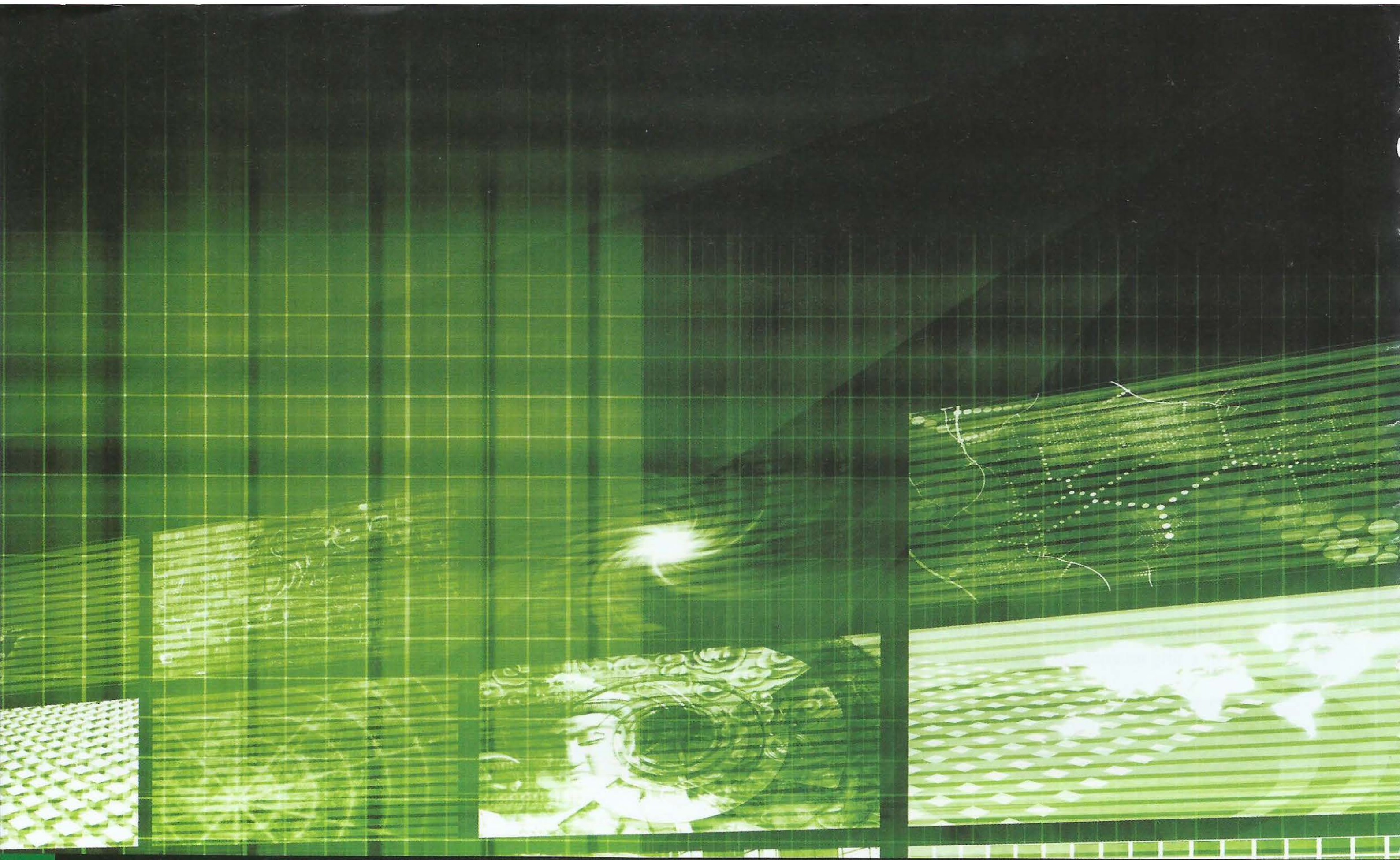

Chapter 22
Monitoring and Control Systems

Learning objectives

By the end of this chapter you should be able to:

- show understanding of the difference between a monitoring system and a control system
- show understanding of sensors and actuators and their usage
- show understanding of the additional hardware required to build these systems
- show understanding of the importance of feedback in a control system
- show understanding of the software requirements of these systems
- show understanding of how bit manipulation can be used to monitor/control a device
- carry out bit manipulations
- show understanding of how to make use of appropriate bit manipulation in monitoring systems and control systems.

22.01 Logistics

Monitoring can be used to describe a very wide range of activities but all are characterised by the measurement of some physical property. Typical examples of the physical property could be temperature, pressure, light intensity, flow rate or movement.

Let's consider temperature as an example. If this was being monitored under human control, the measurement could be made with a standard mercury thermometer. However, in this chapter we are interested in systems where a computer or microprocessor is being used. In this scenario, monitoring requires a measuring device that records a value which can be transmitted to the computer. Such a measuring device is a called a **sensor**. For monitoring temperature, a sensor could contain a thermocouple which outputs a temperature-dependent voltage.

There can only be one of two reasons to monitor a system:

- to check whether or not the monitored value is within acceptable limits; in a safety system, if the measured property has reached a dangerous level, some immediate action is required.
- to ensure routinely and continuously that the monitored property is as required; if the value measured indicates that a change has occurred, then the control part of the system may have to take measures to reverse this change.

The control element of a monitoring and control system needs a device, called an **actuator**. Figure 22.01 shows a schematic diagram of a computer-controlled environment.

KEY TERMS

Sensor: a hardware device that measures a property and transmits a value to a controlling computer

Actuator: a hardware device that receives a signal from a computer and adjusts the setting of a controlling device

Figure 22.01 includes an analogue-to-digital converter (ADC) and a digital-to-analogue converter (DAC) as separate components but they are likely to be integral to the device in the controlled environment. It should be noted that the diagram also shows an actuator as a single component. This is a simplification. An actuator is an electric motor that drives a controlling device which is not shown.

The system shown in Figure 22.01 involves a continuous process where a measurement is made and then, if needed, a control action is initiated. Following this control action, a measurement is taken again. There is therefore an element of feedback in the system.

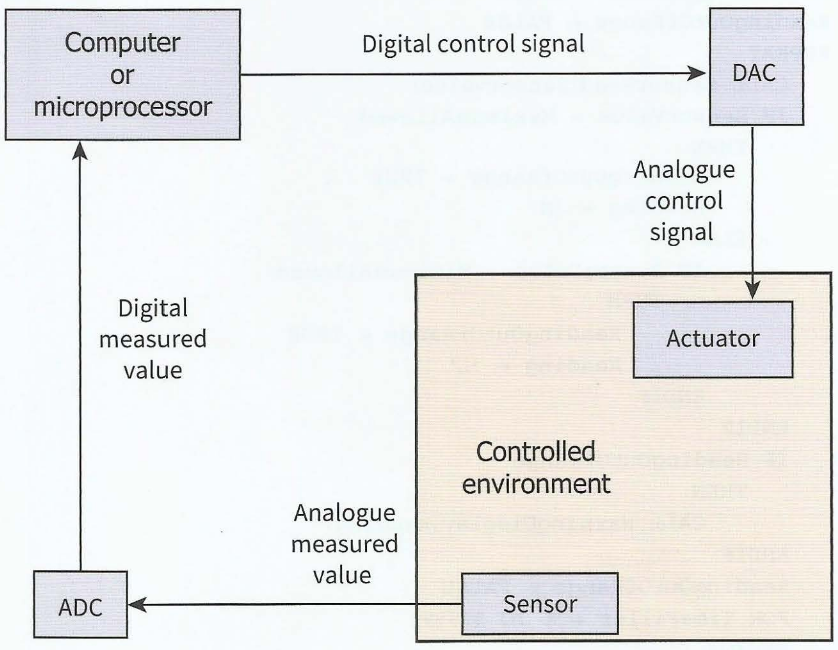

Figure 22.01 Computer-controlled environment

A closed-loop feedback control system is a special type of monitoring and control system where the feedback directly controls the operation. Figure 22.02 shows a schematic diagram of such a system. A microprocessor functions as the controller. This compares the value for the actual output, as read by the sensor, with the desired output. It then transmits a value to the actuator which depends on the difference calculated.

22.02 Real-time programming

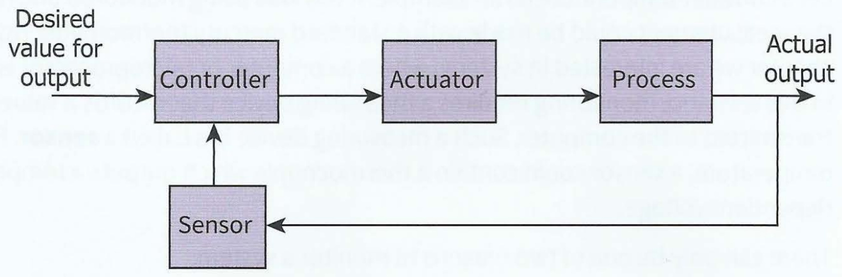

Figure 22.02 Closed-loop feedback control system

Monitoring and control systems require real-time programming. Whether a program is just monitoring or is monitoring and controlling, it must incorporate a structure for repetitive sensor reading. This must continue for the whole duration of the period that the system is switched on. A simple loop structure will achieve this but reading sensor values every clock cycle of a processor is unnecessarily frequent. The program must control the timing of the repetitions. This might be done by creating a timed sequence for reading values or possibly by including a time delay inside a loop.

Extension question 22.01

Research the capabilities for controlling the timing sequence for continuous running in your chosen programming language. Which ones would be best suited to a monitoring and control program?

An example monitoring program

Consider the following fragment of pseudocode:

```
EndReadingSensor ← FALSE
ReadingOutOfRange ← FALSE
REPEAT
   CALL SensorRead(SensorValue)
   IF SensorValue > MaximumAllowed
      THEN
         ReadingOutOfRange ← TRUE
         Reading ← 'H'
      ELSE
         IF SensorValue < MinimumAllowed
            THEN
               ReadingOutOfRange ← TRUE
               Reading ← 'L'
         ENDIF
   ENDIF
   IF ReadingOutOfRange
      THEN
         CALL WarningDisplay(Reading)
   ENDIF
   ReadingOutOfRange ← FALSE
   FOR TimeFiller ← 1 TO 999999
   ENDFOR
UNTIL EndReadingSensor
```

Note the following features of the program:

- There is an infinite loop.
- The loop finishes with another loop that does nothing other than create a delay before the outer loop repeats.
- When the sensor reading indicates a problem, the loop calls a procedure to handle whatever notification method is to be used.
- Following this call, the loop continues so the Boolean variable has to be reset to prevent the warning procedure being repetitively called.

An example monitoring and control program

Consider a system which is controlling an enclosed environment. The environment has a sensor to monitor a property and an actuator to control that property. The following fragment of code might be used:

```
EndReadingSensor ← FALSE
READ DesiredOutputLevel
REPEAT
   CALL SensorRead(SensorValue)
   SensorDifference ← DesiredOutputLevel - SensorValue
   IF ABS(SensorDifference) < DesiredOutputLevel/100
      THEN
         SensorDifference ← 0
   ENDIF
   IF SensorDifference > 0
      THEN
         ActuatorAdjustmentFactor ← SensorDifference/DesiredOutputLevel
         AdjustmentDirection ← 'up'
         CALL ActivateActuator(AdjustmentDirection, ActuatorAdjustmentFactor)
   ENDIF
   IF SensorDifference < 0
      THEN
         ActuatorAdjustmentFactor ← ABS(SensorDifference)/DesiredOutputLevel
         AdjustmentDirection ← 'down'
         CALL ActivateActuator(AdjustmentDirection, ActuatorAdjustmentFactor)
   ENDIF
   FOR TimeFiller ← 1 TO 999999
   ENDFOR
UNTIL EndReadingSensor
```

Note the following features of the program:

- A procedure is called to activate the actuator only if the sensor reading shows a significant change.
- The code will only work properly if it can be guaranteed that the activation of the actuator has caused a change in the property before the sensor reading in the next iteration of the loop.

22.03 Bit manipulation

The two fragments of code in Section 22.02 have a direct call to a procedure to take some action. A slightly different approach would be to set values for Boolean variables subject to what the sensors detect. For instance if a controlled environment had two properties to be monitored and controlled, four Boolean variables could be used. Values could be set by assignment statements such as:

```
IF SensorDifference1 > 0 THEN Sensor1HighFlag ← TRUE
IF SensorDifference1 < 0 THEN Sensor1LowFlag ← TRUE
IF SensorDifference2 > 0 THEN Sensor2HighFlag ← TRUE
IF SensorDifference2 < 0 THEN Sensor2LowFlag ← TRUE
```

Another part of the monitoring and control program would be checking whether any of the four flags were set. The machine code for running such a program could use individual bits to represent each flag. The way that flags could be set and read are illustrated by the following assembly language code fragments in which the three least significant bits (positions 0, 1 and 2) of the byte are used as flags:

`LDD 0034`	Loads a byte into the accumulator from an address
`AND #B00000000`	Uses a bitwise AND operation of the contents of the accumulator with the operand to convert each bit to 0
`STO 0034`	Stores the altered byte in the original address
⋮	
`LDD 0034`	
`XOR #B00000001`	Uses a bitwise XOR operation of the contents of the accumulator with the operand to toggle the value of the bit stored in position 0. This changes the value of the flag it represents.
`STO 0034`	
⋮	
`LDD 0034`	
`AND #B00000010`	Uses a bitwise AND operation of the contents of the accumulator with the operand to leave the value in position 1 unchanged but to convert every other bit to 0. A subsequent instruction can now compare the value of the byte with denary 2 to see if the flag represented by this bit position is set.
`STO 0034`	
⋮	
`LDD 0034`	
`OR #B00000100`	Uses a bitwise OR operation of the contents of the accumulator with the operand to set the flag represented by the bit in position 2. All other bit positions remain unchanged.
`STO 0034`	

Summary

- A monitoring system requires sensors; a monitoring and control system also requires actuators.
- A program used for a monitoring and control system has to operate in real time with an infinite loop that accepts input from the sensors at timed intervals.
- The program transmits signals to the actuators if the values received from the sensors indicate a need for control measures to be taken.
- Bit manipulation can be used within an assembly language program to monitor or control devices.

Exam-style Questions

1 A zoo reptile house has sixteen tanks which accommodate its reptiles. Each tank has to have its own microclimate where the appropriate levels of heat and humidity are crucial. The zoo implements a computer system which supplies the conditions in each of the tanks to a terminal in a central area. Warning messages are flashed up on the screen if any condition arises which requires the intervention of a zoo-keeper.

a State the name of the type of computing system described. [1]

b State **two** items of hardware which need to be present in the tanks for this system to function correctly. [2]

c This is the polling routine which is used to run the system indefinitely:

```
01 REPEAT
02    FOR i ← 1 TO ..............
03       READ Condition1, Condition2 in tank (i)
04       IF Condition1 < Extreme[i,1] OR Condition1 > Extreme[i,2]
05          THEN
06             OUTPUT "Warning! Problem in Tank ", i
07       ENDIF
08       IF Condition2 < Extreme[i,3] OR Condition3 > Extreme[i,4]
09          THEN
10             OUTPUT "Warning! Problem in Tank ", i
11       ENDIF
12    ENDFOR
13
14    FOR i ← 1 TO 999999
15    ENDFOR
16 UNTIL .............................
```

i Fill in the gaps in the pseudocode. [2]

ii Explain what is stored in the array `Extreme`. [2]

iii Explain what happens in lines 04 to 11. [3]

iv Explain the purpose of the loop in lines 14 to 15. [1]

d The zoo decides that the computer system needs to be updated. The computer system will now make use of actuators. These actuators will operate devices which adjust the microclimate.

Actuators can be in two states, on or off. Whether an actuator is on or off is determined by a single bit value (0 means off, 1 means on) in a specific 8-bit memory location.

The actuators to control the climate in Tank 4 use memory location 0804. Bit 5 of this memory location controls the heater.

7	6	5	4	3	2	1	0	bit number
0	0	1	1	0	1	0	1	value

Use some of the assembly language instructions to write the instructions that will ensure bit 5 of location 0804 is set to 1. [6]

Instruction		**Explanation**
Op Code	**Operand**	
`LDM #n`		Immediate addressing. Load the number n to ACC
`LLD <address>`		Direct addressing. Load the contents of the given address to ACC
`STO <address>`		Store the contents of ACC at the given address
`OUT`		Output to the screen the character whose ASCII value is stored in ACC
`AND #n`		Bitwise AND operation of the contents of ACC with the operand
`AND <address>`		Bitwise AND operation of the contents of ACC with the contents of <address>
`XOR #n`		Bitwise XOR operation of the contents of ACC with the operand
`OR #n`		Bitwise OR operation of the contents of ACC with the operand

Cambridge International AS and A Level Computer Science 9608 Specimen Paper 3 Q 3