above we divided the software tasks to be performed into three major areas:

- plant input tasks;
- plant output tasks; and
- control tasks.

In doing so we have assumed that communication with the operator is treated as part of the plant input and plant output tasks. However, in many applications communication will extend beyond simple indicators and switches. Plant engineers, plant managers, pilots, air traffic controllers, drivers, and machine operators, for example, will require detailed information on all aspects of the operation of the

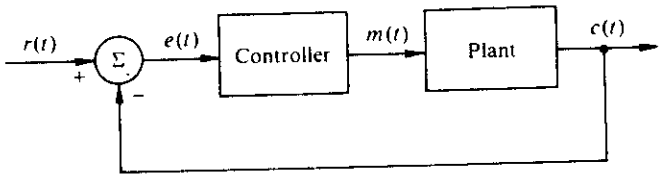

Figure 1.6 Simplified block diagram of continuous feedback control system: $r(t)$ = set point, $c(t)$ = controlled variable, $e(t) = r(t) - c(t)$ = error, and $m(t)$ = manipulated variable.

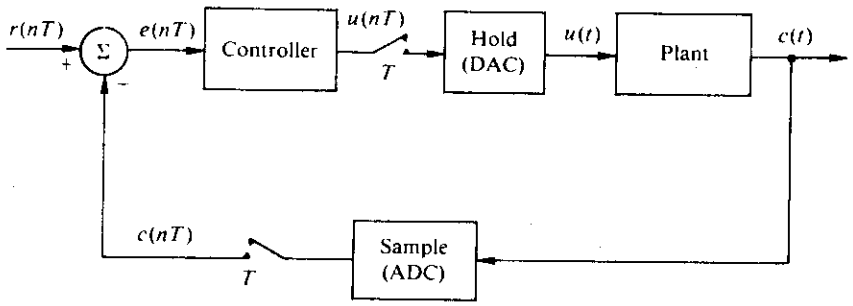

Figure 1.7 Simplified block diagram of a sampled feedback control system: $c(nT)$, $r(nT)$, $e(nT)$, $u(nT)$ are sampled values of $c(t)$, $r(t)$, $e(t)$, $u(t)$ at sample times nT where n is an integer and T is the sampling interval.

plant, aircraft, car, radar system, etc. Control of the system may be shared between several computers, not necessarily all on the same site, and hence information may have to be transmitted between computers. We must therefore extend the model to include communication tasks as shown in Figure 1.8. The communication tasks are assumed to cover input and output from keyboards/VDUs, remote transmission links, backing store (disks, etc.), output to printers, chart recorders, graph plotters, local-area networks (LANs) and wide-area networks (WANs).

The computer system may operate in a purely sequential way with the tasks – plant input, control, plant output and communication – being carried out in turn and with the sequence then being repeated indefinitely; or it may operate in a parallel manner, with the various tasks being carried out *concurrently*. In the case of a computer system with a single processing unit full parallel operation cannot be achieved and the system is said to operate *pseudo-concurrently*. The use of concurrency (or pseudo-concurrency) gives rise to problems relating to synchronising the various tasks and the sharing of resources between the tasks. Concurrency and its associated problems are covered in more detail in Chapters 5 and 6.

1.3 REAL-TIME SYSTEMS – DEFINITION

The title of this book includes the words *Real Time* and throughout the book we will use phrases such as real-time systems, real-time control: what do we mean by real time? The *Oxford Dictionary of Computing* offers the definition:

> Any system in which the time at which the output is produced is significant. This is usually because the input corresponds to some movement in the physical world, and the output has to relate to that same movement. The lag from input time to output time must be sufficiently small for acceptable timeliness.

This definition covers a very wide range of systems; for example, from workstations running under the UNIX operating system from which the user expects to receive

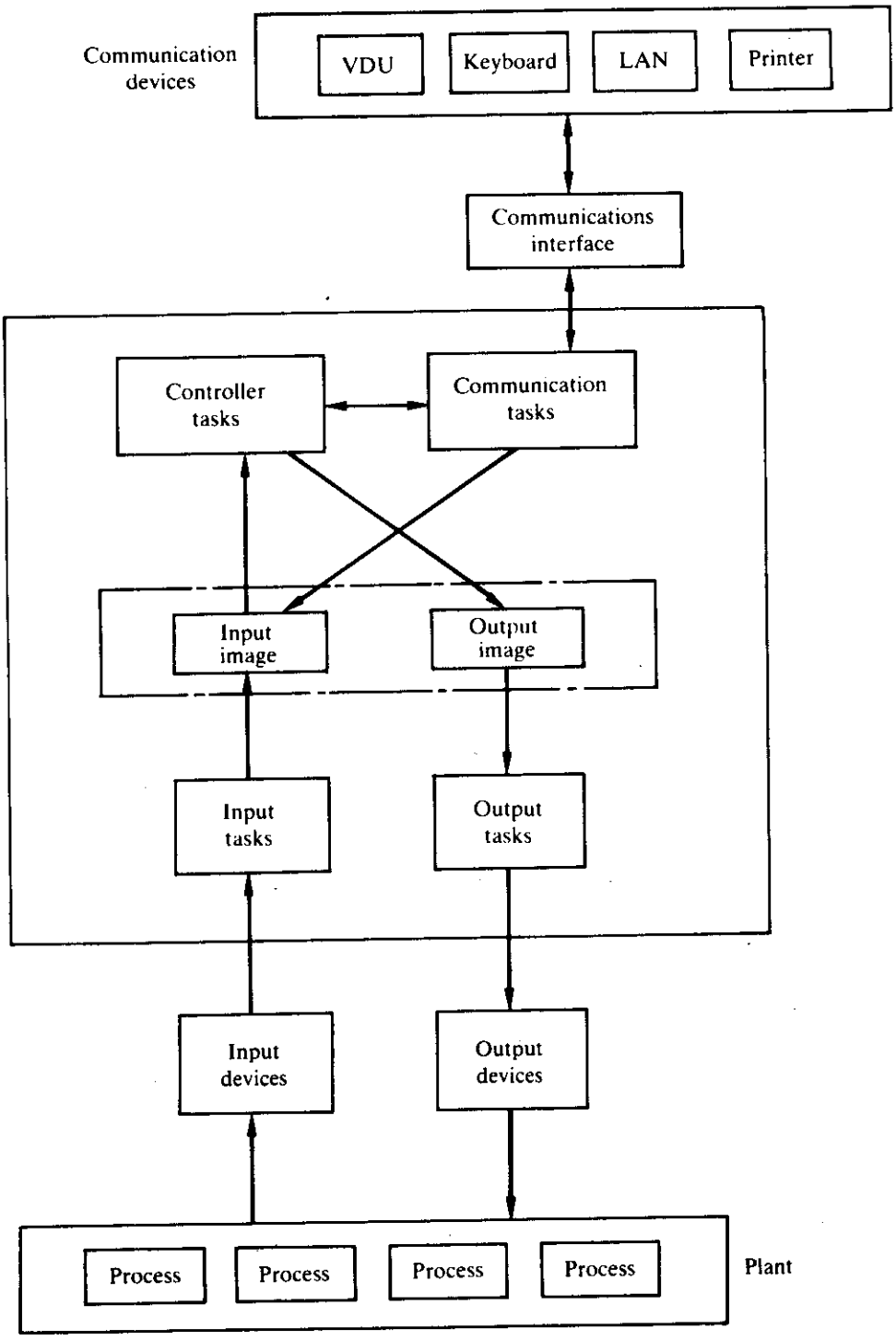

Figure 1.8 Computer control system showing communication tasks.

a response within a few seconds through to aircraft engine control systems which must respond within a specified time and failure to do so could cause the loss of control and possibly the loss of many lives. We are concerned with the latter type of system and Cooling (1991) offers the definition:

> Real-time systems are those which must produce correct responses within a definite time limit. Should computer responses exceed these time bounds then performance degradation and/or malfunction results.

An alternative definition is:

> A real-time system reads inputs from the plant and sends control signals to the plant at times determined by plant operational considerations – not at times limited by the capabilities of the computer system.

We can therefore define a real-time program as:

> A program for which the correctness of operation depends both on the logical results of the computation and the time at which the results are produced.

The computer systems that we shall be considering fall into a category known as *embedded* computers by which is meant that the computer is just one functional element of a real-time system and is not a stand-alone computing machine.

1.4 CLASSIFICATION OF REAL-TIME SYSTEMS

A common feature of real-time systems and embedded computers is that the computer is connected to the environment within which it is working by a wide range of interface devices and receives and sends a variety of stimuli. For example, the plant input, plant output, and communication tasks shown in Figure 1.8 have one feature in common – they are connected by physical devices to processes which are external to the computer. These external processes all operate in their own time-scales and the computer is said to operate in real time if actions carried out in the computer relate to the time-scales of the external processes.

Synchronisation between the external processes and the internal actions (tasks) carried out by the computer may be defined in terms of the passage of time, or the actual time of day, in which case the system is said to be *clock based* and the operations are carried out according to a time schedule. Or it may be defined in terms of events, for example the closure of a switch, in which case it is said to be *event based.*

There is a third category, *interactive*, in which the relationship between the actions in the computer and the system is much more loosely defined. Typically, the requirement is that a set of operations in the computer should be completed within a predetermined time. The majority of communication tasks fall into this category.

The control tasks, although not obviously and directly connected to the external environment, also need to operate in real time, since time is usually involved in

determining the parameters of the algorithms used (see Chapter 4). It is useful to divide tasks to be carried out by embedded computers into the above categories and you should learn to recognise the characteristics of each class.

1.4.1 Clock-based Tasks (cyclic, periodic)

A process plant operates in real time and thus we talk about the plant *time constants* (a time constant is a measure of the time taken by a plant to respond to a change in input or load and is used as a *characteristic* of the plant). Time constants may be measured in hours for some chemical processes or in milliseconds for an aircraft system. For feedback control the required *sampling rate* will be dependent on the time constant of the process to be controlled. The shorter the time constant of the process, the faster the required sampling rate. The computer which is used to control the plant must therefore be synchronised to real time and must be able to carry out all the required operations – measurement, control and actuation – within each sampling interval.

The completion of the operations within the specified time is dependent on the number of operations to be performed and the speed of the computer. Synchronisation is usually obtained by adding a clock to the computer system – normally referred to as a *real-time* clock – and using a signal from this clock to *interrupt* the operations of the computer at some predetermined fixed time interval.

The computer may carry out the plant input, plant output and control tasks in response to the clock interrupt or, if the clock interrupt has been set at a faster rate than the sampling rate, it may count each interrupt until it is time to run the tasks. In larger systems the tasks may be subdivided into groups for controlling different parts of the plant and these may need to run at different sampling rates. The clock interrupt is frequently used to keep a clock and calendar so that the computer system is aware of both the time and the date. Clock-based tasks are typically referred to as *cyclic* or *periodic* tasks where the terms can imply either that the task is to run once per time period T (or cycle time T), or is to run at exactly T unit intervals.

1.4.2 Event-based Tasks (aperiodic)

There are many systems where actions have to be performed not at particular times or time intervals but in response to some event. Typical examples are: turning off a pump or closing a valve when the level of a liquid in a tank reaches a predetermined value (consider what happens when the level of the fuel in a car petrol tank reaches the pump nozzle); or switching a motor off in response to the closure of a microswitch indicating that some desired position has been reached. Event-based systems are also used extensively to indicate alarm conditions and initiate alarm actions, for example as an indication of too high a temperature or too great a pressure. The specification of event-based systems usually includes a requirement that the system must respond within a given maximum time to a particular event.

Event-based systems normally employ interrupts to inform the computer system that action is required. Some small, simple systems may use *polling*; that is, the computer periodically asks (polls) the various sensors to see if action is required.

Events occur at non-deterministic intervals and event-based tasks are frequently referred to as *aperiodic* tasks. Such tasks may have deadlines expressed in terms of having start times or finish times (or even both). For example, a task may be required to start within 0.5 seconds of an event occurring, or alternatively it may have to produce an output (end time) within 0.5 seconds of the event.

1.4.3 Interactive Systems

Interactive systems probably represent the largest class of real-time systems and cover such systems as automatic bank tellers; reservation systems for hotels, airlines and car rental companies; computerised tills, etc. The real-time requirement is usually expressed in terms such as 'the average response time must not exceed...'. For example, an automatic bank teller system might require an average response time not exceeding 20 seconds. Superficially this type of system seems similar to the event-based system in that it apparently responds to a signal from the plant (in this case usually a person), but it is different because it responds at a time determined by the internal state of the computer and without any reference to the environment. An automatic bank teller does not know that you are about to miss a train, or that it is raining hard and you are getting wet: its response depends on how busy the communication lines and central computers are (and of course the state of your account).

Many interactive systems give the impression that they are clock based in that they are capable of displaying the date and time; they do indeed have a real-time clock which enables them to keep track of time. The test as to whether or not they are clock based as described above is to ask, 'Can the system be tightly synchronised to an external process?' If the answer is 'yes' they are clock based; if it is 'no' then they are interactive.

1.5 TIME CONSTRAINTS

It is now common practice to divide real-time systems (and real-time tasks) into two categories:

- *Hard real-time*: these are systems that must satisfy the deadlines on each and every occasion.
- *Soft real-time*: these are systems for which an occasional failure to meet a deadline does not comprise the correctness of the system.

A typical example of a hard real-time control system is the temperature control

loop of the hot-air blower system described above. In control terms, the temperature loop is a *sampled data* system which can be represented in block diagram form as shown in Figure 1.9. Design of a suitable control algorithm for this system involves the choice of the sampling interval T_s. If we assume that a suitable sampling interval is 10 ms, then at 10 ms intervals the input value must be read, the control calculation carried out and the output value calculated, and the output value sent to the heater drive.

As an example of hard time constraints associated with event-based tasks let us assume that the hot-air blower is being used to dry a component which will be damaged if exposed to temperatures greater than 50°C for more than 10 seconds. Allowing for the time taken for the air to travel from the heater to the outlet and the cooling time of the heater element – and for a margin of safety – the alarm response requirement may be, say, that overtemperature should be detected and the heater switched off within seven seconds of the overtemperature occurring. The general form of this type of constraint is that the computer must respond to the event within some specified maximum time.

An automatic bank teller provides an example of a system with a soft time constraint. A typical system is event initiated in that it is started by the customer placing their card in the machine. The time constraint on the machine responding will be specified in terms of an average response time of, say, 10 seconds, with the average being measured over a 24 hour period. (Note that if the system has been carefully specified there will also be a maximum time, say 30 seconds, within which the system should respond.) The actual response time will vary: if you have used such a system you will have learned that the response time obtained between 12 and 2 p.m. on a Friday is very different from that at 10 a.m. on a Sunday.

In practice the above categories are only guides: for example, in the hot-air blower an occasional missed sample would not seriously affect the performance; neither would a variation in sampling interval such that the inequality $9.95 \leqslant T_s \leqslant 10.05$ ms with a mean of $T_s = 10$ ms was satisfied. The system would not be satisfactory, however, if the sampling interval T_s was in the range $1 < T_s < 1000$ ms with a mean of 10 ms over a 24 hour period. Similarly, the automatic bank teller would not be satisfactory if at busy times customers had to wait 10 minutes, even if it achieved a mean response measured over a 24 hour period of 20 seconds. A typical specification might be a mean response measured over a 24 hour period of 15 seconds, with 95% of requests being satisfied within 30 seconds and no response time greater than 60 seconds.

A hard time constraint obviously represents a much more severe constraint on the performance of the system than a soft time constraint and such systems present a difficult challenge both to hardware and to software designers. Most real-time systems contain a mixture of activities that can be classified as clock based, event based, and interactive with both hard and soft time constraints (they will also contain activities which are not real time). A system designer will attempt to reduce the number of activities (tasks) that are subject to a hard time constraint.

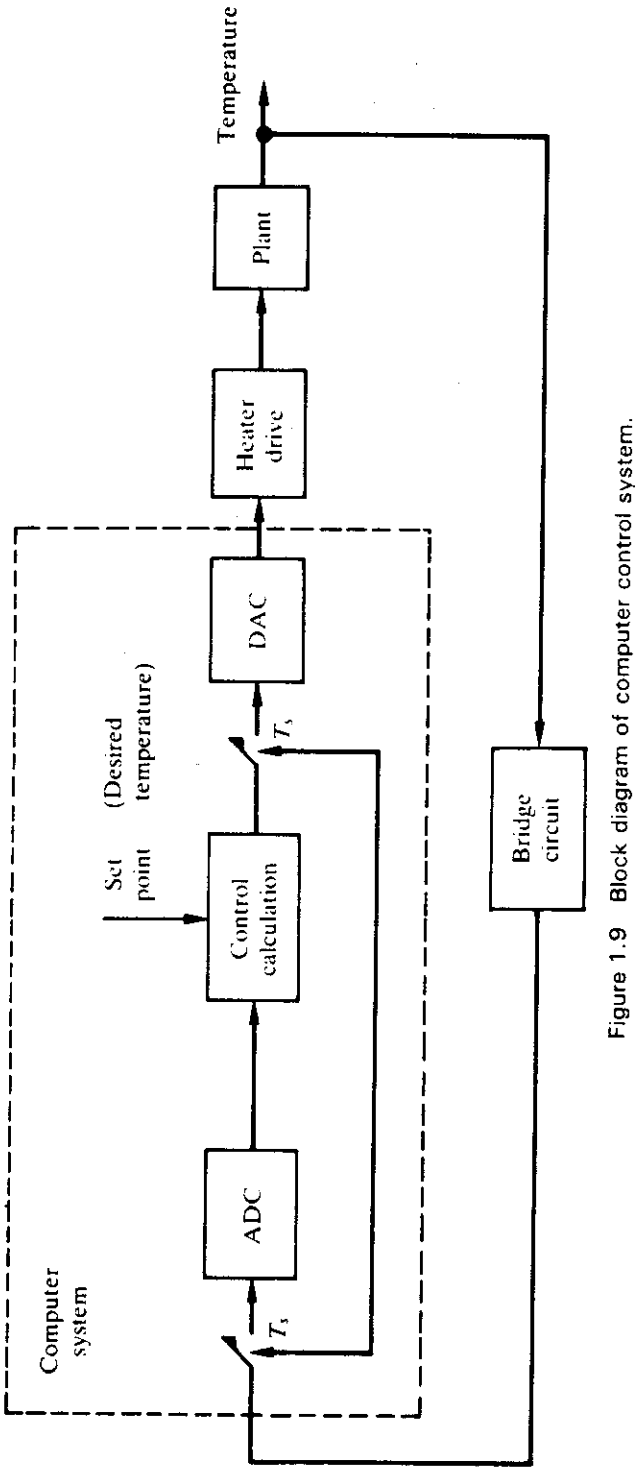

Figure 1.9 Block diagram of computer control system.

We can formally define the constraints as follows:

Hard		*Soft*	
Periodic (cyclic)	*Aperiodic (event)*	*Periodic (cyclic)*	*Aperiodic (event)*
$t_c(i) = t_s \pm a$	$t_e(i) \leqslant T_e$	$\frac{1}{n}\sum_{i=1}^{n} t_c(i) = t_s \pm a$	$\frac{1}{n}\sum_{i=1}^{n} t_e(i) \leqslant T_a$
		$n = T/t_s$	$n = T/t_s$

$t_c(i)$ the interval between the i and $i-1$ cycles,
$t_e(i)$ the response time to the ith occurrence of event e,
t_s the desired periodic (cyclic) interval,
T_e the maximum permitted response time to event e,
T_a the average permitted response time to event e measured over some time interval T,
n the number of occurrences of event e within the time interval T, or the number of cyclic repetitions during the time interval T,
a a small timing tolerance.

For some systems and tasks the timing constraints may be combined in some form or other, or relaxed in some way. For example, soft aperiodic (event) time constraints are often combined with a hard time constraint such that $t_c(i) \leqslant T_m$ where $T_m \gg T_a$. In some circumstances an occasional missed periodic hard time constraint might not be serious; for example, in a feedback control system missing the occasional sample will not result in a major disturbance of the controlled system (note: this is the case only if the manipulated variable is held constant at the previous value).

1.6 CLASSIFICATION OF PROGRAMS

The importance of separating the various activities carried out by computer control systems into real-time and non-real-time tasks, and in subdividing real-time tasks into the two different types, arises from the different levels of difficulty of designing and implementing the different types of computer program. Experimental studies have shown clearly that certain types of program, particularly those involving real-time and interface operations, are substantially more difficult to construct than, for instance, standard data processing programs (Shooman, 1983; Pressman, 1992). As we shall discuss later, the division of software into small, coherent *modules* is an important design technique and one of the guidelines for module division that we

shall introduce is to put activities with different types of time constraints into separate modules.

Theoretical work on mathematical techniques for proving the correctness of a program, and the development of formal specification languages, such as 'Z' and VDM, has clarified the understanding of differences between different types of program. Pyle (1979), drawing on the work of Wirth (1977), presented definitions identifying three types of programming:

- sequential;
- multi-tasking; and
- real-time.

The definitions are based on the kind of arguments which would have to be made in order to verify, that is to develop a formal proof of correctness for programs of each type.

1.6.1 Sequential

In classical sequential programming *actions* are strictly ordered as a time sequence: the behaviour of the program depends only on the effects of the individual *actions* and their *order*; the time taken to perform the action is not of consequence. Verification, therefore, requires two kinds of argument:

1. that a particular statement defines a stated action; and
2. that the various program structures produce a stated sequence of events.

1.6.2 Multi-tasking

A multi-task program differs from the classical sequential program in that the actions it is required to perform are not necessarily disjoint in time; it may be necessary for several actions to be performed in parallel. Note that the *sequential relationships* between the actions may still be important. Such a program may be built from a number of parts (processes or tasks are the names used for the parts) which are themselves partly sequential, but which are executed concurrently and which communicate through shared variables and synchronisation signals.

Verification requires the application of arguments for sequential programs with some additions. The task (processes) can be verified separately only if the constituent variables of each task (process) are distinct. If the variables are shared, the potential concurrency makes the effect of the program unpredictable (and hence not capable of verification) unless there is some further rule that governs the sequencing of the several actions of the tasks (processes). Note that the use of a

synchronising procedure means the time taken for each individual action is not relevant to the verification procedure. The task can proceed at any speed: the correctness depends on the actions of the synchronising procedure. (Synchronisation techniques are discussed extensively in Chapters 5 and 6.)

1.6.3 Real-time

A real-time program differs from the previous types in that, in addition to its actions not necessarily being disjoint in time, the sequence of some of its actions is not determined by the designer but by the environment – that is, by events occurring in the outside world which occur in real time and without reference to the internal operations of the computer. Such events cannot be made to conform to the intertask synchronisation rules. A real-time program can still be divided into a number of tasks but communication between the tasks cannot necessarily wait for a synchronisation signal: the environment task cannot be delayed. (Note that in process control applications the main environment task is usually that of keeping real time, that is a real-time clock task. It is this task which provides the timing for the scanning tasks which gather information from the outside world about the process.) In real-time programs, in contrast to the two previous types of program, the *actual time taken* by an action is an essential factor in the process of verification.

In the rest of the book we shall assume that we are concerned with real-time software and references to sequential and multi-tasking programs should be taken to imply that the program is real time. Non-real-time programs will be referred to as *standard* programs.

Consideration of the types of reasoning necessary for the verification of programs is important, not because we, as engineers, are seeking a method of formal proof, but because we are seeking to understand the factors which need to be considered when designing real-time software. Experience shows that the design of real-time software is significantly more difficult than the design of sequential software. The problems of real-time software design have not been helped by the fact that the early high-level languages were sequential in nature and they did not allow direct access to many of the detailed features of the computer hardware. As a consequence, real-time features had to be built into the operating system which was written in the assembly language of the machine by teams of specialist programmers. The cost of producing such operating systems was high and they had therefore to be general purpose so that they could be used in a wide range of applications in order to reduce the unit cost of producing them. These operating systems could be *tailored*, that is they could be reassembled to exclude or include certain features, for example to change the number of tasks which could be handled, or to change the number of input/output devices and types of device. Such changes could usually only be made by the supplier.

1.7 SUMMARY

In this chapter a brief history of computer control has been given; information on further development and the reasons for particular computer structures is given in Chapter 2. The simple example of a hot-air blower has been used to illustrate the breadth of knowledge required to design and implement a computer control system. The engineer is required to have knowledge of:

the plant;
transducers;
actuators;
computer hardware;
interface techniques;
communication systems;
software design;
programming languages;
control algorithm design; and
signal processing.

In this book we do not attempt to cover all these areas in detail, but instead concentrate on software design and the implementation of software.

As should have already been apparent, there will be an emphasis on dividing the operations to be performed into separate tasks. The detail of operations within the tasks will be hidden within the software for that task. This is a process known as *abstraction*. The organisation of the tasks to form an overall system can then be carried out without the distraction the detail of their operations would otherwise provide.

So far we have used the terms process, plant, program and task loosely and without defining them. In subsequent chapters they are used as follows:

Process, plant: The physical system that is to be controlled.
Program: The complete software package used to provide the control. Sometimes also referred to as the *application* program.
Module: A subdivision of the program.
Task: A subdivision of the program (or of a module of the program) whose execution can be separately scheduled from other parts of the program. A program or module may be divided into several tasks and because the tasks can be separately scheduled they have the potential to run concurrently.

The key ideas described in this chapter are:

- Computer control involves many different activities that have to be carried out concurrently.
- Digital computers are sequential devices and an individual computer can give, because of its speed of operation, the appearance of carrying out activities concurrently.

- Real-time systems have to carry out both periodic and aperiodic activities.
- Real-time systems have to satisfy time constraints that can be either a hard constraint or a soft (average value) constraint.
- Real-time software is more difficult to specify, design and construct than non-real-time software.

EXERCISES

1.1 You have been asked to design a computer-based system to control all the operations of a retail petrol (gasoline) station (control of pumps, cash receipts, sales figures, deliveries, etc.). What type of real-time system would you expect to use?

1.2 An automatic bank teller works by polling each teller in turn. Some tellers are located outside buildings and others inside. How could the polling system be organised to ensure that the waiting time at the outside locations was less than at the inside locations?

1.3 Would you classify any of the following systems as real-time? In each case give reasons for your answer and classify the real-time systems as hard or soft.

(a) A simulation program run by an engineer on a personal computer.
(b) An airline seat-reservation system with on-line terminals.
(c) A microprocessor-based automobile ignition and fuel injection system.
(d) A computer system used to obtain and record measurements of force and strain from a tensile strength testing machine.
(e) An aircraft autopilot.

1.4 For (a) the petrol (gas) station system in Exercise 1.1 and (b) the automatic bank teller in Exercise 1.2, list the activities which have to be performed and estimate the time requirements of each activity.

1.5 For the hot-air blower described in section 1.2, estimate the precision required for the analog-to-digital and digital-to-analog converters.

1.6 In the passage given below identify the different types of time constraints.

> The boiler pressure alarm lamp must be lit within 0.6 seconds of the boiler pressure high signal being set true. The temperature of the steam leaving the boiler must be read every 0.5 seconds. The operator display should be updated on average at 2-second intervals and on no occasion should the update interval exceed 10 seconds. The response to a request for a printout of the current plant status should typically be completed within 2 minutes.

1.7 Explain the difference between a real-time program and a non-real-time program. Why are real-time programs more difficult to verify than non-real-time programs?

2

Concepts of Computer Control

In this chapter we first use some typical process control applications to illustrate the range of operations that a computer-controlled system has to carry out. We then consider some of the differences between the process control applications and other embedded system applications.

In the final sections of the chapter we examine some of the computer configurations that are used for implementing computer control systems.

The aims of this chapter are to:

- Explain the terms used in process control applications and to provide examples of particular types of application.
- Describe in general terms the various types of control strategies that are used.
- Explain and compare basic computer configurations used for control.
- Explain the importance of the human–computer interface.

2.1 INTRODUCTION

Computers are now used in so many different ways that we could take up the whole book by simply describing various applications. However, when we examine the applications closely we find that there are many common features. The basic features of computer control systems are illustrated in the following sections using examples drawn from industrial process control. In this field applications are typically classified under the following headings:

- batch;
- continuous; and
- laboratory (or test).

The categories are not mutually exclusive: a particular process may involve activities which fall into more than one of the above categories; they are, however, useful for describing the general character of a particular process.

2.1.1 Batch

The term *batch* is used to describe processes in which a sequence of operations are carried out to produce a quantity of a product – the batch – and in which the sequence is then repeated to produce further batches. The specification of the product or the exact composition may be changed between the different runs. A typical example of batch production is rolling of sheet steel. An ingot is passed through the rolling mill to produce a particular gauge of steel; the next ingot may be either of a different composition or rolled to a different thickness and hence will require different settings of the rolling mill.

An important measure in batch production is *set-up* time (or *change-over* time), that is, the time taken to prepare the equipment for the next production batch. This is *wasted* time in that no output is being produced; the ratio between *operation* time (the time during which the product is being produced) and set-up time is important in determining a suitable batch size. In mechanical production the advent of the NC (Numerically Controlled) machine tool which can be set up in a much shorter time than the earlier automatic machine tool has led to a reduction in the size of batch considered to be economic.

2.1.2 Continuous

The term *continuous* is used for systems in which production is maintained for long periods of time without interruption, typically over several months or even years. An example of a continuous system is the catalytic cracking of oil in which the crude oil enters at one end and the various products – fractionates – are removed as the process continues. The ratio of the different fractions can be changed but this is done without halting the process. Continuous systems may produce batches, in that the product composition may be changed from time to time, but they are still classified as continuous since the change in composition is made without halting the production process. A problem which occurs in continuous processes is that during change-over from one specification to the next, the output of the plant is often not within the product tolerance and must be scrapped. Hence it is financially important that the change be made as quickly and smoothly as possible.

There is a trend to convert processes to continuous operation – or, if the whole process cannot be converted, part of the process. For example, in the baking industry bread dough is produced in batches but continuous ovens are frequently used to bake it whereby the loaves are placed on a conveyor which moves slowly through the oven. An important problem in mixed mode systems, that is systems in which batches are produced on a continuous basis, is the tracking of material through the process; it is obviously necessary to be able to identify a particular batch at all times.

2.1.3 Laboratory Systems

Laboratory-based systems are frequently of the operator-initiated type in that the computer is used to control some complex experimental test or some complex equipment used for routine testing. A typical example is the control and analysis of data from a vapour phase chromatograph. Another example is the testing of an audiometer, a device used to test hearing. The audiometer has to produce sound levels at different frequencies; it is complex in that the actual level produced is a function of frequency since the sensitivity of the human ear varies with frequency. Each audiometer has to be tested against a sound-level meter and a test certificate produced. This is done by using a sound-level meter connected to a computer and using the output from the computer to drive the audiometer through its frequency range. The results printed out from the test computer provide the test certificate.

As with attempts to classify systems as batch or continuous so it can be difficult at times to classify systems solely as laboratory. The production of steel using the electric arc furnace involves complex calculations to determine the appropriate mix of scrap, raw materials and alloying additives. As the melt progresses samples of the steel are taken and analysed using a spectrometer. Typically this instrument is connected to a computer which analyses the results and calculates the necessary adjustment to the additives. The computer used may well be the computer which is controlling the arc furnace itself.

In whatever way the application is classified the activities being carried out will include:

- data acquisition;
- sequence control;
- loop control (DDC);
- supervisory control;
- data analysis;
- data storage; and
- human–computer interfacing (HCI).

The objectives of using a computer to control the process will include:

- efficiency of operation;
- ease of operation;
- safety;
- improved products;
- reduction in waste;
- reduced environmental impact; and
- a reduction in direct labour.

2.1.4 General Embedded Systems

If we examine the general range of systems which use embedded computers – from domestic appliances, through hi-fi systems, automobile management systems,

intelligent instruments, active control of structures, to large flexible manufacturing systems and aircraft control systems – we will find that the activities that are carried out in the computer and the objectives of using a computer are similar to those listed above. The major differences will lie in the balance between the different activities, the time-scales involved, and the emphasis given to the various objectives.

2.2 SEQUENCE CONTROL

Although sequence control occurs in some part of most systems it often predominates in batch systems and hence a batch system is used to illustrate it. Batch systems are widely used in the food processing and chemical industries where the operations carried out frequently involve mixing raw materials, carrying out some process, and then discharging the product. A typical reactor vessel for this purpose is shown in Figure 2.1. A chemical is produced by the reaction of two other chemicals at a specified temperature. The chemicals are mixed together in a sealed vessel (the reactor) and the temperature of the reaction is controlled by feeding hot or cold water through the water jacket which surrounds the vessel. The water flow

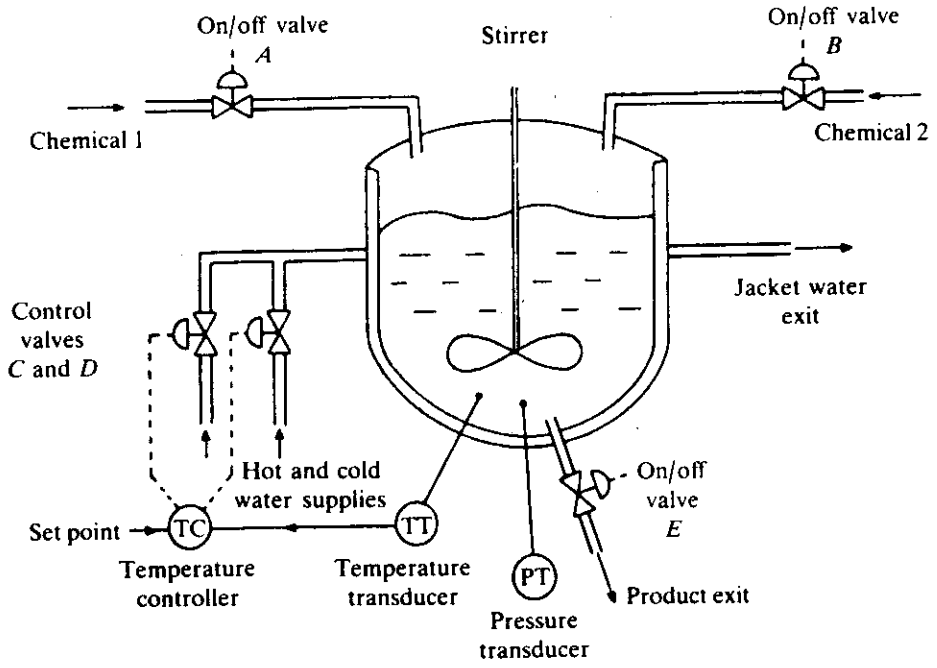

Figure 2.1 A simple chemical reactor vessel (redrawn from Bennett and Linkens, *Real-time Computer Control*, Peter Peregrinus (1984)).

is controlled by adjusting valves C and D. The flow of material into and out of the vessel is regulated by the valves A, B and E. The temperature of the contents of the vessel and the pressure in the vessel are monitored.

The procedure for the operation of the system may be as follows:

1. Open valve A to charge the vessel with chemical 1.
2. Check the level of the chemical in the vessel (by monitoring the pressure in the vessel); when the correct amount of chemical has been admitted, close valve A.
3. Start the stirrer to mix the chemicals together.
4. Repeat stages 1 and 2 with valve B in order to admit the second chemical.
5. Switch on the three-term controller and supply a set point so that the chemical mix is heated up to the required reaction temperature.
6. Monitor the reaction temperature; when it reaches the set point, start a timer to time the duration of the reaction.
7. When the timer indicates that the reaction is complete, switch off the controller and open valve C to cool down the reactor contents. Switch off the stirrer.
8. Monitor the temperature; when the contents have cooled, open valve E to remove the product from the reactor.

When implemented by computer all of the above actions and timings would be based upon software. For a large chemical plant such sequences can become very lengthy and intricate and, to ensure efficient operating, several sequences may take place in parallel.

The processes carried out in the single reactor vessel shown in Figure 2.1 are often only part of a larger process as is shown in Figure 2.2. In this plant two reactor vessels ($R1$ and $R2$) are used alternately, so that the processes of preparing for the next batch and cleaning up after a batch can be carried out in parallel with the actual production. Assuming that $R1$ has been filled with the mixture and the catalyst, and the reaction is in progress, there will be for $R1$: loop control of the temperature and pressure; operation of the stirrer; and timing of the reaction (and possibly some in-process measurement to determine the state of the reaction). In parallel with this, vessel $R2$ will be cleaned – the washdown sequence – and the next batch of raw material will be measured and mixed in the mixing tank. Meanwhile, the previous batch will be thinned down and transferred to the appropriate storage tank and, if there is to be a change of product or a change in product quality, the thin-down tank will be cleaned. Once this is done the next batch can be loaded into $R2$ and then, assuming that the reaction in $R1$ is complete, the contents of $R1$ will be transferred to the thin-down tank and the washdown procedure for $R1$ initiated.

The various sequences of operations required can become complex and there may also be complex decisions to be made as to when to begin a sequence. The sequence initiation may be left to a human operator or the computer may be programmed to supervise the operations (*supervisory control* – see below). The decision to use human or computer supervision is often very difficult to make. The

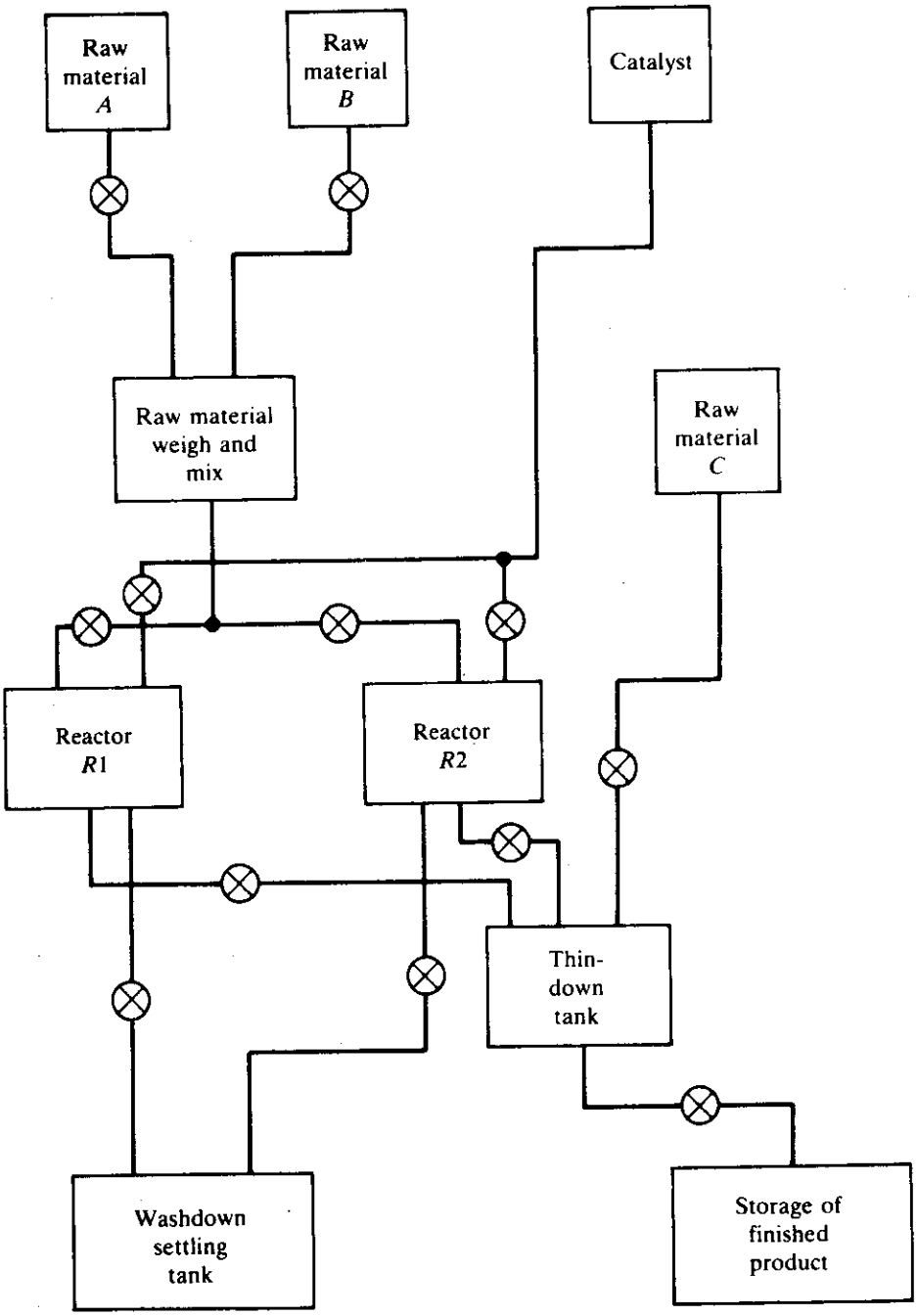

Figure 2.2 Typical chemical batch process.

aim is usually to minimise the time during which the reaction vessels are idle since this is unproductive time. The calculations needed and the evaluation of options can be complex, particularly if, for example, reaction times vary with product mix, and therefore it would be expected that decisions made using computer supervisory control would give the best results. It is certainly true that completely automatic control of the process would avoid the quite natural tendency for operators to avoid starting the next batch sequence close to the end of their shift; however, it is difficult using computer control to obtain the same flexibility that can be achieved using a human operator (and to match the ingenuity of good operators). As a consequence many supervisory systems are mixed; the computer is programmed to carry out the necessary supervisory calculations and to present its decisions for confirmation or rejection by the operator, or alternatively it presents a range of options to the operator.

In most batch systems there is also, in addition to the sequence control, some continuous feedback control: for example, control of temperatures, pressures, flows, speeds or currents. In process control terminology continuous feedback control is referred to as *loop control* or *modulating control* and in modern systems this would be carried out using DDC.

A similar mixture of sequence, loop and supervisory control can be found in continuous systems. Consider the float glass process shown in Figure 2.3. The raw material – sand, powdered glass and fluxes (the frit) – is mixed in batches and fed into the furnace. It melts rapidly to form a molten mixture which flows through the furnace. As the molten glass moves through the furnace it is refined. The process requires accurate control of temperature in order to maintain quality and to keep fuel costs to a minimum – heating the furnace to a higher temperature than is necessary wastes energy and increases costs. The molten glass flows out of the furnace and forms a ribbon on the float bath; again, temperature control is important as the glass ribbon has to cool sufficiently so that it can pass over rollers without damaging its surface. The continuous ribbon passes into the lehr where it is annealed and where temperature control is again required. From the lehr the glass ribbon moves down the line towards the cut-up stations at a speed which is too great for manual inspection so automatic inspection is used, faults being marked by spraying paint onto the ribbon. It then passes under the cutters which cut it into sheets of the required size; automatic stackers then lift the sheets from the production line. The whole of this process is controlled by several computers and involves loop, sequence and supervisory control.

Sequence control systems can vary from the large – the start-up of a large boiler turbine unit in a power station when some 20 000 operations and checks may have to be made – to the small – starting a domestic washing machine. Most sequence control systems are simple and frequently have no loop control. They are systems which in the past would have been controlled by relays, discrete logic, or integrated circuit logic units. Examples are simple presses where the sequence might be: locate blank, spray lubricant, lower press, raise press, remove article, spray lubricant. Special computer systems known as *programmable logic controllers* (PLCs)

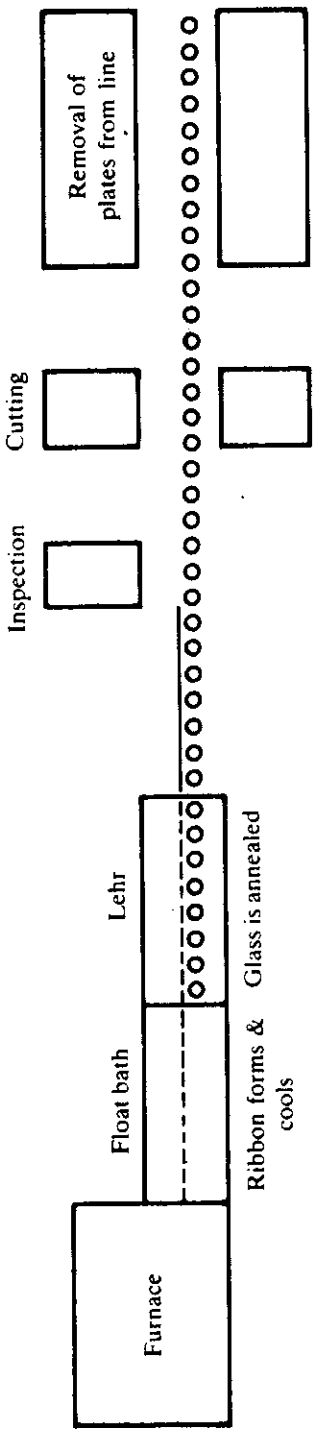

Figure 2.3 Schematic of float glass process.

have been developed for these simple sequence systems together with special programming languages (Henry, 1987; Kissell, 1986).

2.3 LOOP CONTROL (DIRECT DIGITAL CONTROL)

In direct digital control (DDC) the computer is in the feedback loop as is shown in Figure 2.4. This is a generalised representation of the system shown in Figures 1.4 and 1.5; the system shown in Figure 2.4 is assumed to involve several control loops all of which are handled within one computer. A consequence of the computer being in the feedback loop is that it forms a *critical* component in terms of the reliability of the system and hence great care is needed to ensure that, in the event of the failure or malfunctioning of the computer, the plant remains in a safe condition. The usual means of ensuring safety are to limit the DDC unit to making *incremental* changes to the actuators on the plant; and to limit the rate of change of the actuator settings (the actuators are labelled *A* in Figure 2.4). The advantages claimed for DDC over analog control are:

1. Cost – a single digital computer can control a large number of loops. In the early days the break-even point was between 50 and 100 loops, but now with the introduction of microprocessors a single-loop DDC unit can be cheaper than an analog unit.

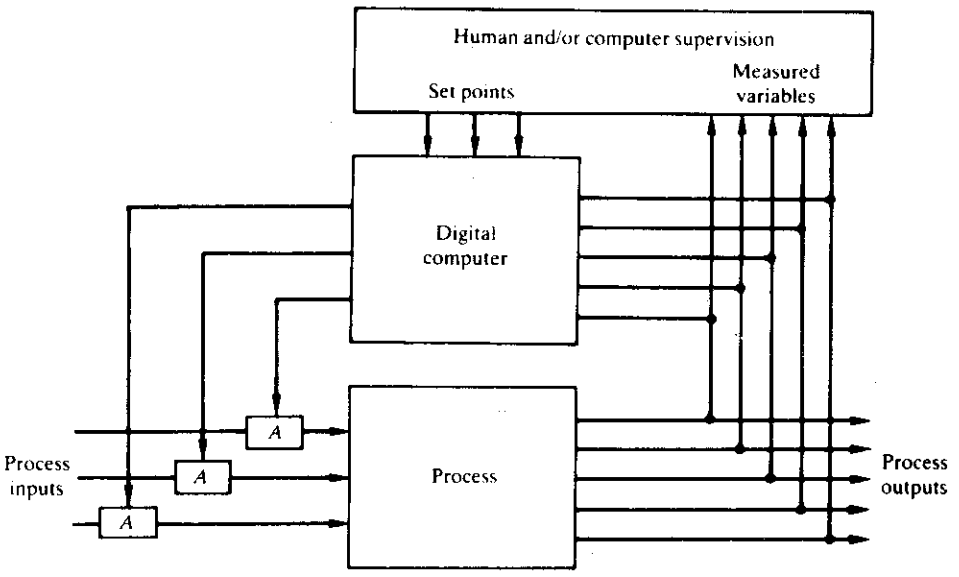

Figure 2.4 Direct digital control.

2. Performance – digital control offers simpler implementation of a wide range of control algorithms, improved controller accuracy and reduced drift.
3. Safety – modern digital hardware is highly reliable with long mean-time-between-failures and hence can improve the safety of systems. However, the software used in programmable digital systems may be much less reliable than the hardware.

The development of integrated circuits and the microprocessor have ensured that in terms of cost the digital solution is now cheaper than the analog. Single-loop controllers used as stand-alone controllers are now based on the use of digital techniques and contain one or more microprocessor chips which are used to implement DDC algorithms.

The adoption of improved control algorithms has, however, been slow. Many computer control implementations have simply taken over the well-established analog PID (Proportional + Integral + Derivative) algorithm.

2.3.1 PID Control

The PID control algorithm has the general form

$$m(t) = K_p[e(t) + 1/T_i \int_0^t e(t)dt + T_d de(t)/dt] \tag{2.1}$$

where $e(t) = r(t) - c(t)$ and $c(t)$ is the measured variable, $r(t)$ is reference value or set point, and $e(t)$ is error; K_p is the overall controller gain; T_i is the integral action time; and T_d is the derivative action time. This algorithm can be expressed in many other forms and we will deal with the details of it and methods of implementation in Chapter 4.

For a wide range of industrial processes it is difficult to improve on the control performance that can be obtained by using either PI or PID control (except at considerable expense) and it is for this reason that the algorithms are widely used. For the majority of systems PI control is all that is necessary. Using a control signal that is made proportional to the error between the desired value of an output and the actual value of the output is an obvious and (hopefully) a reasonable strategy. The ratio between the control signal and the error signal can be adjusted using the proportionality constant (gain) K_p. Choosing the value of K_p involves a compromise: a high value of K_p gives a small steady-state error and a fast response, but the response will be oscillatory and may be unacceptable in many applications; a low value gives a slow response and a large steady-state error. By adding the integral action term the steady-state error can be reduced to zero since the integral term, as its name implies, integrates the error signal with respect to time. For a given error value the rate at which the integral term increases is determined by the integral action time T_i. The major advantage of incorporating an integral term arises from the fact that it compensates for changes that occur in the process being controlled.

A purely proportional controller operates correctly only under one particular set of process conditions: changes in the load on the process or some environmental condition will result in a steady-state error; the integral term compensates for these changes and reduces the error to zero. For a few processes which are subjected to sudden disturbances the addition of the derivative term can give improved performance. Because derivative action produces a control signal that is related to the rate of change of the error signal, it *anticipates* the error and hence acts to reduce the error that would otherwise arise from the disturbance.

There is much more to computer control than simple DDC and the use of DDC is not limited to the PID control algorithm; algorithms developed using various digital control design techniques can be equally effective and a lot more flexible than the three-term controller. However, the art of tuning PID controllers is well established and the technique gives a well-behaved controller so that the introduction of new techniques has been slow. In fact, because the PID controller copes perfectly adequately with 90% of all control problems, it provides a strong deterrent to the adoption of new control system design techniques.

2.3.2 DDC Applications

DDC may be applied either to a single-loop system implemented on a small microprocessor or to a large system involving several hundred loops. The loops may be cascaded, that is with the output or actuation signal of one loop acting as the set point for another loop, signals may be added together (ratio loops) and conditional switches may be used to alter signal connections. A typical industrial system is shown in Figure 2.5. This is a steam boiler control system. The steam pressure is controlled by regulating the supply of fuel oil to the burner, but in order to comply with the pollution regulations a particular mix of air and fuel is required. We are not concerned with how this is achieved but with the elements which are required to implement the chosen control system.

The steam pressure control system generates an actuation signal which is fed to an auto/manual bias station. If the station is switched to auto then the actuation signal is transmitted; if it is in manual mode a signal which has been entered manually (say, from a keyboard) is transmitted. The signal from the bias station is connected to two units, a high signal selector and a low signal selector each of which has two inputs and one output. The high selector transmits the higher of the two input signals, the low selector transmits the lower of the two inputs. The signal from the low selector provides the set point for the DDC loop controlling the oil flow, the signal from the high selector provides the set point for the air flow controller (two cascade loops). A ratio unit is installed in the air flow measurement line. A signal from the controller which monitors the combustion flames directly (using an optical pyrometer) is added to the air flow signal to provide the input to the air flow controller.

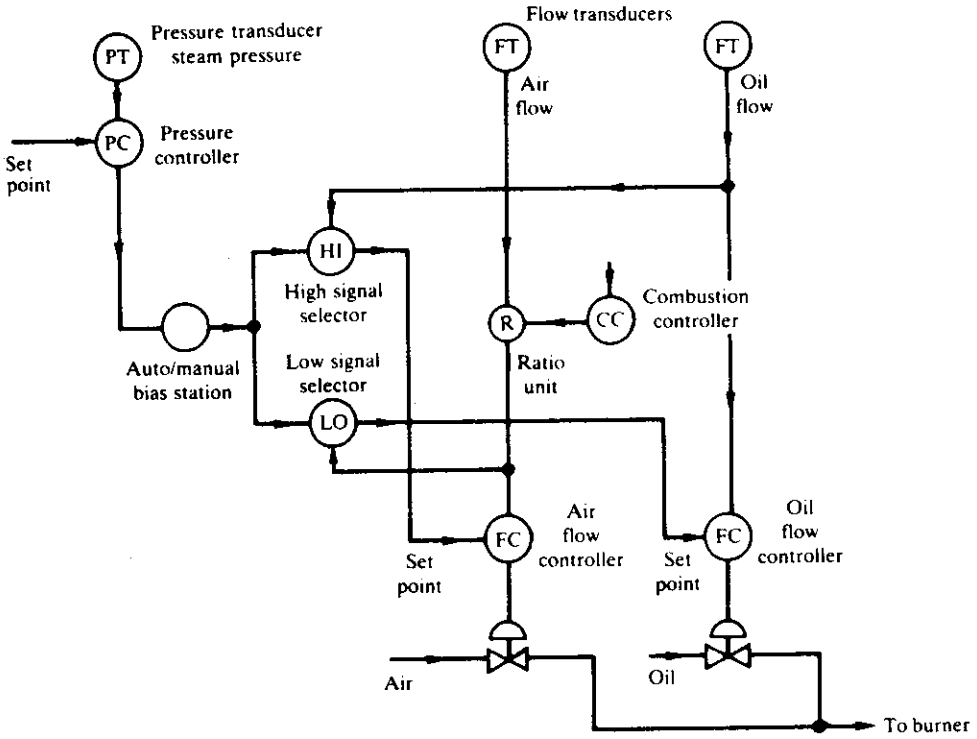

Figure 2.5 A boiler control scheme (redrawn from Bennett and Linkens, *Real-time Computer Control*, Peter Peregrinus (1984)).

DDC is not necessarily limited to simple feedback control as shown in Figure 2.6. It is possible to use techniques such as inferential, feedforward and adaptive or self-tuning control. Inferential control, illustrated in Figure 2.7, is the term applied to control where the variables on which the feedback control is to be based cannot be measured directly, but have to be 'inferred' from measurements of some other quantity. In Figure 2.7, some of the outputs can be measured and used directly in the feedback control; other outputs required by the controller cannot be measured directly and hence some other process measurement is made and from this the value of the controlled variable is inferred.

Inferential measurements are frequently used in distillation column control. A schematic of a binary distillation column is shown in Figure 2.8. The four independent variables usually controlled are the liquid levels H_a and H_b in the accumulator and the reboiler, and the compositions X_a and X_b of the top and bottom products. The compositions can be measured directly by spectrographic techniques but it is more usual to measure the temperatures at points Y_a and Y_b near the top and bottom of the column and the pressure P in the column. The temperatures represent the boiling points of the mixture at the position in the

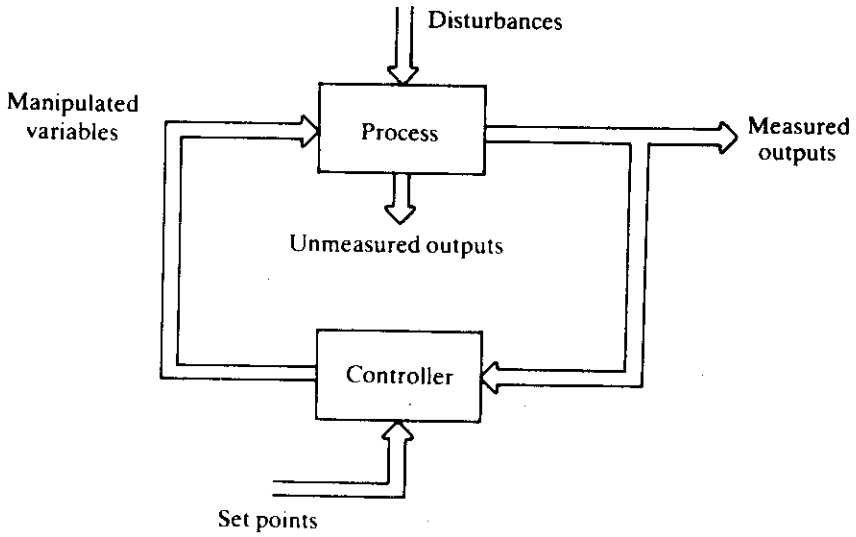

Figure 2.6 General structure of a feedback control configuration.

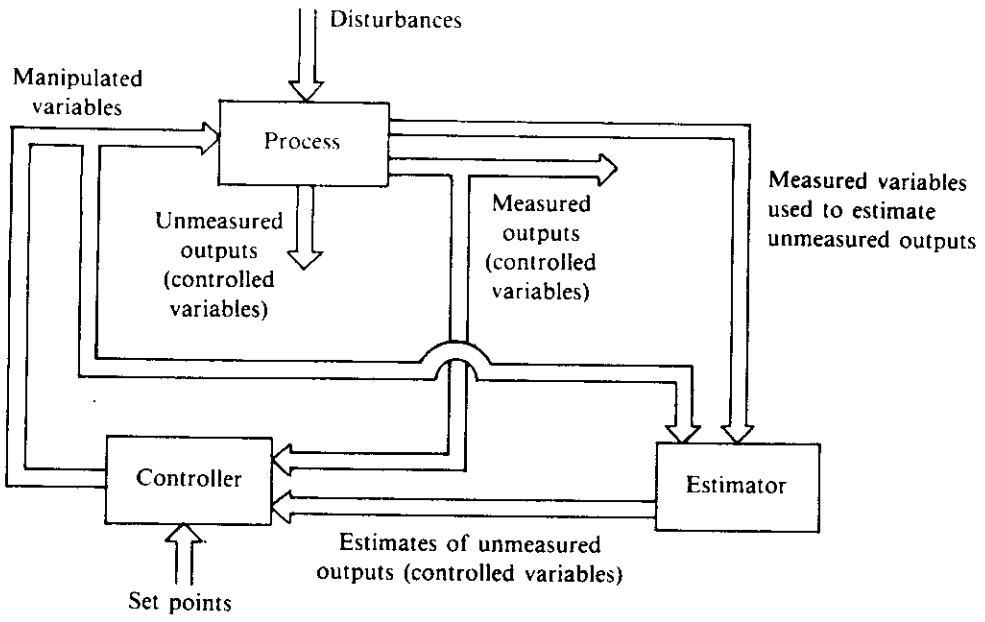

Figure 2.7 General structure of inferential control configuration.

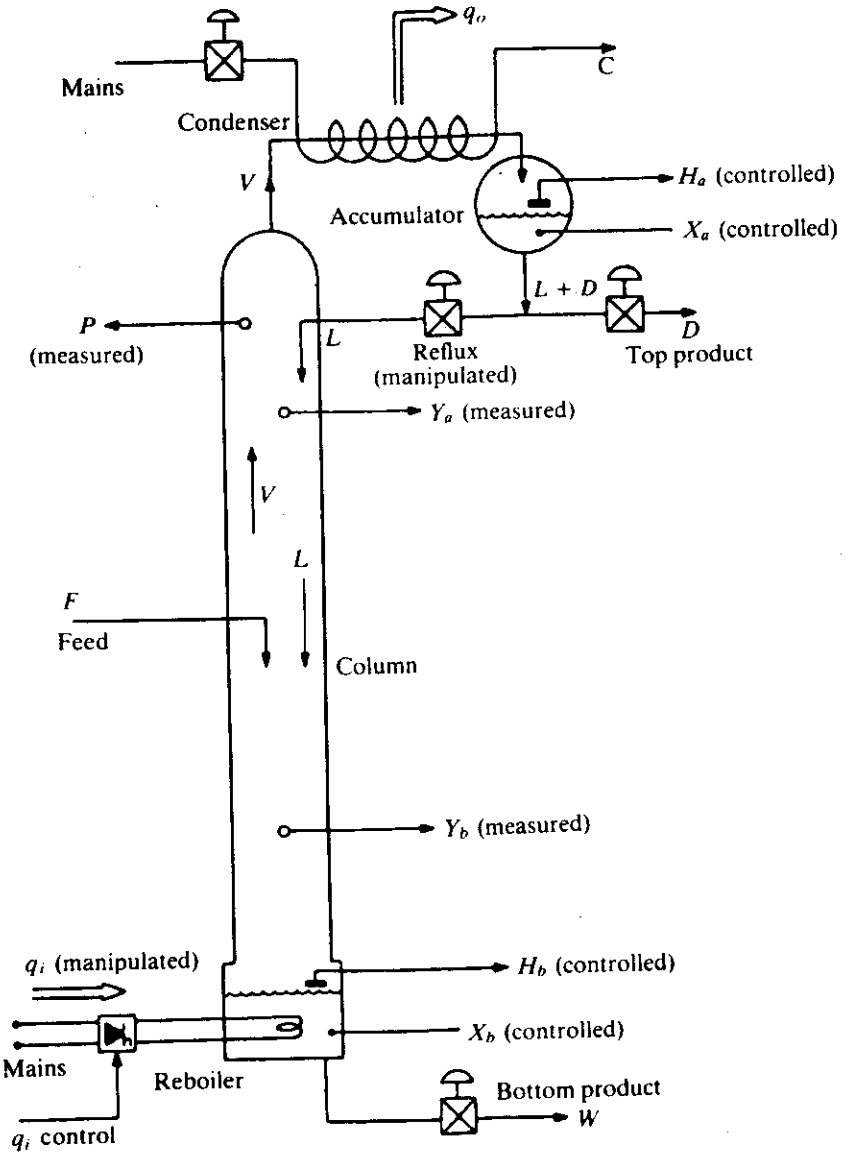

Figure 2.8 Manipulated and control variables for a binary distillation column (redrawn from Bennett and Linkens, *Computer Control of Industrial Processes*, Peter Peregrinus (1982)).

column and from measurements of pressure and temperature the compositions can be inferred (Edwards, 1982).

Feedforward control is frequently used in the process industries; it involves measuring the disturbances on the system rather than measuring the outputs and is illustrated in Figure 2.9. For example, in the hot rolling of sheet steel, if the temperature of the billet is known as it approaches the first-stage mill, the initial setting of the roll gap can be calculated accurately and estimates of the reduction at each stage of the mill can be made; hence the initial gaps for the subsequent stages can also be calculated. If this is done the time taken to get the gauge of the steel within tolerance can be much reduced and hence the quantity of scrap (out of tolerance) steel reduced. The effect of introducing feedforward control is to speed up the response of the system to disturbances; it can, however, only be used for disturbances which can be measured, and in plants where the effects of the disturbances can be predicted accurately.

2.3.3 Adaptive Control

Adaptive control can take several forms. Three of the most common are:

- preprogrammed adaptive control (gain scheduled control);
- self-tuning; and
- model-reference adaptive control.

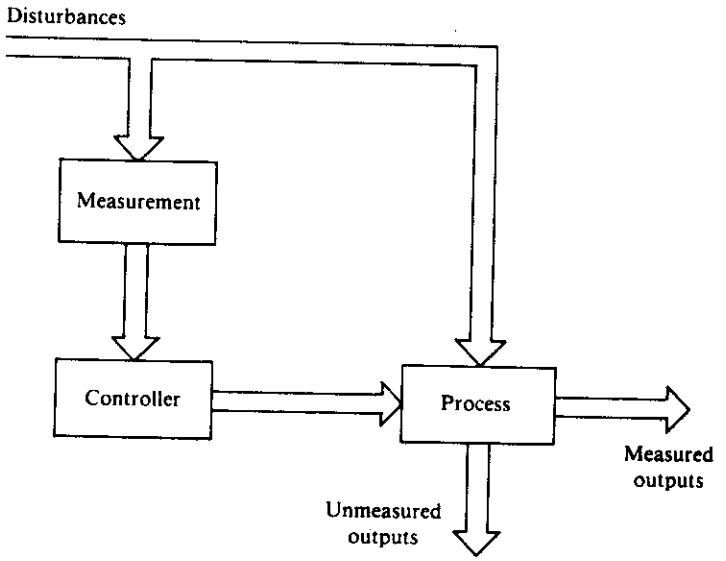

Figure 2.9 General structure of a feedforward control configuration.

Programmed adaptive control is illustrated in Figure 2.10a. The adaptive, or adjustment, mechanism makes preset changes on the basis of changes in auxiliary process measurements. For example, in a reaction vessel a measurement of the level of liquid in the vessel (an indicator of the volume of liquid in the vessel) might be used to change the gain of the temperature controller; in many aircraft controls the measured air speed is used to select controller parameters according to a preset schedule. An alternative form is shown in Figure 2.10b in which measurements of changes in the external environment are used to select the gain or other controller parameters. For example, in an aircraft autostabiliser, control parameters may be changed according to the external air pressure. Another example is the use of

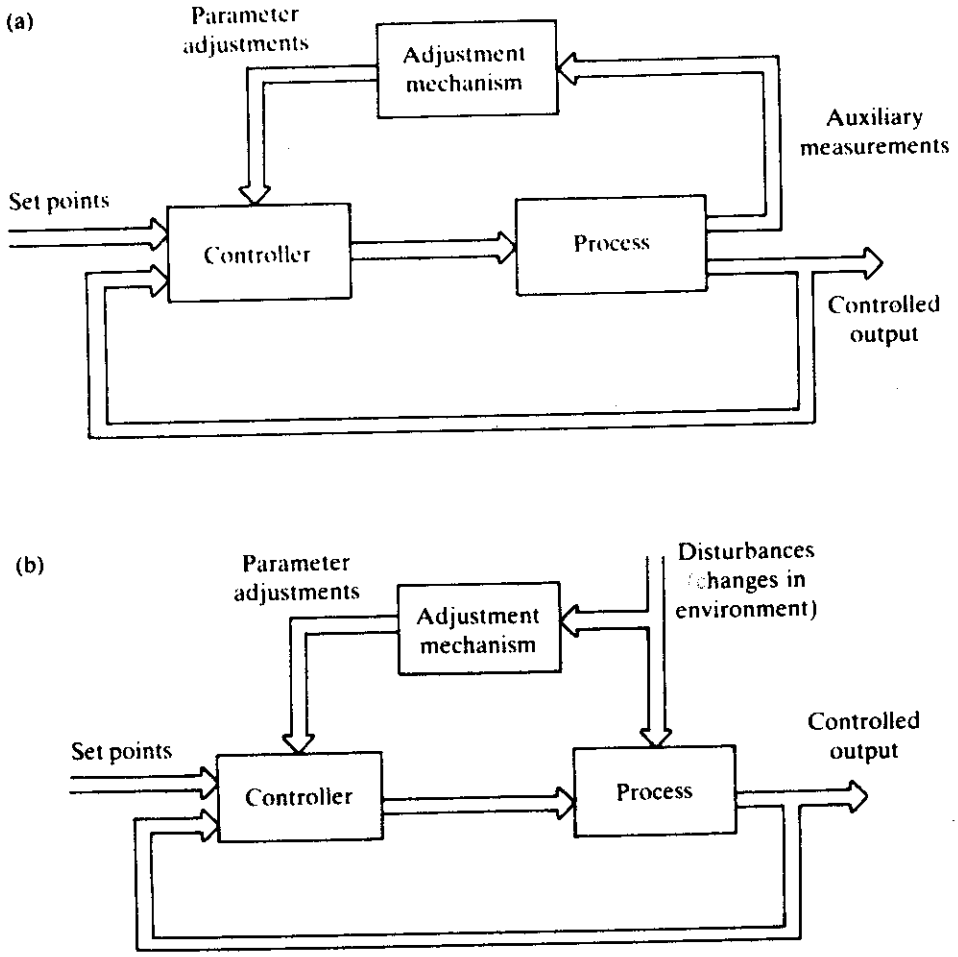

Figure 2.10 Programmed adaptive control (gain scheduled): (a) auxiliary process measurements; (b) external environment (open loop).

measurements of external temperature and wind velocities to adjust control parameters for a building environment control system.

Adaptive control using self-tuning is illustrated in Figure 2.11 and uses identification techniques to achieve continual determination of the parameters of the process being controlled; changes in the process parameters are then used to adjust the actual controller.

An alternative form of self-tuning is frequently found in commercial PID controllers (usually called autotuning). At operator-initiated intervals or periodically, the controller injects a small disturbance to the process and measures the response, the response is compared to some desired response and controller parameters are adjusted to bring the response closer to the desired response. The comparison may be based on a simple measure such as percentage overshoot or some more complex comparators.

The model reference technique is illustrated in Figure 2.12; it relies on the ability to construct an accurate model of the process and to measure the disturbances which affect the process.

2.4 SUPERVISORY CONTROL

The adoption of computers for process control has increased the range of activities that can be performed, for not only can the computer system directly control the operation of the plant, but also it can provide managers and engineers with a comprehensive picture of the status of the plant operations. It is in this *supervisory* role and in the presentation of information to the plant operator – large rooms full of dials and switches have been replaced by VDUs and keyboards – that the major changes have been made: the techniques used in the basic feedback control of the plant have changed little from the days when pneumatically operated three-term controllers were the norm. Direct digital control (DDC) is often simply the computer implementation of the techniques used for the traditional analog controllers.

Many of the early computer control schemes used the computer in a supervisory role and not for DDC. The main reasons for this were (a) computers in the early days were not always very reliable and caution dictated that the plant should still be able to run in the event of a computer failure; (b) computers were very expensive and it was not economically viable to use a computer to replace the analog control equipment in current use. A computer system that was used to adjust the set points of the existing analog control system in an optimum manner (to minimise energy or to maximise production) could perhaps be economically justified. The basic idea of supervisory control is illustrated in Figure 2.13 (compare this with Figure 2.4). The circles labelled C in Figure 2.13 represent individual controllers in the feedback loop; these can be themselves digital computers (or some other form of controller), but their operation is supervised by a digital computer.

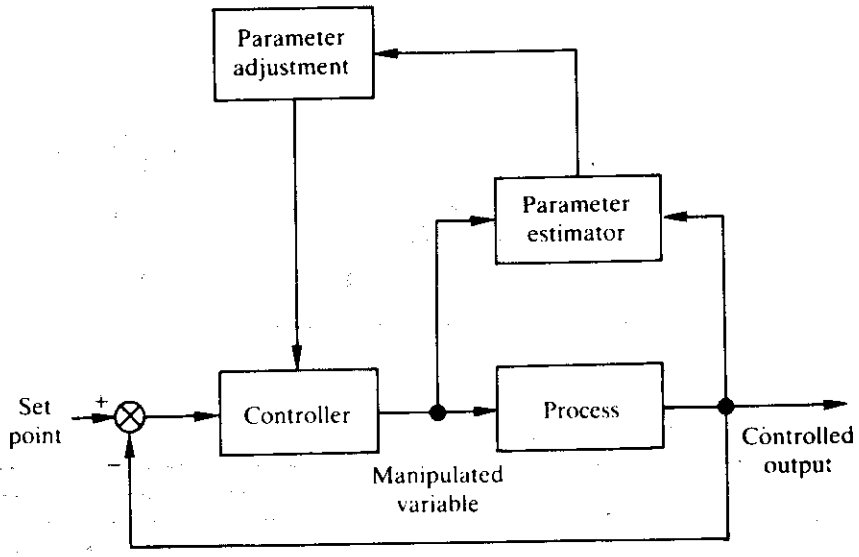

Figure 2.11 Self-tuning adaptive control.

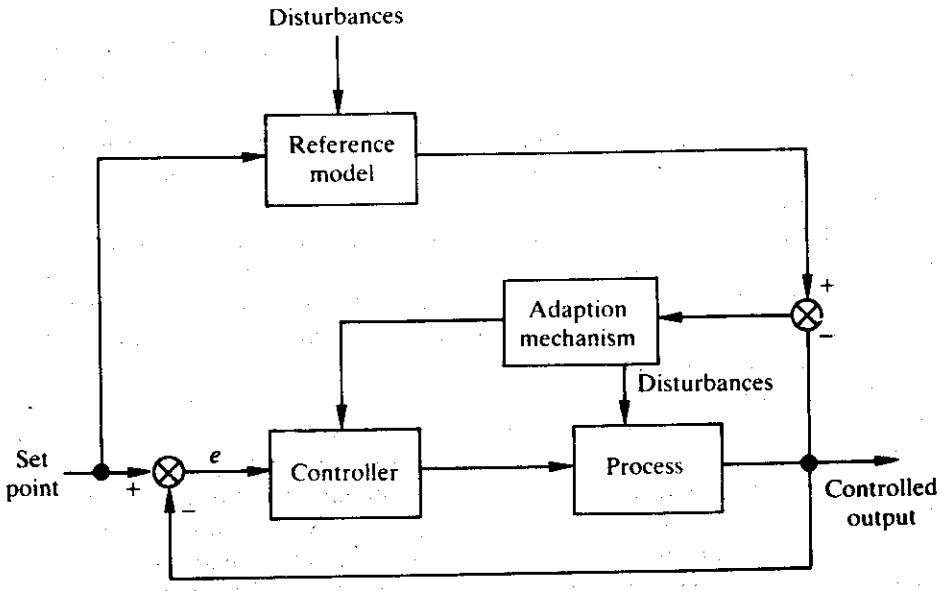

Figure 2.12 Model-reference adaptive control.

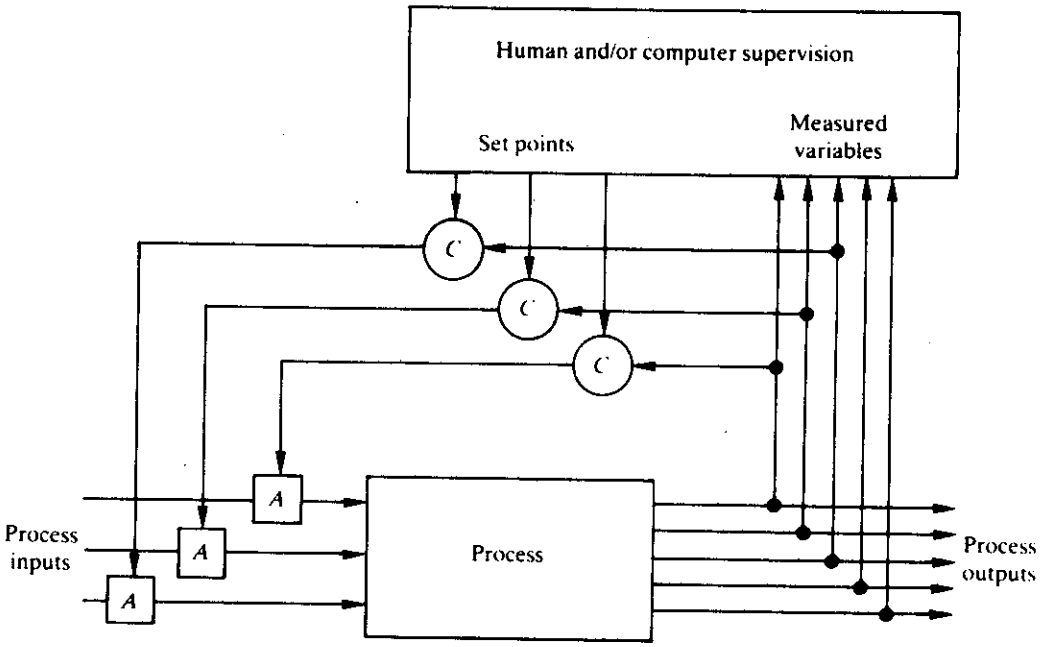

Figure 2.13 Supervisory control.

An example of supervisory control is shown in Figure 2.14. Two evaporators are connected in parallel and material in solution is fed to each unit. The purpose of the plant is to evaporate as much water as possible from the solution. Steam is supplied to a heat exchanger linked to the first evaporator and the steam for the second evaporator is supplied from the vapours boiled off from the first stage. To achieve maximum evaporation the pressures in the chambers must be as high as safety permits. However, it is necessary to achieve a balance between the two evaporators; if the first is driven at its maximum rate it may generate so much steam that the safety thresholds for the second evaporator are exceeded. A supervisory control scheme can be designed to balance the operation of the two evaporators to obtain the best overall evaporation rate.

Most applications of supervisory control are very simple and are based upon knowledge of the steady-state characteristics of the plant. In a few systems complex control algorithms have been used and have been shown to give increased plant profitability. The techniques used have included optimisation based on hill climbing, linear programming and simulations involving complex non-linear models of plant dynamics and economics. In these applications the complex algorithms have to be computed in real time in parallel with plant operation. There is now a growing interest in the use of *expert* systems to provide supervisory control (see Efstathiou, 1989; Bennett, 1992).

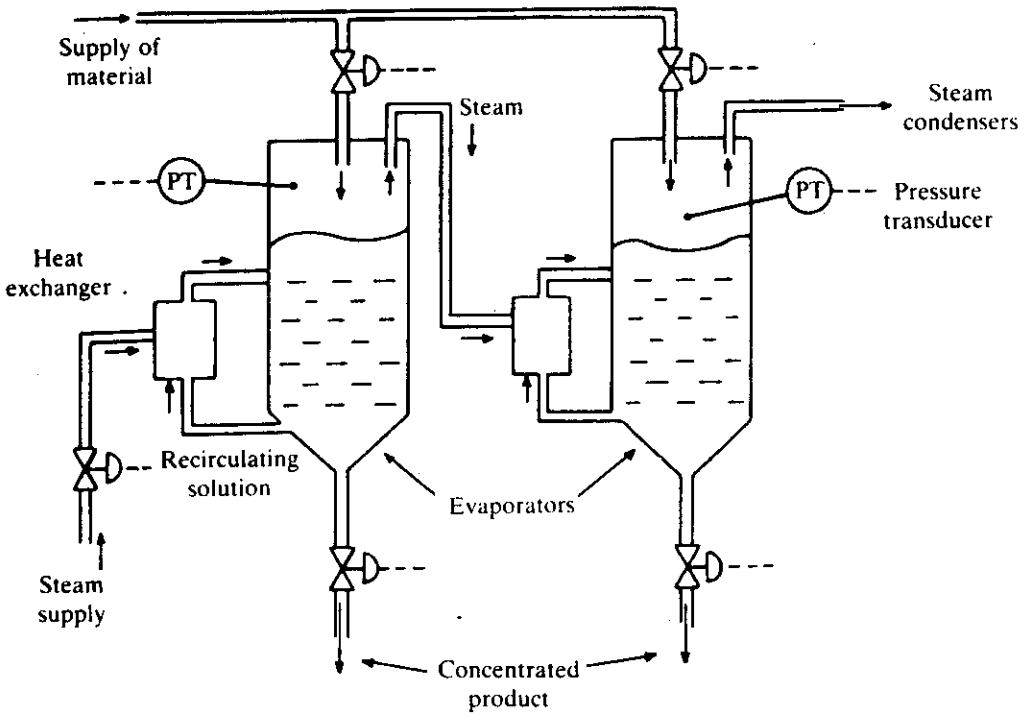

Figure 2.14 An evaporation plant (redrawn from Bennett and Linkens, *Real-time Computer Control*, Peter Peregrinus (1984)).

2.5 CENTRALISED COMPUTER CONTROL

Throughout most of the 1960s computer control implied the use of one central computer for the control of the whole plant. The reason for this was largely financial: computers were expensive. From the previous sections it should now be obvious that a typical computer-operated process involves the computer in performing many different types of operations and tasks. Although a general purpose computer can be programmed to perform all of the required tasks the differing time-scales and security requirements for the various categories of task make the programming job difficult, particularly with regard to the testing of software. For example, the feedback loops in a process may require calculations at intervals measured in seconds while some of the alarm and switching systems may require a response in less than 1 second; the supervisory control calculations may have to be repeated at intervals of several minutes or even hours; production management will want summaries at shift or daily intervals; and works management will require weekly or monthly analyses. Interrelating all the different time-scales can cause serious difficulties.

A consequence of centralised control was the considerable resistance to the use of DDC schemes in the form shown in Figure 2.4; with one central computer in the feedback loop, failure of the computer results in the loss of control of the whole plant. In the 1960s computers were not very reliable: the mean-time-to-failure of the computer hardware was frequently of the order of a few hours and to obtain a mean-time-to-failure of 3 to 6 months for the whole system required *defensive* programming to ensure that the system could continue running in a safe condition while the computer was repaired. Many of the early schemes were therefore for supervisory control as shown in Figure 2.13.

However, in the mid 1960s the traditional process instrument companies began to produce digital controllers with analog back-up. These units were based on the standard analog controllers but allowed a digital control signal from the computer to be passed through the controller to the actuator: the analog system tracked the signal and if the computer did not update the controller within a specified (adjustable) interval the unit dropped on to local analog control. This scheme

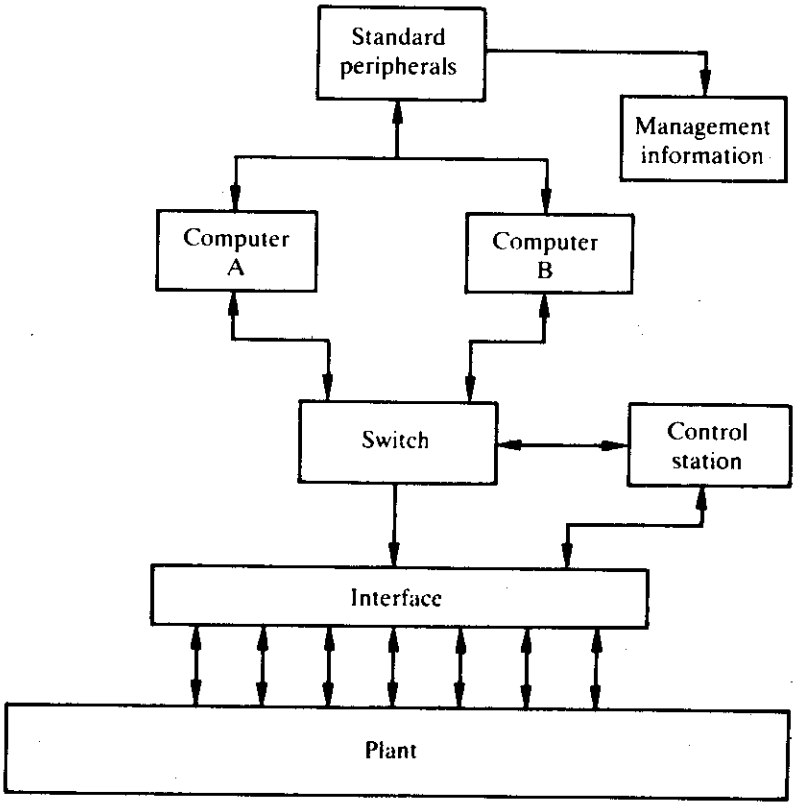

Figure 2.15 Dual computer scheme.

enabled DDC to be used with the confidence that if the computer failed, the plant could still be operated. The cost, however, was high in that two complete control systems had to be installed.

By 1970 the cost of computer hardware had reduced to such an extent that it became feasible to consider the use of dual computer systems (Figure 2.15). Here, in the event of failure of one of the computers, the other takes over. In some schemes the change-over is manual, in others automatic failure detection and change-over is incorporated. Many of these schemes are still in use. They do, however, have a number of weaknesses: cabling and interface equipment is not usually duplicated, neither is the software – in the sense of having independently designed and constructed programs – so that the lack of duplication becomes crucial. Automatic failure and change-over equipment when used becomes in itself a critical component. Furthermore, the problems of designing, programming, testing and maintaining the software are not reduced: if anything they are further complicated in that provision for monitoring ready for change-over has to be provided.

The continued reduction of the cost of hardware and the development of the microprocessor has made multi-computer systems feasible. These fall into two types:

1. Hierarchical – Tasks are divided according to function, for example with one computer performing DDC calculations and being subservient to another which performs supervisory control.
2. Distributed – Many computers perform essentially similar tasks in parallel.

The above can of course be combined to give a hierarchical, distributed system.

2.6 HIERARCHICAL SYSTEMS

This is the most natural development in that it follows the typical company decision-making structure shown in the pyramid in Figure 2.16: each decision element receives commands from the level above and sends information back to that level and, on the basis of information received from the element or elements below and from constraints imposed by elements at the same level, sends commands to the element(s) below and information to element(s) at the same level. This structure also follows a natural division of the production process in terms of the time-response requirements of the different levels. At the bottom of the pyramid, or hierarchy, a fast response (measured in milliseconds or seconds) to simple problems is required: as one progresses up the hierarchy the complexity of the calculations increases as does the time allowed for the response.

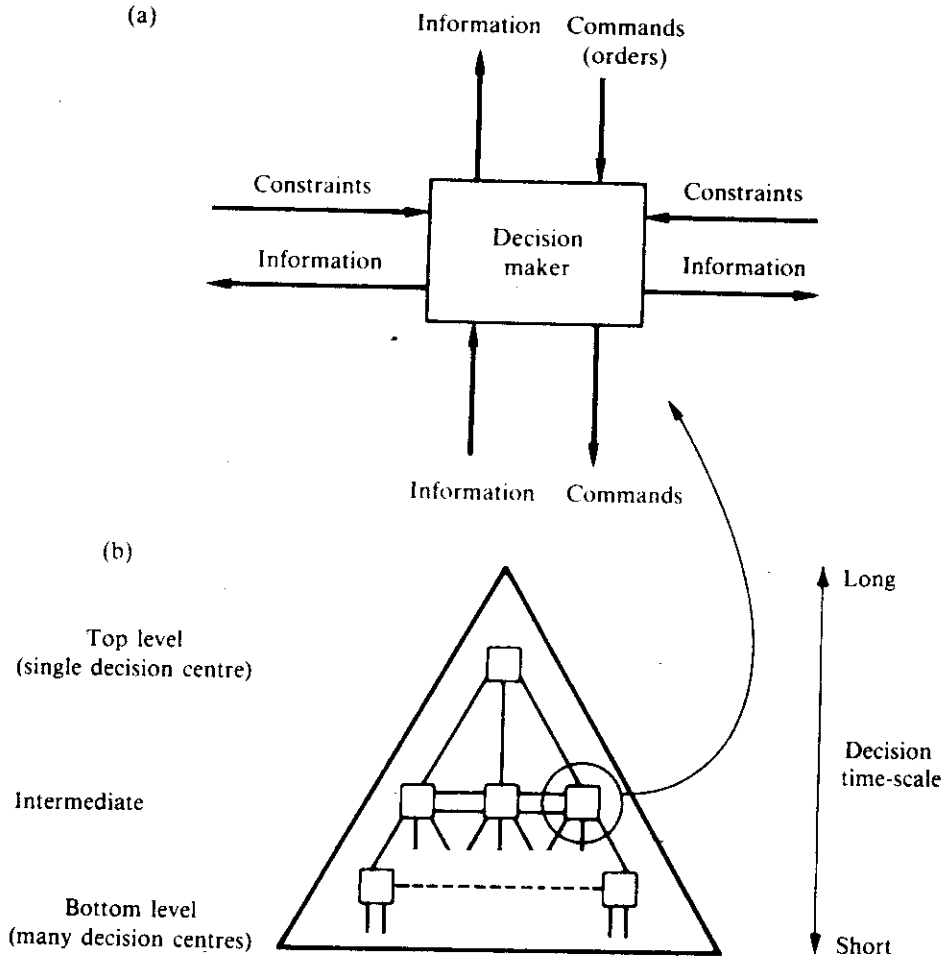

Figure 2.16 Hierarchical decision-making: (a) decision-making function; (b) decision-making structure.

A typical example of a hierarchical system is the batch system shown in Figures 2.17 and 2.18. This system has three levels which we have called manager, supervisor and unit control. It is assumed that single computers are used for the manager and supervisor functions and that for each processing unit a single unit control computer is used. At the manager level the functions such as resource allocation, production scheduling and production accounting are carried out. The input information may be, for example, sales orders (actual and forecast), stock levels, selling cost and production costs (or profit margins) on each product, operating costs for each process unit and scheduled maintenance plans for each operating unit, and the current state of production units. On the basis of this information the production schedule, that is the list of products to be produced and the quantities and process

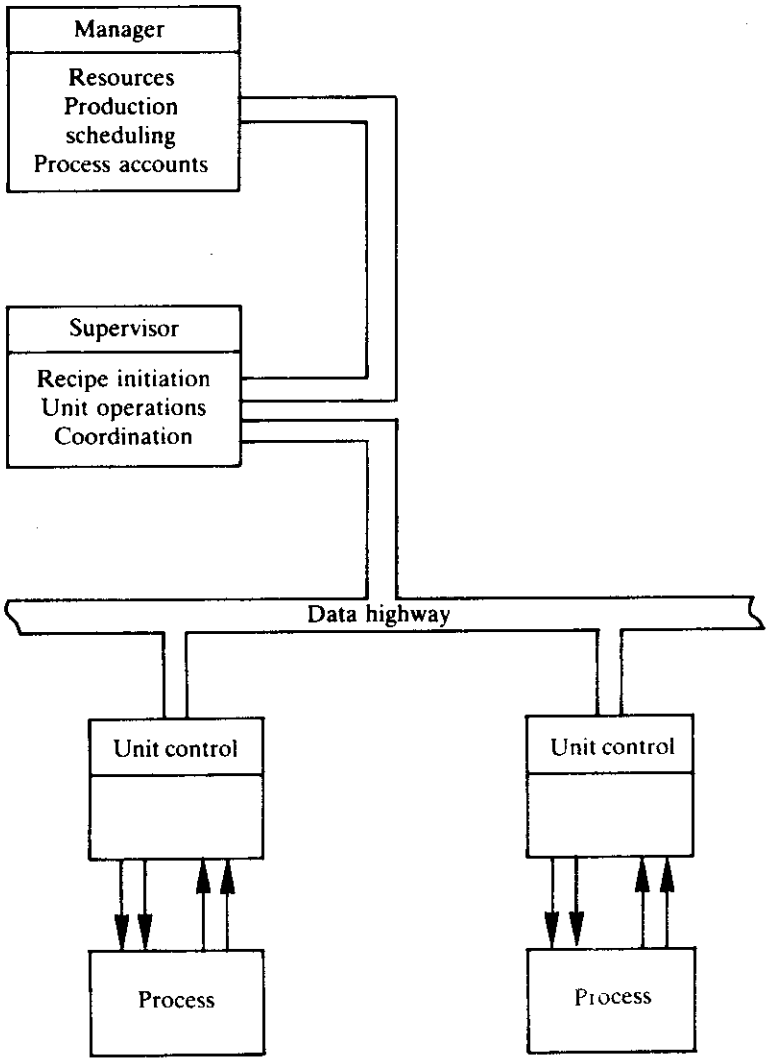

Figure 2.17 Batch control using a hierarchical system.

unit to be used, will be calculated. This may be done daily or at some other interval depending on production times, etc.

The information regarding the production schedule is transferred to the supervisor. It is assumed that the supervisor has a store containing the product recipe (that is, how to make a particular product) and a store of the operation sequences for making the product. When the appropriate unit, as selected by the production plan, is ready the information on the product – set points, alarm conditions, tolerances, etc. – is loaded down into the unit controller as are the

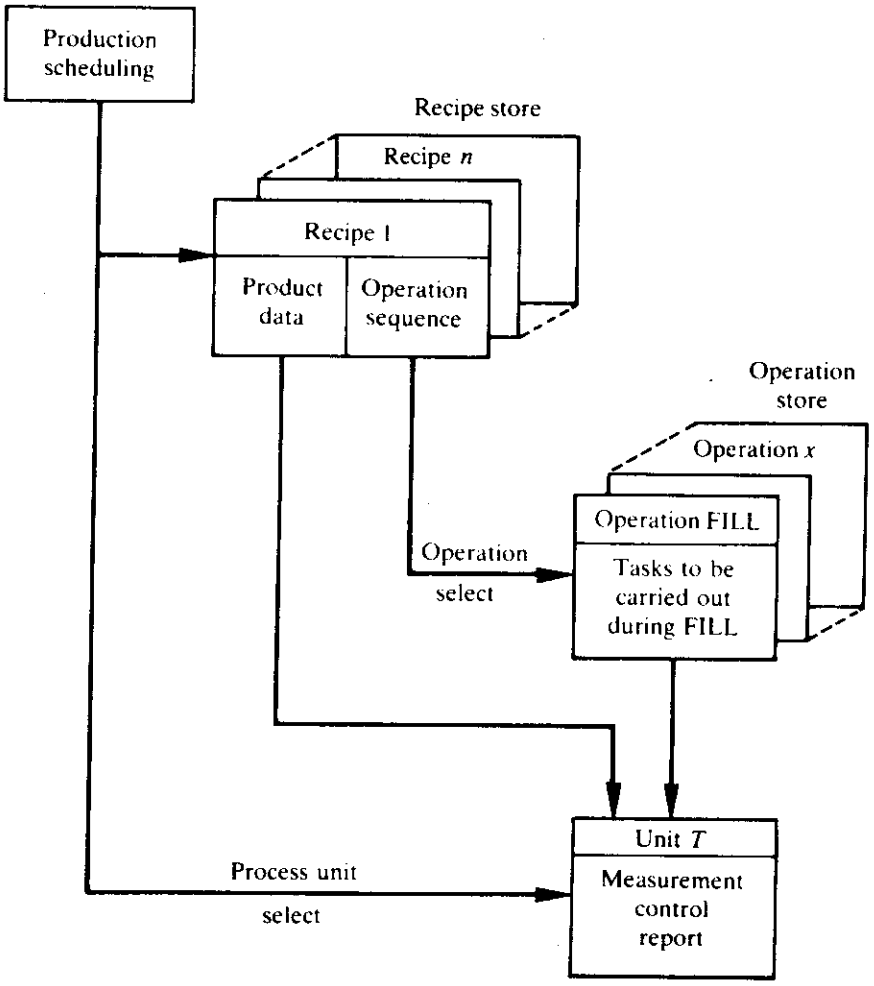

Figure 2.18 Batch control (software).

particular operations to be carried out during each stage of production. In addition the supervisor will receive regular reports of the progress of each process unit and will resolve any conflicts in the demand for resources, such as those that can arise where the units share a weigher (the supervisor will decide which is to use it at a particular time) or where the process requires energy input and the total available is limited, either because of the boiler capacity, or because penalties are imposed for exceeding a specified kV A rating (it will be the responsibility of the supervisor to decide, either automatically or through a request to a human supervisor, which unit is to reduce consumption or in what proportion consumption is to be reduced).

At the lowest level, the unit controllers are responsible for operating the plant:

opening and closing valves and switches, controlling temperatures, pressures, speeds, flows, monitoring alarms, and reporting plant conditions.

Most hierarchical systems will involve some form of distributed network and hence most systems will be a mixture of hierarchical and distributed control.

2.7 DISTRIBUTED SYSTEMS

The underlying assumptions of the distributed approach are that:

1. each unit is carrying out essentially similar tasks to all the other units; and
2. in the event of failure or overloading of a particular unit all or some of the work can be transferred to other units.

In other words, the work is not divided by function and allocated to a particular computer as in hierarchical systems: instead, the total work is divided up and spread across several computers. This is a conceptually simple and attractive approach – many hands make light work – but it poses difficult hardware and software problems since, in order to gain the advantages of the approach, allocation of the tasks between computers has to be dynamic, that is there has to be some mechanism which can assess the work to be done and the present load on each computer in order to allocate work. Because each computer needs access to all the information in the system, high-bandwidth data highways are necessary. There has been considerable progress in developing such highways and the various types are discussed below: computer scientists and engineers are also carrying out considerable research on multi-processor computer systems and this work could lead to totally distributed systems becoming feasible.

There is also a more practical approach to distributing the computing load whereby no attempt is made to provide for the dynamic allocation of resources but instead a simple *ad hoc* division is adopted with, for example, one computer performing all non-plant input and output, one computer performing all DDC calculations, another performing data acquisition and yet another performing the control of the actuators.

In most modern schemes a mixture of distributed and hierarchical approaches is used as shown in Figure 2.19. The tasks of measurement, DDC, operator communications, etc., are distributed among a number of computers which are linked together via a common serial communications highway and are configured in a hierarchical command structure. Five broad divisions of function are shown:

Level 1 All computations and plant interfacing associated with measurement and actuation. This level provides a measurement and actuation database for the whole system.

Level 2 All DDC calculations.

Level 3 All sequence calculations.
Level 4 Operator communications.
Level 5 Supervisory control.
Level 6 Communications with other computer systems.

It is not necessary to preserve rigid boundaries; for example, a DDC unit may perform some sequencing or may interface directly to plant.

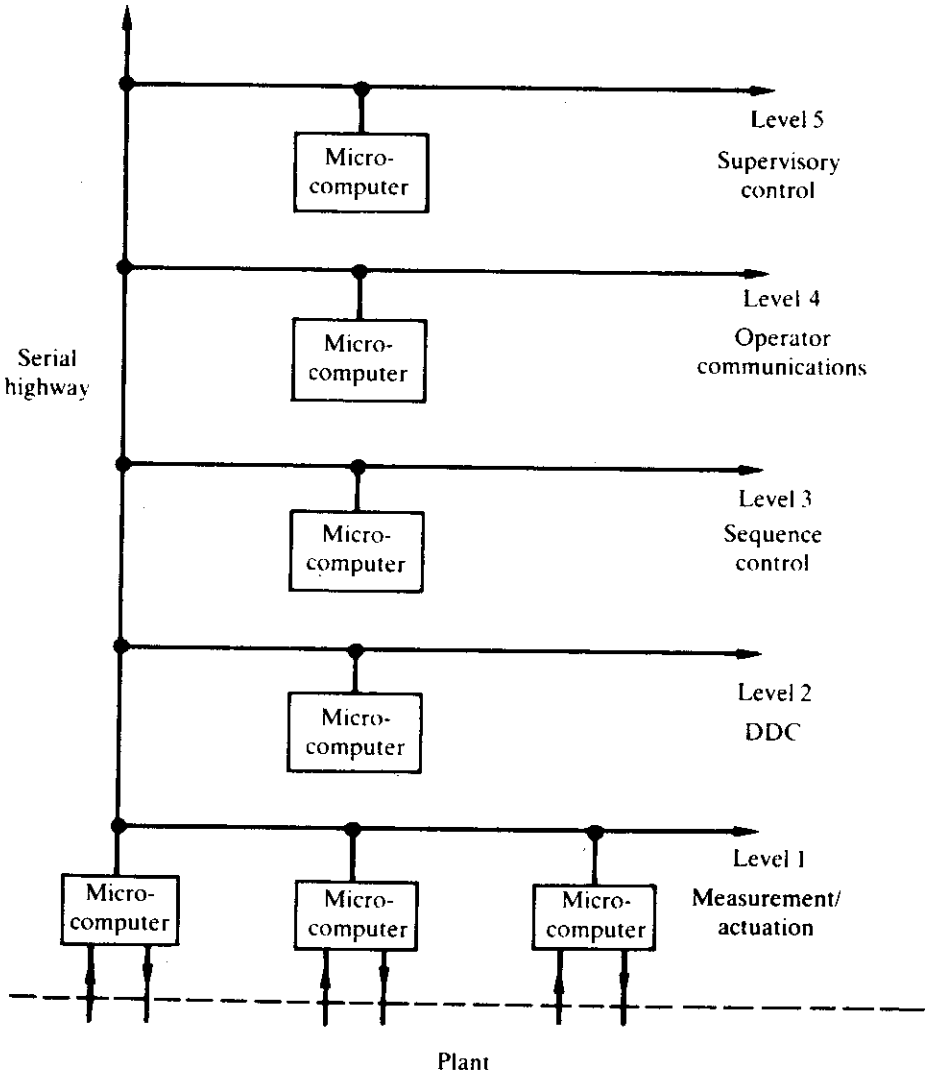

Figure 2.19 A distributed and hierarchical system.

The major advantages of this approach are:

1. The system capabilities are greatly enhanced by the sharing of tasks between processors – the burden of computation for a single processor becomes very great if all of the described control features are included. One of the main computing loads is that of measurement scanning, filtering and scaling, not because any one calculation is onerous but because of the large number of signals involved and the frequency at which the calculations have to be repeated. Separation of this aspect from the DDC, even if only into two processors, greatly enhances the number of control loops that can be handled. The DDC computer will collect measurements, already processed, via the communications link at a much lower frequency than that at which the measurement computer operates.
2. The system is much more flexible than the use of a single processor: if more loops are required or an extra operator station is needed, all that is necessary is to add more boxes to the communication link – of course the other units on the link will need to be updated to be aware of the additional items. It also allows standardisation, since it is much easier to develop standard units for well-defined single tasks than for overall control schemes.
3. Failure of a unit will cause much less disruption in that only a small portion of the overall system will not be working. Provision of automatic or semi-automatic transfer to a back-up system is much easier.
4. It is much easier to make changes to the system, in the form of either hardware replacements or software changes. Changing large programs is hazardous because of the possibility of unforeseen side-effects: with the use of small modules such effects are less likely to occur and are more easily detected and corrected.
5. Linking by serial highway means that the computer units can be widely dispersed: hence it is unnecessary to bring cables carrying transducer signals to a central control room.

2.8 HUMAN–COMPUTER INTERFACE (HCI)

The key to the successful adoption of a computer control scheme is often the facilities provided for the plant operator or user of the system. A simple and clear system for the day-to-day operation of the plant must be provided. All the information relevant to the current state of its operation should be readily available and facilities to enable interaction with the plant – to change set points, to adjust actuators by hand, to acknowledge alarm conditions, etc. – should be provided. A large proportion of the design and programming effort goes into the design and construction of operator facilities and the major process control equipment companies have developed extensive schemes for the presentation of information.

A typical operator station has specially designed keyboards and several display and printer units; extensive use is made of colour displays and mimic diagrams; video units are frequently provided to enable the operator to see parts of the plant (Jovic, 1986).

The standard software packages typically provide a range of display types: an alarm overview presenting information on the alarm status of large areas of the plant; a number of area displays presenting information on the control systems associated with each area; and loop displays giving extensive information on the details of a particular control loop. The exact nature of the displays is usually determined by the engineer responsible for the plant or part of the plant. Additional displays, including trends and summaries of past operations, are frequently available to the engineer, often in the form of hard copy. In addition the plant engineer (or maintenance engineer) will require information on which to base decisions about maintenance schedules and instrument, actuator and plant component replacements.

The plant manager requires access to different information: hard copy printouts – including graphs – that summarise the day-to-day operation of the plant and also provide a permanent plant operating history. Data presented to the manager will frequently have been analysed statistically to provide more concise information and to make decision-making more straightforward. The manager will be interested in assessing the economic performance of the plant and in determining possible improvements in plant operation.

The design of user interfaces is a specialist area. The safe operation of complex systems such as aircraft, nuclear power stations, chemical plants, air traffic control systems and other traffic control systems can be crucially affected by the way in which information is presented to the operator.

2.9 THE CONTROL ENGINEER

Assuming that a decision has been made on the most suitable computer system, the control engineer's responsibility is as follows:

1. To define the appropriate control strategy to meet the system requirements.
2. To define the measurements and actuations and to set up scaling and filter constants, alarm and actuator limits, sampling intervals, etc.
3. To define the DDC controllers, the interlinking or cascading of such controllers and the connections with any other elements in the control scheme.
4. To tune the control scheme, that is to select the appropriate gains so that it performs according to some desired specification.
5. To define and program the sequence control procedures necessary for the automation of plant operation.
6. To determine and implement satisfactory supervisory control schemes.

For a large project all the above requirements are too great for any one person to handle and in such cases a team of engineers would be involved. Additionally, if the programming of the system had to be done from scratch for each individual case, the task would be very burdensome and costly. However, process control applications have many features in common and the major suppliers of process control computer systems offer an extensive range of software packages, so that for each application the main task of the engineer/programmer is to select the appropriate modules and to assemble the required database describing the particular plant. The major omission of much of the standard software is that it fails to provide assistance with the tuning of the plant.

Software for the supervisory and sequencing operations has to be much more flexible in that the range of actions that may be required for any plant is much greater than for the DDC part. Frequently, it is necessary to program this part of the system specifically for each plant.

With the increasing use of microprocessors in a wide variety of applications the standard software approach used by the process controllers is becoming inadequate. It was based largely on the requirements of large plants with long time constants, controlled by a single large computer. Systems now have a computer or computers *embedded* as part of the plant or as part of a piece of equipment and are programmed to carry out only those functions required, that is they do not have within them general purpose software, a large proportion of which may be irrelevant to the particular application. This development has changed the approach to the design of real-time computer control systems and made it necessary for the control engineer to have a greater knowledge of programming languages and operating systems, as well as the techniques of distributed computing and communications.

2.10 ECONOMICS AND BENEFITS OF COMPUTER CONTROL SYSTEMS

Before the widespread availability of microprocessors, computer control was expensive and a very strong case was needed to justify the use of computer control rather than conventional instrumentation. In some cases computers were used because otherwise plant could not have been made to work profitably: this is particularly the case with large industrial processes that require complex sequencing operations. The use of a computer permits the repeatability that is essential, for example, in plants used for the manufacture of drugs. In many applications flexibility is important – it is difficult with conventional systems to modify the sequencing procedure to provide for the manufacture of a different product. Flexibility is particularly important when the product or the product specification may have to be changed frequently: with a computer system it is simple to maintain a database containing the product recipes and thus to change to a new recipe quickly and reliably.

The application of computer control systems to many large plants has frequently been justified on the grounds that even a small increase in productivity (say 1 or 2%) will more than pay for the computer system. After installation it has frequently been difficult to establish that an improvement has been achieved; sometimes production has decreased, but the computer proponents have then argued that but for the introduction of the computer system production would have decreased by a greater amount!

Some of the major benefits to accrue from the introduction of computer systems have been in the increased understanding of the behaviour of the process that has resulted from the studies necessary to design the computer system and from the information gathered during running. This has enabled supervisory systems to keep the plant running at an operating point closer to the desired point to be designed. The other main area of benefit has been in the control of the starting and stopping of batch operations in that computer-based systems have generally significantly reduced the dead time associated with batch operations.

The economics of computer control have been changed drastically by the microprocessor in that the reduction in cost and the improvement in reliability have meant that computer-based systems are the first choice in many applications. Indeed, microprocessor-based instrumentation is frequently cheaper than the equivalent analog unit. The major costs of computer control are now no longer the computer hardware, but the system design and the cost of software: as a consequence attention is shifting towards greater standardisation of design and of software products and the development of improved techniques for design (particularly software design) and for software construction and testing.

The availability of powerful, cheap and highly reliable computer hardware and communications systems makes it possible to conceive and construct large, complex, computer-based control systems. The complexity of such systems raises concern about their dependability and safety. A major concern arises from the difficulty of verifying the correctness of software and of validating a system which contains software. Verification is concerned with answering the question: are we building the product *correctly*? Validation is concerned with the question: are we building the *right* product?

2.11 SUMMARY

In this chapter we have described in some detail typical process control applications in which computer control is used. From these examples we have shown that the control aspects can be subdivided into several main activities which include:

- Data acquisition and processing.
- Sequence control.
- Loop control (modulating control).

- Supervisory control.
- Human–computer interfacing.

These activities are to be found in one form or another in all embedded computer applications and so although in this chapter we have concentrated on process control applications the ideas (and problems) are common to a wide range of other applications.

Also covered were the ways in which several computers can be configured for control applications. These include dual computer systems to increase reliability, and distributed and hierarchical configurations. A brief mention was made of some advanced control strategies.

EXERCISES

2.1 List the characteristics of (a) batch processes and (b) continuous processes.

2.2 You are the manager of a plant which can produce ten different chemical products in batches which can be between 500 and 5000 kg. What factors would you expect to consider in calculating the optimum batch size? What arguments would you put forward to justify the use of an on-line computer to calculate optimum batch size?

2.3 What are the advantages/disadvantages of using a continuous oven? How will the control of the process change from using a standard oven on a batch basis to using an oven in which the batch passes through on a conveyor belt? Which will be the easier to control?

2.4 List the advantages and disadvantages of using DDC.

2.5 List the advantages of using several small computers instead of one large computer in control applications. Are there any disadvantages that arise from using several computers?

2.6 In the section on human–computer interfacing we made the statement 'the design of user interfaces is a specialist area'. Can you think of reasons to support this statement and suggest what sort of background and training a specialist in user interfaces might require?

3

Computer Hardware Requirements for Real-time Applications

This chapter provides a brief overview of some of the basic ideas relating to computer hardware. A brief description of the various types of computers such as microprocessors, microcomputers and special purpose computers is given. A detailed explanation of the standard methods for data transfer including consideration of the use of interrupts is provided. Also given is a brief overview on communication methodologies. The emphasis throughout is on the principles involved and not on the characteristics of a particular microprocessor or microprocessor support chips.

The aims of the chapter are to provide:

- A basic description of the major features of microprocessors.
- A description of the standard interfacing techniques.
- An overview of the standard communication methodologies.

3.1 INTRODUCTION

Although almost any digital computer can be used for real-time computer control and other real-time operations, they are not all equally easily adapted for such work. In the majority of embedded computer-based systems the computer used will be a microprocessor, a microcomputer or a specialised digital processor. Specialised digital processors include fast digital signal processors, parallel computers such as the transputer, and special RISC (Reduced Instruction Set Computers) for use in safety-critical applications (for example, the VIPER (Cullyer and Pygott, 1987)).

3.2 GENERAL PURPOSE COMPUTER

The general purpose microprocessors include the Intel XX86 series, Motorola 680XX series, National 32XXX series and the Zilog Z80 and Z8000 series.