1

Introduction to Microcontrollers

Digital systems are designed to store, process, and communicate information in digital form. They are found in a wide range of applications including process control, communication systems, digital instruments, and consumer products. A digital computer, more commonly known simply as a *computer*, is an example of a typical digital system.

A computer manipulates information in digital, or more precisely, binary form. A *binary number* has only two discrete values: zero or one. Each discrete value is represented by the OFF or ON status of an electronic switch called a *transistor*. All computers understand only binary numbers. Any decimal number (base 10, with 10 digits from 0 to 9) can be represented by a binary number (base 2, with digits 0 and 1).

The basic blocks of a computer are the central processing unit (CPU), the memory, and the input/output (I/O). The CPU of a computer is basically the same as the brain of a human being; so, computer memory is conceptually similar to human memory. Asking a question to a human being is analogous to entering a program into a computer using an input device such as a keyboard, and a person answering a question is similar in concept to outputting the program result to a computer output device such as a printer. The main difference is that human beings can think independently, whereas computers can only answer questions for which they are programmed. Computer *hardware* includes components such as memory, CPU, transistors, nuts, and bolts. Programs can perform a specific task, such as addition if the computer has an electronic circuit capable of adding two numbers. Programmers cannot change these electronic circuits but can perform tasks on them using instructions.

Computer *software* consists of a collection of programs. Note that programs contain instructions and data for performing a specific task. All programs, written using a programming language (e.g., C or assembly language), must be translated into binary prior to execution by a computer because the computer understands only binary numbers. Therefore, a translator is necessary to convert such a program into binary. This is achieved using a translator program called a *compiler* (for C) or *assembler* (for assembly). Programs in binary form containing 0's and 1's are then stored in the computer memory for execution.

Due to advances in semiconductor technology, it is possible to fabricate a CPU on a single chip. The result is a *microprocessor*. Both metal-oxide semiconductor (MOS) and bipolar technologies are used in the fabrication process. The CPU can be placed on a single chip when MOS technology is used. However, several chips are required with bipolar technology. At present, HCMOS (high-speed complementary MOS) or BICMOS (combination of bipolar and HCMOS) technology is normally used to fabricate a microprocessor on a single chip. Along with the microprocessor chip,

Microcontroller Theory and Applications with the PIC18F, Third Edition. M. Rafiquzzaman.

Companion website: www.wiley.com/go/Microcontroller_Theory_and_Applications_with_the_PIC18F/3e

appropriate memory and I/O chips can be used to design a *microcomputer*. The pins on the microprocessor chip are connected to the proper pins on the memory and I/O chips using wires to build a microcomputer. These wires are called the *system bus* and carry address, data, and control signals. In the past, some manufacturers designed a complete microcomputer (CPU, memory, and I/O) on a single chip with limited capabilities. Single-chip microcomputers such as the Intel 8048 were used in a wide range of industrial and home applications.

Microcontrollers evolved from single-chip microcomputers. Microcontrollers are normally used for dedicated applications such as automotive systems, home appliances, and home entertainment systems. Typical microcontrollers include a CPU, memory, I/O, along with certain peripheral functions such as timers, and Analog-to-Digital Converter (ADC) all in a single chip. Microchip Technology's PIC (Peripheral Interface Controller) is an example of a typical microcontroller.

Note that an ADC converts a DC voltage into a binary number. For example, an 8-bit ADC chip will convert a DC voltage, between 0 V and 5 V, into an 8-bit binary number. This means that the lowest voltage, 0 V at the ADC input will be converted to an 8-bit binary 00000000 at the ADC output; on the other hand, the highest voltage, 5 V at the ADC input, will be converted to an 8-bit binary number 11111111 (255 in decimal). Other voltages between 0 V and 5 V will be converted accordingly. Since microcontrollers only understand binary numbers, these binary values can be inputted into the microcontrollers via programming to perform meaningful applications. The example depicted in Figure 1.1 illustrates this.

To put microcontrollers into perspective, it is important to explore a simple application. For example, consider the microcontroller-based dedicated controller shown in Figure 1.1. Suppose that it is necessary to maintain the temperature of a furnace to a desired level to maintain the quality of a product. Assume that the designer has decided to control this temperature by adjusting the fuel. This can be accomplished using a typical microcontroller such as the PIC18F along with the interfacing components as follows.

Temperature is an analog (continuous) signal. It can be measured by a temperature-sensing (measuring) device such as a thermocouple. The thermocouple provides the measurement in millivolts (mV) equivalent to the temperature.

Since microcontrollers only understand binary numbers (0's and 1's), each analog mV signal must be converted to a binary number using the microcontroller's on-chip ADC. Note that the PIC18F contains on-chip ADC. However, the PIC18F does not include on-chip digital-to-analog converter (DAC). An external DAC chip can be interfaced to the PIC18F.

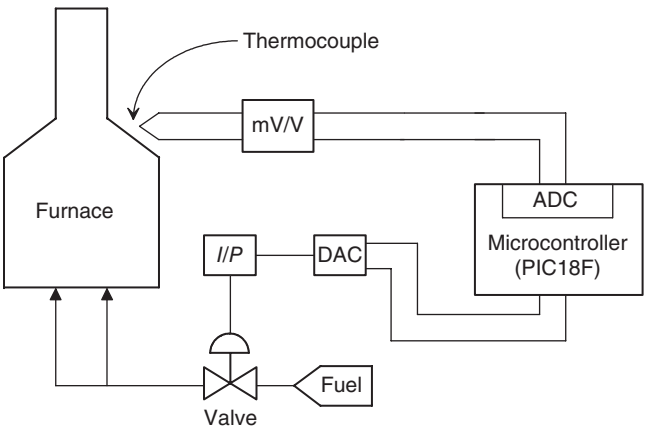


Figure 1.1 Furnace Temperature Control.

First, the millivolt signal is amplified by an mV/V amplifier to make the signal compatible for ADC input. A microcontroller such as the PIC18F can be programmed to solve an equation with the furnace temperature as an input. This equation compares the temperature measured with the temperature desired which can be entered into the microcontroller via programming. The output of this equation will provide the proper opening and closing of the fuel valve to maintain the appropriate temperature. Since this output is computed by the microcontroller, it is a binary number. This binary output must be converted into an analog current or voltage signal.

The DAC chip inputs this binary number and converts it into a current (I). This signal is then outputted to the current/pneumatic (I/P) transducer for opening or closing the fuel input valve by air pressure to adjust the fuel to the furnace. The desired furnace temperature can thus be achieved. Note that a transducer converts one form of energy (electrical current in this case) to another form (air pressure in this example).

Microcontrollers are extensively used in the Internet of Things (IoT), artificial intelligence (AI), and robotics.

IoT refers to a network of physical objects that can communicate and exchange data over the Internet. Microcontrollers can enable IoT by providing logic, memory, and connectivity for these objects. Typical objects include smart home devices such as temperature controllers, and automotive applications including cruise control, ignition systems, and radiator fans. To use microcontrollers for IoT, one needs to select a microcontroller that supports Wi-Fi and Bluetooth.

AI refers to the ability of machines to perform tasks that require human intelligence such as recognition, learning, reasoning, and decision-making. Microcontrollers can enable AI by running algorithms and models that can process data from sensors, cameras, microphones, and other sources. To use microcontrollers for AI, one needs to choose a device that has enough processing power, memory, and speed to handle AI tasks such as image classification and speech recognition.

Robotics refers to the design, construction, and operation of machines that can perform tasks independently or with human guidance. Microcontrollers can enable robotics by controlling the motors, servos, sensors, and other components that make the robot's body and behavior. To use microcontrollers for robotics, one needs to choose a device that can handle multiple inputs and outputs along with pulse width modulation (PWM), analog, digital, or serial signals and program it with a robotics platform.

Next, we first define some basic terms associated with microcontrollers. Then, we will briefly describe the evolution of microcontrollers. Finally, a comparison of the basic features of popular microcontrollers along with an overview of embedded controllers will be included.

1.1 Explanation of Terms

Before we go on, it is necessary to understand some basic terms.

- *Address* is a pattern of 0's and 1's that represents a specific location in memory or a particular I/O device. An 8-bit microcontroller with 16 address bits for program memory can produce 2^{16} unique 16-bit patterns from 0000000000000000 to 1111111111111111, representing 65,536 different address combinations (addresses 0 to 65,535 in decimal). This means that the maximum size of the program memory of this microcontroller is 64k or 64 kB (64 kilobytes since $2^{16} = 2^6 \times 2^{10} = 64 \times 1\text{k} = 64\text{k}$). All programs stored in this program memory can be executed by the microcontroller's CPU. This memory is also called the "Main memory" or "Directly addressable memory."

- *Addressing mode* is the manner in which the microcontroller determines the operand (data) and destination addresses during execution of an instruction.
- *Arithmetic-logic unit* (ALU) is a digital circuit that performs arithmetic and logic operations on two n-bit numbers. The value of n for microcontrollers can be 8-bit or 16-bit or 32-bit. Typical operations performed by an ALU are addition, subtraction, ANDing, ORing, and comparison of two n-bit numbers. The size of the ALU defines the size of the microcontroller. For example, an 8-bit microcontroller such as Microchip's PIC18F contains an 8-bit ALU.
- *Big endian* convention is used to store a 16-bit number such as 16-bit data in two bytes of memory locations as follows: the low memory address stores the high byte while the high memory address stores the low byte. The NXP/Freescale/Motorola HC12 16-bit microcontroller follows the big-Endian format.
- *Bit* is an abbreviation for the term *binary digit*. A binary digit can have only two values, which are represented by the digits 0 and 1, whereas a decimal digit can have 10 values, represented by the digits 0 through 9. The bit values are easily implemented in electronic and magnetic media by two-state devices whose states portray of the binary digits either 0 or 1. Examples of such two-state devices are a transistor that is conducting or not conducting, a capacitor that is charged or discharged, and a magnetic material that is magnetized north to south or south to north.
- *Bit size* refers to the number of bits that can be processed simultaneously by the basic arithmetic unit of a microcontroller. Several bits taken as a group in this manner are called a *word*. For example, an 8-bit microcontroller can process an 8-bit word. An 8-bit word is referred to as a *byte* and a 4-bit word is known as a *nibble*.
- *Bus* consists of several wires that connect different elements inside a microcontroller. The wires in a bus can be grouped in terms of their functions. A microcontroller normally has an address bus, a data bus, and a control bus. Address bits are sent to memory or to an I/O (input/output) device on the *address bus*. Instructions from memory and data to/from memory or I/O devices normally travel on the *data bus*. Control signals such as the read/write are transmitted on the *control bus*. Buses such as the *data bus* are *bidirectional*; information that can be transmitted in either direction on the bus; but some buses such as the *address bus* are unidirectional (information that can travel only in one direction).
- *Clock* is analogous to human heartbeats. The microcontroller requires synchronization among its components, and this is provided by a *clock* or timing circuits.
- *Complex instruction set computer* (CISC) contains a large instruction set. NXP/ Freescale/ Motorola HC12 is a CISC-based microcontroller.
- *CPU* (central processing unit) contains several registers (fast memory elements), an ALU, and a control unit. Note that the control unit translates instructions and generates enable signals for appropriate hardware units inside the CPU to perform the task desired by an instruction.
- *EEPROM* or E^2PROM (electrically erasable programmable ROM) is nonvolatile. EEPROMs can be programmed without removing the chip from the socket. EEPROMs are called Read Most Memories (RMMs), because they have much slower write times than read times. Therefore, these memories are usually suited for applications when mostly reading rather than writing is performed. An example of EEPROM is the 2864 (8k × 8).
- *EPROM* (erasable programmable ROM) is nonvolatile. EPROMs can be programmed and erased. The EPROM chip must be removed from the socket for programming. This memory is erased by exposing the chip to ultraviolet light via a lid or window on the chip. Typical erase times vary between 10 and 30 minutes. The EPROM is programmed by inserting the chip into a socket of the EPROM programmer and providing proper addresses and voltage pulses at the appropriate pins of the chip. An example of EPROM is the 2764 (8k × 8).

- *Flash memory* is designed using a combination of EPROM and EEPROM technologies. Flash memory is nonvolatile and was invented by Toshiba in the mid-1980s. Flash memory can be programmed electrically while embedded on the board. One can change multiple bytes at a time. An example of flash memory is the Intel 28F020 (256k×8). Flash memory is typically used in cell phones, digital cameras, and microcontrollers for storing programs. For example, the PIC18F uses flash memory for program memory.
- *Harvard architecture* is a type of CPU architecture which uses separate program and data memory units along with separate buses for program and data. This means that these processors can execute programs and access data simultaneously. Processors designed with this architecture require four buses for program memory and data memory. These are one data bus for instructions, one address bus for addresses of instructions, one data bus for data, and one address bus for addresses of data. The sizes of the address and data buses for instructions may be different from the address and data buses for data. Several microcontrollers including the PIC18F are designed using the Harvard architecture. This is because it is inexpensive to implement these buses inside the chip since both program and data memories are internal to the chip. Note that the PIC18F contains a 21-bit address bus and a 16-bit data bus for program memory, and a 12-bit address bus and an 8-bit data bus for data memory.
- *Instruction set* of a microcontroller is a list of commands that the microcontroller is designed to execute. Typical instructions are ADD, SUBTRACT, and STORE. Individual instructions are coded as unique bit patterns which are recognized and executed by the microcontroller. If a microcontroller has three bits allocated to the representation of instructions, the microcontroller will recognize a maximum of 2^3, or eight, different instructions. The microcontroller will then have a maximum of eight instructions in its instruction set.
- *Little endian* convention is used to store a 16-bit number such as 16-bit data in two bytes of memory locations as follows: the low memory address stores the low byte while the high memory address stores the high byte. The PIC18F microcontroller follows the little-endian format.
- *Microcomputer* typically consists of a microprocessor (CPU) chip, input and output chips, and memory chips in which programs (instructions and data) are stored.
- *Microcontroller* is implemented in a single chip containing a CPU, memory, and IOP (I/O and peripherals). Note that a typical IOP contains an I/O unit, timers, ADC, and other peripheral functions (to be discussed later).
- *Microprocessor* is the CPU of a microcomputer contained in a single chip and must be interfaced with memory and I/O chips in order to function.
- *Pipelining* is a technique that overlaps instruction fetch (instruction read) with execution. This allows a microcontroller's processing operation to be broken down into several steps (dictated by the number of pipeline levels or stages) so that the individual step outputs can be handled by the microcontroller in parallel. Pipelining is often used to fetch the microcontroller's next instruction while executing the current instruction, which speeds up the overall operation of the microcontroller considerably. Microchip technology's PIC18F (8-bit microcontroller) uses a two-stage instruction pipeline in order to speed up instruction execution.
- *Program* contains instructions and data. Two conventions are used to store a 16-bit number such as 16-bit data in two bytes of memory locations. These are called little endian and big-endian byte ordering. In little endian convention, the low memory address stores the low byte while the high memory address stores the high byte. For example, the 16-bit hexadecimal number, 2050 will be stored as two bytes in two 16-bit locations (Hex 5000 and Hex 5001) as follows: Address 5000 will contain 50 while address 5001 will store 20. In big endian convention, on the other hand, the low memory address stores the high byte while the high memory address stores the

low byte. For example, the same 16-bit hexadecimal number, 2050 will be stored as 2 bytes in two 16-bit locations (Hex 5000 and Hex 5001) as follows: Address 5000 will contain 20 while address 5001 will store 50. NXP/Freescale/Motorola HC12 (16-bit microcontroller) follows big endian convention. Microchip PIC18F (8-bit microcontroller), on the other hand, follows the little-endian format.

- *Random-access memory* (RAM) is a read/write memory. RAM normally provides *volatile storage*, which means that its contents are lost if the power is turned off. There are two types of RAMs: static RAM (SRAM), and dynamic RAM (DRAM). *Static RAM* stores data in flip-flops. Therefore, this memory does not need to be refreshed. An example of SRAM is 6116 (2k×8). *Dynamic RAM*, on the other hand, stores data as charge in capacitors. That is, it can hold data for a few milliseconds. Hence, dynamic RAMs are refreshed typically by using refresh circuitry. Dynamic RAMs (DRAMs) are used in applications requiring large memory. DRAMs have higher densities than static RAMs (SRAMs). Typical examples of DRAMs are the 4464 (64k×4), 44,256 (256k×4), and 41,000 (1M×1). DRAMs are inexpensive compared to SRAMs, occupy less space, and dissipate less power than SRAMs.

  The PIC18F uses SRAM for data memory.
- *Read-only memory* (ROM) is a storage device whose contents can only be read. Its contents cannot be altered once programmed. A typical ROM is fabricated on a chip and can store, for example, 2048 eight-bit words, which can be accessed individually by presenting to it one of 2048 addresses. This ROM is referred to as a 2k by 8-bit ROM. 10110111 is an example of an 8-bit word that might be stored in one location in this memory. A ROM is a *nonvolatile storage* device, which means that its contents are retained in the case that power is turned off. Because of this characteristic, ROMs are used to store permanent programs such as keyboard monitor program. ROMs are programmed by the manufacturers.
- *Reduced instruction set computer* (RISC) contains a simple instruction set. Because of a reduced instruction set, the RISC microcontrollers need fewer transistors which enables a smaller die size of the integrated circuitry (IC). Thus, the RISC microcontrollers consume less power compared to CISC, and hence, are suitable for portable devices. Microchip's PIC18F is an RISC-based microcontroller.
- *Register* can be considered as volatile storage for a number of bits. These bits may be entered into the register simultaneously (in parallel) or sequentially (serially) from right to left or from left to right, 1 bit at a time. An 8-bit register storing the bits 11110000 is represented as follows:

1	1	1	1	0	0	0	0

- *von Neumann* (*Princeton) architecture* uses a single memory unit and the same bus for accessing both instructions and data. CPUs designed using this architecture are slower compared to Harvard architecture. Instructions and data cannot be accessed simultaneously because of the single bus. Typical microprocessors use this architecture. This is because memory units such as ROMs, EPROMs, and RAMs are external to the microprocessor. This will require almost half the number of wires on the motherboard since address and data pins for only two buses rather than four buses (Harvard architecture) are required. This is the reason Harvard architecture would be very expensive if utilized for designing microprocessors. Note that microcontrollers using Harvard architecture internally will have to use von Neumann architecture externally. Texas Instruments' MSP430 uses the von Neumann architecture. MSP430 contains the same memory for both program memory and data memory with a 16-bit address bus and a 16-bit data bus.

1.2 Microcontroller Data Types

In this section, we will discuss data types used by typical microcontrollers: unsigned and signed binary numbers, binary-coded decimal (BCD), ASCII (American Standard Code for Information Interchange), and EBCDIC (extended binary-coded decimal interchange code).

1.2.1 Unsigned and Signed Binary Numbers

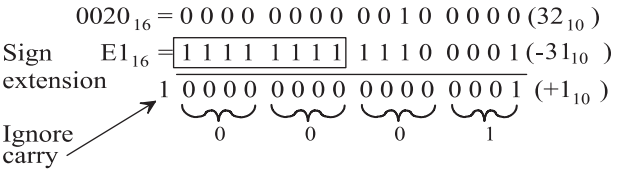
An *unsigned binary number* is always positive. Typical examples are your age or a memory address, which are always positive numbers. An 8-bit unsigned binary integer represents all numbers from 00_{16} through FF_{16} (0_{16} through 255_{10}).

A *signed binary number*, on the other hand, includes both positive and negative numbers. It is represented in the microcontroller in two's-complement form. For example, the decimal number +15 is represented in 8-bit two's-complement form as 00001111 (binary) or 0F (hexadecimal). The decimal number −15 can be represented in 8-bit two's-complement form as 11110001 (binary) or F1 (hexadecimal). Also, the most significant bit (MSB) of a signed number represents the sign of the number. For example, bit 7 of an 8-bit number, bit 15 of a 16-bit number, and bit 31 of a 32-bit number represent the signs of the respective numbers. A "0" at the MSB represents a positive number; a "1" at the MSB represents a negative number. Note that the 8-bit binary number 11111111 is 255_{10} when represented as an unsigned number. On the other hand, 11111111_2 is -1_{10} when represented as a signed number.

One can convert an unsigned binary number from lower to higher length using zero extension. For example, an 8-bit unsigned number FF (hex) can be converted to a 16-bit unsigned number 00FF (hex) by extending 0's to the upper byte of 00FF (hex). Both FF (hex) and 00FF (hex) have the same decimal value of 255. This is called zero extension. Zero extension is useful for performing arithmetic operations between two unsigned binary numbers of different lengths.

A signed binary number, on the other hand, can be converted from lower to higher length using sign extension. For example, an 8-bit signed number FF (hex) can be converted to a 16-bit signed number FFFF (hex) by extending the sign bit ("1" in this case) to the upper byte of FFFF (hex). Both FF (hex) and FFFF (hex) have the same decimal value of −1. Sign extension is useful for performing arithmetic operations between two signed binary numbers of different lengths.

Sign extension is useful when one wants to perform an arithmetic operation on two signed numbers of different lengths. For example, the 16-bit signed number 0020 (hex) can be added with the 8-bit signed number E1 (hex) by sign-extending E1 as follows:



An error (indicated by overflow in a microcontroller) may occur while performing two's complement arithmetic. The microcontroller automatically sets an overflow bit to 1 if the result of an arithmetic operation is too big for the microcontroller's maximum word size; otherwise, it is cleared to 0. For signed arithmetic operations such as addition, the overflow $V = C_f \oplus C_p$, where C_f is the final carry and C_p is the previous carry. This can be illustrated by the following examples.

Consider the following examples for two 8-bit signed numbers. Let C_f be the final carry (carry out of the most significant bit or sign bit) and C_p be the previous carry (carry out of bit 6 or seventh bit). We will show by means of numerical examples that as long as C_f and C_p are the same, the result is always correct. If, however, C_f and C_p are different, the result is incorrect and sets the overflow bit to 1. Now, consider the following cases.

Case 1: C_f and C_p are the same.

$$\begin{array}{rll} & 0\,0\,0\,0\,0\,1\,1\,0 & 06_{16} \\ & 0\,0\,0\,1\,0\,1\,0\,0 & +14_{16} \\ \hline 0 & 0\,0\,0\,1\,1\,0\,1\,0 & 1A_{16} \end{array}$$

$C_f = 0$, $C_p = 0$

$$\begin{array}{rll} & 0\,1\,1\,0\,1\,0\,0\,0 & 68_{16} \\ & 1\,1\,1\,1\,1\,0\,1\,0 & -06_{16} \\ \hline 1 & 0\,1\,1\,0\,0\,0\,1\,0 & 62_{16} \end{array}$$

$C_f = 1$, $C_p = 1$

Therefore, when C_f and C_p are either 0 or both 1, a correct answer is obtained.

Case 2: C_f and C_p are different.

$$\begin{array}{rll} & 0\,1\,0\,1\,1\,0\,0\,1 & 59_{16} \\ & 0\,1\,0\,0\,0\,1\,0\,1 & +45_{16} \\ \hline 0 & 1\,0\,0\,1\,1\,1\,1\,0 & -62_{16}\,? \end{array}$$

$C_f = 0$, $C_p = 1$

$C_f = 0$ and $C_p = 1$ give an incorrect answer because the result shows that the addition of two positive numbers is negative.

$C_f = 1$ and $C_p = 0$ provide an incorrect answer because the result indicates that the addition of two negative numbers is positive. Hence, the overflow bit (V) will be set to zero if the carries C_f and C_p are the same, that is, if both C_f and C_p are either 0 or 1. On the other hand, the overflow bit (V) will be set to 1 if carries C_f and C_p are different. The relationship among C_f, C_p, and V can be summarized in a truth table as follows:

Inputs		**Output**
C_f	**C_p**	**V**
0	0	0
0	1	1
1	0	1
1	1	0

From the truth table, overflow, $V = C_f \oplus C_p$.

Note that the symbol $\oplus$ represents exclusive-OR logic operation. Exclusive-OR means that when two inputs are the same (both one or both zero), the output is zero. On the other hand, if two inputs

are different, the output is one. The overflow can be considered as an output while C_f and C_p are the two inputs. The answer is incorrect when the overflow bit is set to 1; the answer is correct if the overflow bit is 0.

Also, typical 16- and 32-bit microprocessors such as NXP/Freescale/Motorola's 68000/68020 have separate unsigned and signed multiplication and division instructions as follows: MULU (multiply two unsigned numbers), MULS (multiply two signed numbers), DIVU (divide two unsigned numbers), and DIVS (divide two signed numbers). It is important for the programmer to understand clearly how to use these instructions.

For example, suppose that it is desired to compute $x^2/255$. If X is a signed 8-bit number, the programmer should use the MULS instruction to compute X * X which is always unsigned (the square of a number is always positive), and then use DIVU to compute $X^2/255$ (16-bit by 8-bit unsigned divide) since 255_{10} is positive. But if the programmer uses DIVS, both X * X and 255_{10} (FF_{16}) will be interpreted as signed numbers. If FF_{16} will be interpreted as -1_{10}, the result will be wrong. On the other hand, if X is an unsigned number, the programmer needs to use MULU and DIVU to compute $X^2/255$.

The PIC18F microcontroller includes unsigned multiplication instructions. However, the PIC18F does not provide any signed multiplication and division (both signed and unsigned) instructions. Hence, as shown in Chapter 7, these instructions can be achieved by writing subroutines using PIC18F instructions.

1.2.2 ASCII and EBCDIC Codes

If it is to be very useful, a microcontroller must be capable of handling nonnumeric information. In other words, a microcontroller must be able to recognize codes that represent numbers, letters, and special characters. These codes are classified as alphanumeric or character codes. A complete and adequate set of necessary characters includes the following:

- 26 lowercase letters
- 26 uppercase letters
- 10 numerical digits (0–9)
- Approximately 25 special characters, which include +, /, #, %, and others.

This totals 87 characters. To represent 87 characters with some type of binary code would require at least 7 bits. With 7 bits there are $2^7 = 128$ possible binary numbers; 87 of these combinations of 0 and 1 bits serve as the code groups representing the 87 different characters.

The two most common alphanumerical codes are the American Standard Code for Information Interchange (ASCII) and the extended binary-coded-decimal interchange code (EBCDIC). ASCII is typically used with microcontrollers; IBM uses EBCDIC code. Eight bits are used to represent characters, although 7 bits suffice because the eighth bit is frequently used to test for errors and is referred to as a *parity bit*. It can be set to 1 or 0 so that the number of 1 bits in the byte is always odd or even.

Note that decimal digits 0 through 9 are represented by 30_{16} through 39_{16} in ASCII. On the other hand, these decimal digits are represented by $F0_{16}$ through $F9_{16}$ in EBCDIC.

A microcontroller program is usually written for code conversion when input/output devices of different codes are connected to the microcontroller. For example, suppose that it is desired to enter the number 5 into a computer via an ASCII keyboard and to print this data on an EBCDIC printer. The ASCII keyboard will generate 35_{16} when the number 5 is pushed. The ASCII code 35_{16} for the decimal digit 5 enters the microcontroller and resides in the memory. To print the digit 5 on the EBCDIC printer, a program must be written that will convert the ASCII code 35_{16}

for 5 to its EBCDIC code, $F5_{16}$. The output of this program is $F5_{16}$. This would be inputted to the EBCDIC printer. Because the printer understands only EBCDIC codes, it inputs the EBCDIC code $F5_{16}$ and prints the digit 5. Note that EBCDIC code is obsolete; it is used here merely for illustrative purposes.

1.2.3 Unpacked and Packed Binary-Coded-Decimal Numbers

The 10 decimal digits 0 through 9 can be represented by their corresponding 4-bit binary numbers. The digits coded in this fashion are called *binary-coded-decimal digits* in 8421 code, or BCD digits. Two unpacked BCD bytes are usually packed into a byte to form *packed BCD*. For example, two unpacked BCD bytes 02_{16} and 05_{16} can be combined as a packed BCD byte 25_{16}.

Let us consider entering data decimal 24 via an ASCII keyboard into a microcontroller. Two keys (2 and 4) will be pushed on the ASCII keyboard. This will generate 32 and 34 (32 and 34 are ASCII codes in hexadecimal for 2 and 4, respectively) inside the microcontroller. A program can be written to convert these ASCII codes into unpacked BCD 02_{16} and 04_{16}. This data can be converted to packed BCD 24 or to binary. Unpacked BCD 02_{16} and 04_{16} can be converted into packed BCD 24 (00100100_2) by logically shifting 02_{16} four times to left to obtain 20_{16}, then logically ORing with 04_{16}. On the other hand, to convert unpacked BCD 02_{16} and 04_{16} into binary, one needs to multiply 02_{16} by 10 and then add 04_{16} to obtain 00011000_2 (the binary equivalent of 24).

Note that BCD correction (adding 6) is necessary for the following:

i) If the binary sum is greater than or equal to decimal 16 (this will generate a carry of one).
ii) If the binary sum is 1010 through 1111.

For example, consider adding packed BCD numbers 97 and 39:

	111←Intermediate Carries		
97	1001	0111	BCD for 97
+39	0011	1001	BCD for 39
136	1101	0000	Invalid sum
	+0110	+0110	add 6 for correction
$\underbrace{0001}_{1}$	$\underbrace{0011}_{3}$	$\underbrace{0110}_{6}$	← correct answer 136

Typical 32-bit microprocessors such as the NXP/Freescale/Motorola 68020 include PACK and UNPK instructions for converting an unpacked BCD number to its packed equivalent, and vice versa. The PIC18F microcontroller contains an instruction called DAW which provides the correct BCD result after binary addition of two packed BCD numbers.

Typical microprocessors such as the Intel Pentium include instructions to provide correct unpacked BCD after performing arithmetic operations in ASCII. The Pentium instruction AAA (ASCII adjust for addition) is such an instruction. The PIC18F does not provide such instructions.

1.3 Evolution of the Microcontroller

Intel Corporation is generally acknowledged as the company that introduced the first microprocessor successfully into the marketplace. Its first microprocessor, the 4004, was introduced in 1971 and evolved from a development effort while making a calculator chip set.

The 4004 microprocessor was the central component in the chip set, which was called the MCS-4. The other components in the set included a 4001 ROM, a 4002 RAM, and a 4003 shift register.

Shortly after Intel 4004 appeared in the commercial marketplace, three other general-purpose microprocessors were introduced: Rockwell International 4-bit PPS-4, Intel 8-bit 8008, and National Semiconductor 16-bit IMP-16. Other companies, such as General Electric, RCA, and Viatron, also made contributions to the development of the microprocessor prior to 1971.

The microprocessors introduced between 1971 and 1972 were the first-generation systems designed using PMOS technology. In 1973, second-generation microprocessors such as the Motorola 6800 and the Intel 8080 (8-bit microprocessors) were introduced. The second-generation microprocessors were designed using NMOS technology. This technology resulted in a significant increase in instruction execution speed over PMOS and higher chip densities. Since then, microprocessors have been fabricated using a variety of technologies and designs. NMOS microprocessors such as the Intel 8085, Zilog Z80, and Motorola 6800/6809 were introduced based on second-generation microprocessors. A third-generation HMOS microprocessor, introduced in 1978, is typically represented by Intel 8086 and Motorola 68000; both of them are 16-bit microprocessors.

During the 1980s, the fourth-generation HCMOS and BICMOS (a combination of bipolar and HCMOS) 32-bit microprocessors evolved. Intel introduced the first commercial 32-bit microprocessor and the problematic Intel 432, which was eventually discontinued. Since 1985, more 32-bit microprocessors have been introduced. These include Motorola's 68020, 68030, 68040, 68060, PowerPC, Intel's 80386, 80486, the Intel Pentium family, Core Duo, and Core2 Duo microprocessors.

The performance offered by the 32-bit microprocessor is more comparable to that of superminicomputers such as Digital Equipment Corporation's VAX11/750 and VAX11/780. Intel and Motorola also introduced RISC microprocessors which had simplified instruction sets: the Intel 80960 and Motorola 88100/PowerPC. Note that the purpose of RISC microprocessors is to maximize speed by reducing clock cycles per instruction. Almost all computations can be obtained from a simple instruction set. Note that, to enhance performance significantly, Intel Pentium Pro and other succeeding members of the Pentium family and Motorola 68060 are designed using a combination of RISC and CISC.

Single-chip microcomputers such as the Intel 8048 evolved during the 1980s. Soon afterward, based on the concept of single-chip microcomputers, Intel introduced the first 8-bit microcontroller: the Intel 8051 which uses Harvard architecture and is designed by using CISC. The 8051 contains a CPU, memory, I/O, ADC and DAC, timers, serial communication interface, all in a single chip. Microcontrollers became popular during the 1980s.

As mentioned before, typical microcontroller applications include automobile engine control systems, implantable medical devices, remote controls, office machines, appliances, power tools, toys, and other embedded systems. Although Intel developed the first microcontroller, Intel is not that active in manufacturing contemporary microcontrollers. Typical microcontroller manufacturers include Microchip Technology, Texas Instruments, Arduino, and ARM.

Microchip Technology manufactures PIC18F 8-bit microcontroller chips. The PIC18F is a very basic microcontroller chip. The PIC18F family is popular with both industrial developers and hobbyists due to their low-cost, wide availability, and availability of low-cost or free development tools. PIC18F microcontrollers are typically used for industrial automation, home automation, embedded systems, robotics, instrumentation, control systems, education, and training.

Arduino (an Italian-based company) offers licensing for microcontroller manufacturers to provide single boards with their microcontrollers. Arduino-based microcontrollers are very popular in industries because they are easy to use and have strong features and tools that make them attractive

for many applications. An example of a simple Arduino board consists of a Microchip/Atmel 8-bit microcontroller, large amount of flash memory, serial I/O, and other features. Arduino is extensively used in home automation systems. Arduino boards, combined with sensors, actuators, and wireless connectivity, allow homeowners to control various aspects including lighting and temperature control, and automated security systems.

ARM (Advanced RISC Machine) is a UK-based company that makes RISC-based designs and licenses to chip manufacturers. ARM does not actually manufacture any chip themselves. Instead, they design the architecture and cores of a processor, and then partner companies such as Apple, ST Microelectronics, and NXP who license those designs to build their own chips. ARM offers 32-bit and 64-bit CPU core. Due to their low costs, low-power consumption, and low heat generation, ARM processors are useful for low-powered devices such as smartphones, laptops, tablet computers, and embedded systems.

The PIC18F family continues to be popular as evidenced by the following Google quote "According to LinkedIn, PIC is not obsolete, but rather a versatile and reliable platform that can suit different needs preferences. PIC devices are popular with both industrial developers and hobbyists because of their low cost, wide availability, large user base, and re-programmable flash memory capability. They also have broad applications for industrial applications for industrial purposes because of low power consumption, ease of supporting software and hardware tools, and high-performance ability." The PIC18F provides a basic understanding of a typical and simple microcontroller from chip level.

Eight-bit microcontrollers gained popularity over the last several years. These microcontrollers are not only small enough for many embedded applications but also powerful enough to allow a lot of complexity and flexibility in the design process of an embedded system. Several billion 8-bit microcontrollers were sold during the last decade. Several contemporary microcontroller manufacturers use RISC architecture, which provides a cost-effective approach. In addition, typical 8-bit microcontrollers such as PIC18F implemented several on-chip enhanced peripheral functions including PWM and flash memories. Note that NXP/Freescale/Motorola's popular 8-bit microcontroller HC11 does not have on-chip flash memory and PWM functions. The microcontroller can generate a periodic waveform with a specified duty cycle using PWM. Hence, the PWM function is a very desirable feature for applications such as automotive and motor control. These applications may include driving servo motors. In HC11, the timer section is used to generate PWM signals. However, NXP/Freescale implemented these features in the HC12 which is a 16-bit microcontroller.

Some of the other microcontrollers comparable to the Microchip Technology's PIC18F include Texas Instrument MSP 430, NXP/Freescale HC11/HC12, and Atmel ATtiny. Microchip Technology's first microcontroller, the PIC1650, was originally developed by General Instrument's Microelectronics Division in the late 1970s. The early models of PIC contained an on-chip ROM and erasable EPROMs for program storage. State-of-the-art models include flash memory for program storage. Note that flash memory can be erased electrically only in blocks. Since the PIC18F uses Harvard architecture, program and data memories are separate. The PIC18F uses flash memory for program memory. SRAM, on the other hand, is used for data memory. Note that "F" is included in the part number "PIC18F" to indicate that the chip contains flash memory.

Some of the different PIC models include the PIC18F family (8-bit microcontroller chips) and the PIC24F family (16-bit microcontroller chips). Microchip has introduced several different versions of the PIC18F microcontroller over the years. All members of the PIC18F family basically contain the same instruction set. However, certain features such as memory sizes, number of I/O ports, ADC channels, and PWM modules may vary from one version to another.

Microchip provides extensive software development support for the PIC18F family of microcontrollers. This software is known as MPLAB which includes assemblers, C compilers, and programmer/debugger hardware under the MPLAB and PICKit series.

Atmel developed the AVR family of microcontrollers in 1996. Note that AVR is not an acronym. AVR microcontrollers use Harvard architecture. Like the PIC18F, the AVR microcontrollers contain on-chip flash memory for program memory and SRAM is used as data memory. AVR microcontrollers include AVR 8- and 32-bit microcontrollers. Atmel also makes a microcontroller that uses Intel 8051 architecture. Furthermore, Atmel makes the ARM-based Cortex-M4 flash microcontrollers. Note that Microchip bought Atmel in 2016.

Also, Philips Semiconductors changed its name to NXP in 2006. Two of the popular 8- and 16-bit NXP/Freescale microcontrollers include 68HC11 (8-bit) and 68HC12 (16-bit). However, NXP manufacturer introduced several high-performance NXP low-power 32-bit microcontrollers in 2013. They are based on ARM Cortex core architecture. As an example, the NXP LPC54113 is a low-power microcontroller with an ARM Cortex-based CPU and contains 256 kB flash and 192 kB SRAM with a typical clock speed of 48 MHz.

Texas Instruments introduced the MSP 430 16-bit microcontroller during the late 90's. MSP430 is a RISC microcontroller based on von Neumann architecture. Texas Instruments announced its 32-bit microcontroller called the MSP 432 in 2015. MSP 432 is based on ARM Cortex core with ultra-low-power consumption.

In this book, a specific PIC18F chip such as the PIC18F4321 will be considered in detail. This is because Microchip's PIC18F is inexpensive and offers a simple instruction set with desirable on-chip features including ADC, PWM, and timers. In addition, Microchip provides user-friendly development support such as MPLAB and PICKit. This makes the PIC18F an excellent educational tool for thorough coverage in a first course on microcontrollers. Table 1.1 provides a comparison of some of the basic features of some of the typical microcontrollers comparable to the PIC18F.

Table 1.1 Comparison of Typical Features of Basic Microcontrollers.

	PIC18F	**MSP 430**	**HC12**	**AVR model ATtiny**
Manufacturer	Microchip Technology	Texas Instruments	NXP/Freescale /Motorola	Microchip/Atmel
Introduced	2000; the first PIC in 1989.	Late 1990s	1990	2003
Size	8-bit	16-bit	16-bit	8-bit
Architecture	Harvard	von Neumann	von Neumann	Modified Harvard
Design approach	RISC	RISC	CISC	RISC
On-chip flash memory	Yes	Yes	Yes	Yes. First to offer on-chip flash.
On-chip PWM	Yes	Yes	Yes	Yes
CPU clock	40-MHz (maximum)	16-MHz (maximum)	8-MHz (maximum)	20-MHz (maximum)
Total instructions	75	27	203	136
Total addressing modes	6	7	6	7

1.4 Embedded Controllers

Embedded microcontroller systems, also called embedded controllers, are designed to manage specific tasks. Once programmed, the embedded controllers can manage the functions of a wide variety of electronic products. In embedded applications, the microcontrollers are embedded (hidden) in the host system; their presence and operation are basically hidden from the host system.

Typical embedded control applications include office automation products such as copiers, laser products, fax machines, and consumer products including microwave ovens. Applications such as printers typically utilize a microcontroller. The RISC microcontrollers are ideal for these types of applications. Note that the personal computer (PC) interfaced with the printer is the host.

The microcontroller hidden (embedded) inside the printer is the "embedded controller." The purpose of the microcontroller, in this case, is to input data from the host and print it. Thus, an embedded controller performs only one task (printing in this case). A PC is normally connected to several peripherals such as a printer, a keyboard, a mouse, and a hard disk controller. Each one of these peripherals contains a microcontroller. Each microcontroller is programmed to execute a specific task desired by the peripheral. For example, the embedded microcontroller for the keyboard will perform keyboard functions for the PC including detecting a key actuation, debouncing it, and then decoding the key.

For large applications, an embedded controller contains three components: a microcontroller, an application software, and a real-time operating system (RTOS). RTOS is not needed for a small embedded system. For a large embedded system, the RTOS supervises the application programs and sets the rules during its execution.

RISC microcontrollers such as the PIC18F are well suited for applications including robotics, controls, instrumentation, and consumer electronics. The key features of the RISC microcontrollers that make them ideal for these applications are their relatively low level of integration in the chip, and instruction pipeline architecture. These characteristics result in low-power consumption, fast instruction execution, and fast recognition of interrupts.

Although simple and popular microcontrollers such as the PIC18F are considered ideal for many embedded applications, sometimes they might not be able to perform certain tasks. For example, applications such as laser printers require a high-performance microcontroller with on-chip floating-point hardware. NXP/Freescale microcontroller family with ARM's Cortex-M7 with on-chip floating-point hardware is ideal for these types of applications. Note that the PC interfaced with the laser printer is the host. The PIC18F will not be suitable for such an application since it does not provide floating-point instructions.