Introduction to Computer Control

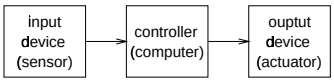
This lecture provides an overview of the topics that will be covered in the course.
After this lecture you should be able to give examples of computer control systems and identify their inputs and outputs.

Computer Control

This course is about using computers for industrial control applications. Although we will look briefly at computer architecture and the details of some input and output (I/O) devices, most of the course is concerned with implementing control systems using off-the-shelf computers and interfaces.

Examples

Computers are used to control everything from children's toys to nuclear power plants. Therefore it's not difficult to find examples of computer control. In spite of this wide range of applications, almost all computer control systems can be broken down into three components: input devices, a computer, and output devices:



For example, a microwave oven controller's inputs might include a keypad and a "door open" sensor. The outputs might include circuits to control the microwave power generator, a timer display and a buzzer.

A traffic light controller's inputs might include timers and vehicle sensors and the outputs would be the traffic lights.

The inputs to an airplane's autopilot might include speed, altitude and attitude sensors and the outputs would include actuators to control the throttle, rudder, elevator, etc.

Exercise: Give two additional examples of devices that are probably computer-controlled and identify the inputs and outputs.

Real-Time Control

One feature that distinguishes control applications from other computer applications is that control computers are usually *real-time* systems. This means that the system must respond to changes in the inputs within a certain maximum allowable delay. Of course, this maximum response time will depend on the application. In the case of an air-bag controller the maximum allowable delay might only be a few milliseconds while in the case of a control system for a water reservoir it might be minutes or hours. If an upper bound on the response time is a critical specification then the system is considered to be a real-time system.

The following definition of a real-time system was taken from the Usenet newsgroup `comp.realtime`:

"A real-time system is one in which the correctness of the computations not only depends upon the logical correctness of the computation but also upon the time at which the result is produced. If the timing constraints of the system are not met, system failure is said to have occurred. Hence, it is essential that the timing constraints of the system are guaranteed to be met. Guaranteeing timing behavior requires that the system be predictable. It is also desirable that the system attain a high degree of utilization while satisfying the timing constraints of the system."

"A good example is a robot that has to pick up something from a conveyor belt. The piece is moving, and the robot has a small window to pick up the object. If the robot is late, the piece won't be there anymore, and thus the job will have been done incorrectly, even though the robot went to the right place."

"Another example is the servo loops in an airplane when on auto-pilot. The sensors of the plane must continuously supply the control computer with proper measurements. If a measurement is missed, the performance of the airplane can degrade, sometimes to unacceptable levels."

"One will occasionally see references to "real-time" systems when what is is meant is "on-line," or "an interactive system with better response time than we used to have."

One will also occasionally see discussions of "soft" vs. "hard" real-time systems. In many of

these discussions, “hard” real-time means the type of real-time system discussed above, and “soft” real-time means systems which have reduced constraints on “lateness” but still must operate very quickly and repeatably. However, the definition is controversial, as some mean by “hard” and “soft” the degree of time constraints. For instance, a real-time process attempting to recognize images may have only a few hundred microseconds in which to resolve each image, but a process that attempts to position a servo-motor may have tens of milli-seconds in which to process its data.

“Unfortunately the topic is controversial, and there doesn’t seem to be 100% agreement over the terminology.”

Exercise: Decide whether each of the following computer systems is a real-time system. If so, give the response time that you expect would be required: an engine fuel injection controller, a washing machine controller, an airline reservation system, a computer accounting system.

Sensors

A sensor converts a physical quantity such as position, light intensity, temperature, pressure, etc. into an electrical signal that can be measured and processed by the computer.

A sensor’s output has to be converted to a number before it can be used by a computer. Often a sensor’s output can only take on one of two values and thus it can be easily converted into a 1-bit number. Other sensors generate a signal (a voltage or a current) that varies continuously. This signal must be quantized into one of a discrete range of values. The circuit that does this is called an analog-to-digital (A/D) converter.

Later in the course we will study various types of sensors and A/D converters in detail.

Exercise: What types of sensors might be used by the examples you chose earlier? Would they generate continuous (analog) or discrete (digital) values?

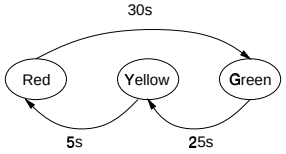
Microcomputers

A *microprocessor* is an integrated circuit (IC) that contains all of the electronics required to implement a computer except the memory and I/O devices. A *microcomputer* is a computer built around a microprocessor. It therefore includes some type of memory and I/O devices. Most modern computers are, in fact, microcomputers. A microcontroller is an IC that combines a microprocessor, memory and I/O devices on the same chip. Microcontrollers are used for simple control applications.

The microprocessor executes a sequence of computer instructions (a program) stored in its memory. The program determines the behaviour of the controller. Most of this course will be spent learning to write computer programs that will make the control system behave as desired. We will describe the behaviour of the controller in two ways: as a state machine and as a computer program written in the C language.

State Machines

All modern digital circuits (including all computers) can be described as state machines. A state machine representation describes the controller as a set of states plus rules that govern the transitions between these states. For example, a traffic light can be in one of three states (red, yellow or green) and the transitions between states (at least in a simple traffic light) are determined by the passage of time. The state diagram below shows the three states and the state transition conditions.



The simplified description provided by the state machine allows us to examine the behaviour of the controller without concerning ourselves with the implementation details.

C Programs

The operation of a controller can also be described in the form of a computer program. In some cases this is clearer than using a state machine. In this course we will use the C language, a popular language for microcomputer control applications. A part of a C program might look as follows:

```
void waitms ( int ms )
```

```
{
  int i ;

  while ( ms-- > 0 ) {
    i = 100 ;
    for ( i=MSLOOP ; i > 0 ; i-- ) ;
    if ( ( P3 & BUTTONMASK ) == 0 )
      pushed = 1 ;
  }
}
```

Note that the method used to describe the behaviour of the controller and its implementation need not be the same. For example, a state machine description can be easily converted into a computer program and there are special compilers (“logic synthesizers”) that can turn some computer programs into logic circuits.

Actuators

For a control system to have any effect it must have outputs. The outputs of most control system are electronically-controlled actuators such as solenoids or electric motors. These may produce the desired effect directly or they may control hydraulic or pneumatic valves.

Later in the course we will look at various types of actuators and simple control circuits.

Safety and Reliability

Some spectacular system failures have been attributed to poor design of computer control systems. In many cases the designers failed to take into account the consequences of certain hardware failures or unexpected combinations of inputs. By studying examples of failed systems we will derive principles that can be used to build safer and more reliable control systems.