CHAPTER 1

Tiny computers, hidden control

We are living in an age of information revolution, with computers of astonishing power available for our use. Computers find their way into every realm of activity. Some are developed to be as powerful as possible, without concern for price, for high-powered applications in industry and research. Others are designed for the home and office, less powerful but also less costly. Another category of computer is little recognised, partly because it is little seen. This is the type of computer that is designed into a product, in order to provide its control. This computer is hidden from view, such that the user often doesn’t know it’s even there. This sort of product is called an embedded system, and it is what this book is about. These little computers we generally call microcontrollers; it is one extended family of these that this book studies.

In this chapter you will learn about:

- The meaning of the term ‘embedded system’.
- The microcontroller which lies at the heart of the embedded system.
- The Microchip PIC family.
- An early PIC microcontroller, the 12F508.

1.1 The main idea – embedded systems in today’s world

1.1.1 What is an embedded system?

The basic idea of an embedded system is a simple one. If we take any engineering product that needs control, and if a computer is incorporated within that product to undertake the control, then we have an embedded system. An embedded system can be defined as [Ref. 1.1]:

A system whose principal function is not computational, but which is controlled by a computer embedded within it.

These days embedded systems are everywhere, appearing in the home, office, factory, car or hospital. Table 1.1 lists some example products that are likely to be embedded systems, all chosen for their familiarity. While many of these examples seem very different from each other, they all draw on the same principles as far as their characteristics as embedded systems are concerned.
The vast majority of users will not recognise that what they are using is controlled by one or more embedded computers. Indeed, if they ever saw the controlling computer they would barely recognise it as such. Most people, after all, recognise computers by their screen, keyboard, disc drives and so on. These embedded computers would have none of those.

### 1.2 Some example embedded systems

Let’s take a look at some example embedded systems, first from everyday life and then from the projects used to illustrate this book.

#### 1.2.1 The domestic refrigerator

A simple domestic refrigerator is shown in Figure 1.1. It needs to maintain a moderately stable, low internal temperature. It does this by sensing its internal temperature and comparing it with the temperature required. It lowers the temperature by switching on a compressor. The temperature measurement requires one or more sensors, and then whatever signal conditioning and data acquisition circuitry that is needed. Some sort of data processing is required to compare the signal representing the measured temperature to that representing the required temperature and deduce an output. Controlling the compressor requires some form of

![Figure 1.1: Embedded system example 1: the refrigerator](image)
electronic interface, which accepts a low-level input control signal and then converts this to the electrical drive necessary to switch the compressor power.

This process of control can be done by a conventional electronic circuit or it can be done by a small embedded computer. If used, the embedded computer could be designed simply to replicate the minimalist control process described above. Once a little computer is in place, however, there is tremendous opportunity for ‘added value’. With the signal in digital form and processing power now readily available, it is an easy step to add features like intelligent displays, more advanced control features, a better user control mechanism and so on.

Taking the idea of added value one step further, once an embedded computer is in place it is possible to network it to other computers, embedded or otherwise. This opens up wide new horizons, allowing a small system to become a subset of a much larger system and to share information with that system. This is now happening with domestic products, like the refrigerator, as well as much more complex items.

The diagram of Figure 1.1, while specific to a fridge, actually represents very well the overall concept of an embedded system. There is an embedded computer, engaged in reading internal variables, and outputting signals to control the performance of the system. It may have human interaction (but in general terms does not have to) and it may have networked interaction. Generally, the user has no idea that there’s a computer inside the fridge!

1.2.2 A car door mechanism

A very different example of an embedded system is the car door, as shown in Figure 1.2. Once again there are some sensors, some human interaction and a set of actuators that must respond to the requirements of the system. One set of sensors relates to the door lock and another to the window. There are two actuators, the window motor and the lock actuator.
It might appear that a car door could be designed as a self-contained embedded system, in a similar way to the fridge. Initially, one might even question whether it is worthy of any form of computer control whatsoever, as the functions seem so simple. Once again, by creating it as an embedded system, we see the opportunity to enhance functionality. Now we have the door status and actuators under electronic control, they can be integrated with the rest of the car. Central locking can be introduced or an alarm sounded if the door is not locked when the driver tries to pull away. There is therefore considerable advantage in having a network which links the humble actions of the door control to other functions of the car. We will see in later chapters that networked interaction is an important feature of the embedded system.

1.2.3 The electronic ‘ping-pong’ game

This little game, shown in Figure 1.3, is one of several projects used to illustrate the material of this book. It is a game for two players, who each have a push-button ‘paddle’. Either player can start the game by pressing his/her paddle. The ball, represented by the row of eight LEDs (light-emitting diodes), then flies through the air to the opposing player, who must press his paddle only when the ball is at the end LED and at no other time. The ball continues in play until either player violates this rule. Once this happens, the non-violating player scores and the associated LED is briefly lit up. When the ball is out of play, an ‘out-of-play’ LED is lit.

All the above action is controlled by a tiny embedded computer, a microcontroller, made by a company called Microchip [Ref. 1.2]. It takes the form of an 18-pin integrated circuit (IC), and has none of the visible features that one would normally associate with a computer. Nevertheless, electronic technology is now so advanced that inside that little IC there are a Central Processing Unit (CPU), a complex array of memories, and a set of timing and interface circuits. One of the memories contains a stored program, which it executes to run the game. It is able to read in as inputs the positions of the switches (the player paddles) and calculate the required LED positions. It then has the output capability to actually power the LEDs to which it is connected. All of this computing action is powered from only two AAA cells!

![Figure 1.3: The electronic ‘ping-pong’ game](image)
1.2.4 The Derbot Autonomous Guided Vehicle

Another project used later in this book is the Derbot Autonomous Guided Vehicle (AGV), pictured in Figure 1.4. How do its features compare with the examples seen thus far? Looking at the photograph, we can see from the front that it bristles with sensors and actuators. Two microswitch bump detectors sense if the Derbot hits an obstacle. An ultrasound detector, mounted on a servo actuator, is there with the aim of ensuring that the Derbot never has an unexpected collision! Two light sensors on either side of the servo are used for light tracking applications; a third, not seen in the photo, is mounted at the rear. A further navigational option is a compass, so that direction can be determined from the earth’s magnetic field. Locomotion is provided by two geared DC motors, while a sensor on each (again not seen in this picture) counts wheel revolutions to calculate actual distance moved. Steering is achieved by driving the wheels at different speeds. A piezo-electric sounder is included for the AGV to alert its human user. The Derbot is powered from six AA alkaline cells, which it carries on a power pack almost directly above its wheels. Its block diagram is shown in Figure 1.5.

As with earlier examples, the Derbot operates as an embedded system, reading in values from its diverse sensors and computing outputs to its actuators. It is controlled by another Microchip microcontroller, hidden from view in the picture by the battery pack. This microcontroller is seemingly more powerful than the one in the ping-pong game, as it needs to interface with far more inputs and drive its outputs in a more complex way.
Interestingly, as we shall see, the CPU of each microcontroller is the same. They are differentiated primarily by their interface capabilities. It is this difference that gives the Derbot microcontroller its far greater power.

### 1.3 Some computer essentials

When designing embedded systems we usually need to understand in some detail the features of the embedded computer that we are working with. This is quite unlike working with a desktop computer used for word processing or computer-aided design, where the internal workings are skilfully hidden from the user. As a preliminary to developing our knowledge, let us undertake a rapid survey of some important computer features.
1.3.1 Elements of a computer

Figure 1.6 shows the essential elements of any computer system. Fundamentally, it must be able to perform arithmetic or logical calculations. This function is provided by the Central Processing Unit (CPU). It operates by working through a series of instructions, called a program, which is held in its memory. Any one of these instructions performs a very simple function. However, because the typical computer runs so incredibly fast, the overall effect is one of very great computational power. Many instructions cause mathematical and logical operations to occur. These take place in a part of the CPU called the ALU, the Arithmetic Logic Unit.

To be of any use the computer must be able to communicate with the outside world, and it does this through its input/output. On a personal computer this implies human interaction, through the keyboard, VDU (Visual Display Unit) and printer. In an embedded system the communication is likely to be primarily with the physical world around it, through sensors and actuators.

The computer revolution that is taking place is due not only to the incredible processing power now at our disposal, but also to the equally incredible ability that we now have to store and access data. Broadly speaking there are two main applications for memory in a computer, as shown in Figure 1.6. One type of memory holds the program that the computer will execute. This memory needs to be permanent. If it is, then the program is retained indefinitely, whether power is applied or not, and it is ready to run as soon as power is applied. The other type of memory is used for holding temporary data, which the program works on as it runs. This memory type need not be permanent, although there is no harm if it is.

Finally, there must be data paths between each of these main blocks, as shown by the block arrows in the diagram.

1.3.2 Instruction sets – the Complex Instruction Set Computer and the Reduced Instruction Set Computer

Any CPU has a set of instructions that it recognises and responds to; all programs are built up in one way or another from this instruction set. We want computers to execute code as fast as
possible, but how to achieve this aim is not always an obvious matter. One approach is to build sophisticated CPUs with vast instruction sets, with an instruction ready for every foreseeable operation. This leads to the CISC, the Complex Instruction Set Computer. A CISC has many instructions and considerable sophistication. Yet the complexity of the design needed to achieve this tends to lead to slow operation. One characteristic of the CISC approach is that instructions have different levels of complexity. Simple ones can be expressed in a short instruction code, say one byte of data, and execute quickly. Complex ones may need several bytes of code to define them and take a long time to execute.

Another approach is to keep the CPU very simple and have a limited instruction set. This leads to the RISC approach – the Reduced Instruction Set Computer. The instruction set, and hence overall design, is kept simple. This leads to fast operation. One characteristic of the RISC approach is that each instruction is contained within a single binary word. That word must hold all information necessary, including the instruction code itself, as well as any address or data information also needed. A further characteristic, an outcome of the simplicity of the approach, is that every instruction normally takes the same amount of time to execute.

1.3.3 Memory types

Traditionally, memory technology has been divided into two categories:

- *Volatile*. This is memory that only works as long as it is powered. It loses its stored value when power is removed, but can be used as memory for temporary data storage. Generally, this type of memory uses simple semiconductor technology and is easier to write to from an electrical point of view. For historical reasons it has commonly been called RAM (Random Access Memory). A slightly more descriptive name is simply ‘data memory’.

- *Non-volatile*. This is memory that retains its stored value even when power is removed. On a desktop computer this function is achieved primarily via the hard disk, a huge non-volatile store of data. In an embedded system it is achieved using non-volatile semiconductor memory. It is a greater challenge to make non-volatile memory, and sophisticated semiconductor technology is applied. Generally, this type of memory has been more difficult to write to electrically, for example in terms of time or power taken, or complexity of the writing process. Non-volatile memory is used for holding the computer program and for historical reasons has commonly been called ROM (Read-Only Memory). A more descriptive name is ‘program memory’.

With the very sophisticated memory technology that is now available, we will see that the division of function between these two memory categories is becoming increasingly blurred. We return to the issue of memory technology and its applications in Chapter 2.
1.3.4 Organising memory

To interact with memory, there must be two types of number moved around: the address of the memory location required and the actual data that belongs in the location. These are connected in two sets of interconnections, called the address bus and the data bus. We must ensure that the data bus and address bus (or a subset of it) reach every memory area.

A simple way of meeting the need just described is shown in Figure 1.7(a). It is called the Von Neumann structure or architecture, after its inventor. The computer has just one address bus and one data bus, and the same address and data buses serve both program and data memories. The input/output may also be interconnected in this way and made to behave like memory as far as the CPU is concerned.

An alternative to the Von Neumann structure is seen in Figure 1.7(b). Every memory area gets its own address bus and its own data bus. Because this structure was invented in the university of the same name, this is called a Harvard structure.

The Von Neumann structure is simple and logical, and gives a certain type of flexibility. The addressable memory area can be divided up in any way between program memory and data memory. However, it suffers from two disadvantages. One is that it is a ‘one size fits all’ approach. It uses the same data bus for all areas of memory, even if one area deals with large words and another deals with small. It also has the problem of all things that are shared. If one person is using it, another can’t. Therefore, if the CPU is accessing program memory, then data memory must be idle and vice versa.

In the Harvard approach we get greater flexibility in bus size, but pay for it with a little more complexity. With program memory and data memory each having their own address and data buses, each can be a different size, appropriate to their needs, and data and program can be accessed simultaneously. On the minus side, the Harvard structure reinforces the distinction.
between program and data memory, even when this distinction is not wanted. This disad-
vantage may be experienced, for example, when data is stored in program memory as a table, 
but is actually needed in the data domain.

1.4 Microprocessors and microcontrollers

1.4.1 Microprocessors

The first microprocessors appeared in the 1970s. These were amazing devices, which for the 
first time put a computer CPU onto a single IC. For the first time, significant processing power 
was available at rather low cost, in a comparatively small space. At first, all other functions, 
like memory and input/output interfacing, were outside the microprocessor, and a working 
system still had to be made of a good number of ICs. Gradually, the microprocessor became 
more self-contained, with the possibility, for example, of including different memory types on 
the same chip as the CPU. At the same time, the CPU was becoming more powerful and faster, 
and moved rapidly from 8-bit to 16- and 32-bit devices. The development of the micropro-
cessor led very directly to applications like the personal computer.

1.4.2 Microcontrollers

While people quickly recognised and exploited the computing power of the microprocessor, 
they also saw another use for them, and that was in control. Designers started putting micro-
processors into all sorts of products that had nothing to do with computing, like the fridge or the 
car door that we have just seen. Here the need was not necessarily for high computational power, 
huge quantities of memory, or very high speed. A special category of microprocessor emerged 
that was intended for control activities, not for crunching big numbers. After a while this type of 
microprocessor gained an identity of its own, and became called a ‘microcontroller’. The 
microcontroller took over the role of the embedded computer in embedded systems.

So what distinguishes a microcontroller from a microprocessor? Like a microprocessor, 
a microcontroller needs to be able to compute, although not necessarily with big numbers. But it 
has other needs as well. Primarily, it must have excellent input/output capability, for example so 
that it can interface directly with the ins and outs of the fridge or the car door. Because many 
embedded systems are both size- and cost-conscious, the microcontroller must be small, self-
contained and low cost. Further, it will not sit in the nice controlled environment that a conven-
tional computer might expect. No, the microcontroller may need to put up with the harsh condi-
tions of the industrial or motor car environment, and be able to operate in extremes of temperature.

A generic view of a microcontroller is shown in Figure 1.8. Essentially, it contains a simple 
microprocessor core, along with all necessary data and program memory. To this it adds all the 
peripherals that allow it to do the interfacing it needs to do. These may include digital and
analog input and output, or counting and timing elements. Other more sophisticated functions are also available, which you will encounter later in the book. Like any electronic circuit the microcontroller needs to be powered, and needs a clock signal (which in some controllers is generated internally) to drive the internal logic circuits.

1.4.3 Microcontroller families

There are thousands of different microcontroller types in the world today, made by numerous different manufacturers. All reflect in one way or another the block diagram of Figure 1.8. A manufacturer builds a microcontroller ‘family’ around a fixed microprocessor core. Different family members are created by using the same core, including with it different combinations of peripherals and different memory sizes. This is shown symbolically in Figure 1.9. This manufacturer has three microcontroller families, each with its own core. One core might be 8-bit with limited power, another 16-bit and another a sophisticated 32-bit machine. To each core are added different combinations of peripherals and memory size, to make a number of family members. Because the core is fixed for all members of one family, the instruction set is fixed and users have little difficulty in moving from one family member to another.

While Figure 1.9 suggests only a few members of each family, in practice this is not the case; there can be more than 100 microcontrollers in any one family, each one with slightly different capabilities and some targeted at very specific applications.

1.4.4 Microcontroller packaging and appearance

Integrated circuits are made in a number of different forms, usually using plastic or ceramic as the packaging material. Interconnection with the outside world is provided by the pins on the
package. Where possible microcontrollers should be made as physically small as possible, so it is worth asking: what determines the size? Interestingly, it is not usually the size of the IC chip itself, in a conventional microcontroller, which determines the overall size. Instead, this is set by the number of interconnection pins provided on the IC and their spacing.

It is worth, therefore, pausing to consider what these pins carry in a microcontroller. The point has been made that a microcontroller is usually input-/output-intensive. It is reasonable then to assume that a good number of pins will be used for input/output. Power must also be supplied and an earth connection made. It is reasonable to assume for the sort of systems we will be looking at that the microcontroller has all the memory it needs on-chip. Therefore, it will not require the huge number of pins that earlier microprocessors needed, simply for connecting external data and address buses. It will, however, be necessary to provide pin interconnections to transfer program information into the memory and possibly provide extra power for the programming process. There is then usually a need to connect a clock signal, a reset and possibly some interrupt inputs.

Figure 1.10, which shows a selection of microprocessors and microcontrollers, demonstrates the stunning diversity of package and size that is available. On the far right, the massive
(and far from recent) 64-pin Motorola 68000 dwarfs almost everything else. Its package is a dual-in-line package (DIP), with its pins arranged in two rows along the longer sides of the IC, the pin spacing being 0.1 inches. Because the 68000 depends on external memory, many of its pins are committed to data and address bus functions, which forces the large size. Second from right is the comparatively recent 40-pin PIC 16F877. While this looks similar to the 68000, it actually makes very different use of its pins. With its on-chip program and data memory it has no need for external data or address buses. Its high pin count is now put to good use, allowing a high number of digital inputs/outputs and other lines. In the middle is the 52-pin Motorola 68HC705. This is in a square ceramic package, windowed to allow the on-chip EPROM (Erasable Programmable Read-Only Memory) to be erased. The pin spacing here is 0.05 inches, so the overall IC size is considerably more compact than the 68000, even though the pin count is still high. To the left of this is a 28-pin PIC 16C72. Again, this has EPROM program memory and thus is also in a windowed ceramic DIP package. On the far left is the tiny 8-pin surface-mounted PIC 12F508 and to the right of this is an 18-pin PIC 16F84A.

1.5 Microchip and the PIC microcontroller

1.5.1 Background

The PIC was originally a design of the company General Instruments. It was intended for simple control applications, hence the name – Peripheral Interface Controller. In the late 1970s General Instruments produced the PIC 1650 and 1655 processors. Although the design was comparatively crude and unorthodox, it was completely stand-alone, and contained some important and forward-looking features. The simple CPU was a RISC structure, with a single Working register and just 30 instructions. The output pins could source or sink much more current than most other microprocessors of the time. Already the trademark characteristics of the PIC were emerging – simplicity, stand-alone, high speed and low cost.
General Instruments sold off its semiconductor division to a group of venture capitalists, who must have realised the immense potential of these odd little devices. Throughout the 1990s the range of available PIC microcontrollers grew, and as they did they gradually overtook many of their better-established competitors. In many cases PIC microcontrollers could run faster, needed a simpler chip-set and were quicker to prototype with than their competitors. Despite the huge advances that were made, however, it was still possible to see features of the old General Instruments microcontroller. Unlike many competitors, Microchip made their development tools simple and low-cost or free. Moreover, Microchip stayed for a long while firmly entrenched in the 8-bit world. It has only been in the past few years that they have branched out beyond 8-bit devices. This book remains primarily concerned with the 8-bit PIC devices, as these provide such a useful entry into the world of embedded systems. The 16- and 32-bit Microchip devices will, however, be surveyed in Chapter 21.

1.5.2 PIC 8-bit microcontrollers today

Without looking any wider than the range of 8-bit PIC microcontrollers today, anyone can be forgiven for a sense of bewilderment. There are hundreds of different devices, offered in different packages, for different applications. Let us therefore try to identify the characteristics that all of these have in common. At the time of writing, all 8-bit PIC microcontrollers are low-cost, self-contained, pipelined, RISC, use the Harvard structure, have a single accumulator (the Working, or W, register), with a fixed reset vector.

Today, Microchip offer 8-bit microcontrollers with four different prefixes, 10-, 12-, 16-, and 18-, for example 10F200, or 18F242. In this book we shall call each of these a ‘Series’, for example ‘12 Series’, ‘16 Series’, ‘18 Series’. A 17 Series has been discontinued; a few are still sold, but most will only be found in legacy systems. Each Series is identified by the first two digits of the device code. The alphabetic character that follows gives some indication of the technology used. The ‘C’ insert implies CMOS technology, where CMOS stands for Complementary Metal Oxide Semiconductor, the leading semiconductor technology for implementing low-power logic systems. The ‘F’ insert indicates incorporation of Flash memory technology (still using CMOS as the core technology). An ‘A’ after the number indicates a technological upgrade on the first issue device. An ‘X’ indicates that a certain digit can take a number of values, the one taken being unimportant to the overall number quoted. For example, the 16C84 was the first of its kind. It was later reissued as the 16F84, incorporating Flash memory technology. It was then reissued as the 16F84A, with certain further technological upgrades.

While it is reasonable to expect that each Series defines a distinct architecture, in fact it is more useful to classify them into three distinct groups, using Microchip terminology, as shown in Table 1.2. In this book we shall refer to each of these groups as ‘families’. What complicates the picture, as the table shows, is that in some cases microcontrollers of one Series can fall into more than one family. For example some 12 Series microcontrollers are baseline, others are
Following the pattern of Figure 1.9, every member of any one family shares the same core architecture and instruction set. The processing power is defined to some extent by the parameters quoted, for example the instruction word size, and the number of instructions. It is possible to see clear evolution from one family to the next, so knowledge of one readily leads to knowledge of another. The families will be described in further detail below.

**The baseline family of PIC microcontrollers**

The baseline PIC microcontroller family represents the most direct descendant of the General Instruments ancestors, and displays the core features of the original PIC design. The first Microchip baseline microcontrollers were coded 16C5X, following the General Instruments 1650 and 1655 numbering. Now, however, there are also 10 and 12 Series microcontrollers which fall into this category. With only a two-level stack and no interrupts, there are real limits to the program and hardware complexity that can be developed. For example, without interrupts there is restriction on the type of on-chip peripheral that can be included, as most peripherals use interrupts to enhance their interface with the CPU.

Baseline devices are ideal for really tiny applications, being packaged in small ICs (right down to only six pins, for example). Despite their small size and simple architecture, baseline microcontrollers carry some interesting peripherals, including analog-to-digital converters and EEPROM (Electrically Erasable Programmable Read-Only Memory) data memory.

Baseline devices include all of the 10 Series, and some of the 12 Series. There is strong interest in this end of the size range, and further additions to the family can be expected.

**The PIC mid-range family**

The mid-range family contains several simple but important developments, when compared to the baseline devices. Interrupts (albeit with a single interrupt vector) are introduced and the stack size is increased. The instruction set is a slight extension of the baseline set.
The introduction of interrupts allows interfacing both with more sophisticated peripherals and with larger numbers of peripherals.

Mid-range devices include all of the 16 Series except those coded 16C5XX or 16F5XX, and some of the 12 Series. A very wide range has been developed, with many different peripherals and technical enhancements. The larger devices, with multiple peripherals and significant on-chip memory, are both powerful and versatile.

The high-performance family

In this family Microchip has come to grips with some of the issues of sophisticated processors. The instruction set is significantly increased, now to 75 instructions, and is designed to facilitate use of the C programming language. In certain versions there is also an ‘extended’ instruction set, with a further small set of instructions. There are two interrupt vectors, which can be prioritised.

The high-performance family is made up only of 18 Series microcontrollers. It is a powerful family and new members are continuously being added to the range.

1.6 An introduction to PIC microcontrollers using the Baseline Series

As the simplest of the PIC microcontroller types, this is a useful family with which to introduce the range. The features identified here will be recognisable in the more advanced PIC microcontrollers, where they appear alongside the more advanced features that have been added.

We will look at the PIC 12F508/509, the pin connection diagram of which is shown in Figure 1.11. The only difference between the 508 and 509 is that the latter has slightly larger
program and data memories. Most (if not all) labels on the pins in the diagram may initially make no sense – don’t worry; their meanings will emerge.

The staggeringly small size of this microcontroller is reinforced in Figure 1.12. While the 12F508 has been chosen as a simple microcontroller for introductory purposes, we also need to recognise that we are also almost looking at a conjuring trick. Remember that it has been said earlier that a microcontroller should be input-/output-intensive. Then consider: how can a microcontroller be useful if it has only eight pins interconnecting with the outside world? We will attempt to answer this question as we look at the microcontroller’s architecture.

1.6.1 The architecture of the 12F508

The annotated block diagram of the 12F508 appears in Figure 1.13. This may be the first Microchip diagram that you have ever looked at. Don’t worry if it initially appears complex – we will aim to break it into digestible pieces.

Let’s start by finding the microcontroller essentials identified in Figure 1.8: the core (containing the CPU), program memory, data memory (or RAM), data paths and any peripherals. We should be able to relate some of these features to the microcontroller pins of Figure 1.11.

The CPU, enclosed in a dotted line bottom right, is made up essentially of the ALU, the Working register (W Reg) and the Status register. This register carries a number of bits that give information on the outcome of the instruction most recently carried out. A multiplexer (MUX) selects from two sources which data is presented to the ALU.

The data memory is just 25 bytes for the 508 or 41 for the 509. Notice that Microchip call the RAM memory locations ‘file registers’ or elsewhere just ‘registers’. Program memory appears top left, with 512 12-bit words for the 12F508 or 1024 for the 509.

A distinctive feature of the PIC architecture is that it is Harvard structure, as discussed above. We should therefore be able to find two address buses (one for program memory, and the other
for data memory and all peripherals) and two data buses (again, one for program memory, and one for data memory and peripherals). The easiest to find is the data bus for data memory and peripherals. This is simply labelled ‘data bus’ and is seen to the right of the diagram. It is 8-bit, and primarily serves the data memory, the General-Purpose Input/Output (GPIO) and the ‘Timer 0’ peripheral. The address bus for data memory is labelled ‘RAM Addr’ and feeds into the RAM data memory. It is derived from the address multiplexer (‘Addr MUX’), which selects the address from one of two sources.

The program address bus arises from the Program Counter and goes only to the program memory, as shown. It is 12-bit, and hence can address \(2^{12}\) memory locations, or 4096 locations. As the program memory itself is given as only 512 or 1024 words, we recognise that the address bus is larger than necessary for this memory size. Coming from the program memory
we see the 12-bit ‘Program bus’. This carries the instruction words from the memory to the ‘Instruction register’.

It is interesting to track the way the instruction word from the program memory is divided up. As this microcontroller is a RISC computer, each instruction word must carry not only the instruction code itself, but also any address or data information needed. In the diagram the Instruction register receives the instruction word and then starts the process of dividing this up into its component parts. Depending on the instruction itself, five bits of the instruction word may carry address information and hence be sent down the ‘Direct Addr’ bus to the address multiplexer (‘Addr MUX’). Eight bits of the instruction word may carry a data byte that is to be used as literal data for the execution of that instruction. This goes to the multiplexer (‘MUX’), which feeds into the ALU. Finally, there is the instruction data itself, which feeds into the ‘Instruction Decode and Control’ unit.

This microcontroller has only two on-chip peripheral devices, a Timer (‘Timer 0’) and the General-Purpose Input/Output port, with pins GP0 to GP5. The IC pins themselves appear in the block diagram as squares with crosses inside. Each of these pins is dual or triple function, so each has a second function identified in the diagram. We do not need to understand now, what each of these is, but we soon will.

Towards the bottom left of the diagram are a number of functions relating to the clock oscillator, power supply and reset. Power supply and ground are connected via pins VDD and VSS respectively. A ‘Power-on Reset’ function detects when power is applied and holds the microcontroller in a Reset condition while the power supply stabilises. The MCLR input can be used to place the CPU in a Reset condition and to force the program to start again. An internal clock oscillator (‘Internal RC OSC’) is provided so that no external pins whatsoever need be committed to this function. External oscillator connections can, however, be made, using input/output pins GP4 and GP5. The oscillator signal is conditioned for use through the microcontroller in the ‘Timing Generation’ unit. The ‘Watchdog Timer’ is a safety feature, used to force a reset in the processor if it crashes.

Having worked through this section, it should be possible for you to appreciate that the diagram of Figure 1.13 is a direct embodiment of the generic microcontroller shown in Figure 1.8. While the detail at this stage is incomplete, it will fall into place in the coming chapters.

**Summary**

- An embedded system is a product that has one or more computers embedded within it, which primarily exercise a control function.

- The embedded computer is usually a microcontroller: a microprocessor adapted for embedded control applications.
Microcontrollers are designed according to accepted electronic and computer principles, and are fundamentally made up of microprocessor core, memory and peripherals; it is important to be able to recognise their principal features.

Microchip offers a wide range of microcontrollers, divided into a number of different families. Each family has identical (or very similar) central architecture and instruction sets. However, common features also appear across all their microcontrollers, and knowledge of one family can lead with ease to knowledge of another.

The Microchip 12F508 is a good microcontroller with which to introduce a range of features of microcontrollers in general and of PIC microcontrollers in particular.

References

1.2 Website of Microchip Technology Inc.: www.microchip.com

Questions and exercises

1. List five possible embedded systems in each of the following: a child’s bedroom/playroom, the kitchen, and the office.

2. Consider a domestic washing machine. Sketch its control system as a block diagram in a similar manner to Fig. 1.5, identifying as best you can its sensors and actuators. Assume it is controlled by a single microcontroller. (Note: the main point of this exercise is to visualise a product in terms of an embedded system, not to achieve technical accuracy.)

3. Repeat exercise 2 for a desktop printer.

4. An application is to use either the 12F508 or the 12F509. It must retain a scratchpad area of memory of 12 bytes, and also be able to store in RAM three variables of 4 bytes each, and 36 single bits of data. How many bytes does this amount to, and can the smaller device be used?

5. Access the 12F508/9 data sheet from the Microchip web site (Ref. 1.2), and by reading the first few pages answer the following questions for the 12F508:
   (a) What is the clock frequency range?
   (b) What is the frequency of the internal oscillator?
   (c) What is the operating voltage range?
   (d) What is the technology of the program memory?
(e) What support tools are available for program development?

(f) What types of operation is the ALU capable of?

(g) What are claimed to be the key advantages of this microcontroller?

(h) What bits in the Status register may be affected by the execution of an instruction?