1 Digital tools

1.1 Introduction

In the past few years, enormous advances have been made in the cost, power, and ease of use of microcomputers and associated analog and digital circuits. It is now possible, with a relatively small expenditure, to purchase a microcomputer system that will take data, quickly analyze them, and display the results or control a process. This has been made possible by the development of technology that can fabricate millions of transistors, diodes, resistors, capacitors, and conductors on a single silicon **integrated circuit chip**.

Normally, the microcomputer is equipped with a number of standard items: the microprocessor chip and associated circuits, random-access memory chips, removable floppy and cartridge disk drives, magnetic hard disk drives, optical disk drives, keyboards, video display screens, serial interfaces, printers, and x–y entry devices such as the mouse, trackball, joystick, bitpad, and touch-sensitive display screen. However, data acquisition and control require additional components, such as digital and analog input/output (I/O) ports, and counters/timers. Analog input ports contain analog multiplexers, sample-and-hold (S/H) amplifiers, and analog-to-digital (A/D) converters. Analog output ports contain digital-to-analog (D/A) converters.

Even for designs requiring only a microprocessor and a few additional circuits, there are considerable advantages to using the resources of the microcomputer during the development stage. These include program code editors and compilers, an operating system for the storage and manipulation of code and data files, and ample random-access memory.

In this chapter, we discuss digital interfacing concepts used in microcomputer-based data-acquisition and control systems (Figure 1.1), including parallel and serial input/output ports, handshaking, and digital counters/timers. Analog tools (amplification and filtering) are treated in Chapter 2, digital-to-analog and analog-to-digital conversion and sampling in Chapter 3, and sensors and actuators in Chapter 4.

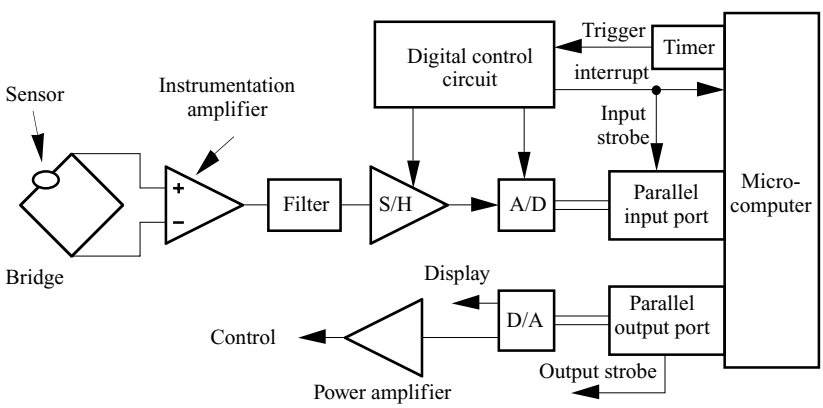

Figure 1.1 A microcomputer system interfaced to sensors and associated analog circuits for data acquisition, analysis, and control.

1.2 The microcomputer

In selecting a system for data acquisition and control, the **microcomputer** itself is a crucial component (Figure 1.2). The microcomputer is sufficiently small to fit on a laboratory bench (or desktop) and yet contains the following components:

1. The **microprocessor** is an integrated circuit that reads program instructions from memory and uses them to determine the sequence of actions that it performs. It is connected to memory and peripheral circuits by an address bus, a data bus, and control lines.

   These actions include reading data and instructions from memory, performing calculations, executing different instructions depending on the outcome of a calculation, printing data, and transferring data to and from peripheral devices such as hard disks. Microprocessors vary greatly in their speed and data-handling capability.
2. **Random-access memory (RAM)** usually consists of high-speed semiconductor memory chips that are used to store and retrieve program instructions and data. The highest data-acquisition speeds are achieved when external data are read directly into RAM, so the size of the RAM places a limit on the number of data values that can be sampled rapidly.
3. Common user interface devices are the keyboard, video display screen, printer, mouse, joystick, and trackball. Some systems provide voice input and synthesized speech output. The IEEE-1284 interface standard includes the standard parallel printer (SPP) port as well as other enhancements. The universal serial bus (USB) is the current standard for keyboards and pointing devices. For higher

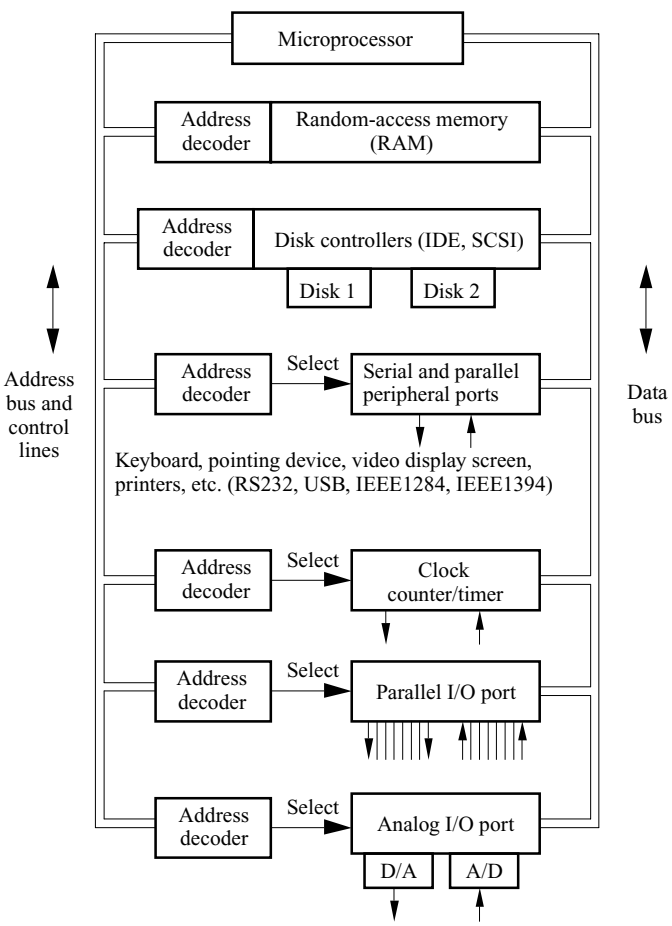

Figure 1.2 The microcomputer consists of a microprocessor that communicates with memory and input/output devices by address and data buses.

speed transfers (external hard drives, digital camcorders, HDTV), the IEEE-1394 standard (FireWire or i.Link) has recently been introduced.

4. **Magnetic disk memory** is used for the long-term storage of programs and data, and consists of one or more flat circular plates coated with a magnetic surface. Magnetic disk capacities range from 500 kbytes to 2 Mbytes for small removable floppy disks and from 1 to 20 Gbytes or more for hard disks. Access time consists of a fixed delay of tens of milliseconds (for the read/write head to locate the desired track) and a transfer time of typically 1 μs per 16-bit word.
5. **Optical disk memory** includes the CD-ROM and the DVD-ROM disks. The **CD-ROM** (compact disk-read-only memory) and DVD-ROM (digital versatile disk) drives use optical storage and retrieval technology that was developed for the music and entertainment industry. The capacity of the CD-ROM is over 600 Mbytes and about ten times larger for the DVD-ROM. Both are 12 cm in

diameter. Microcomputers and workstations are commonly shipped with a CD-ROM containing a back-up copy of the system software and on-line documentation, eliminating many floppy disks and thousands of pages of paper. CD-W (write once) and CD-RW (rewritable) and DVD-RAM (random-access) technology allows information to be written onto these disks.

6. The operating system permits the user to manipulate program and data files and supports a high-level compiled programming language (FORTRAN, Pascal, C, compiled BASIC, etc.).
7. A **compiler's** function is to translate a high-level language into microprocessor code that is able to:
   (i) perform numerical computations and conditional branching,
   (ii) communicate directly with a data-acquisition and control board or parallel I/O port (see below),
   (iii) read and write files to the disk.
   Additional useful features include:
   (i) a full range of scientific functions (sine, cosine, exp, log, etc.), and the ability to compute using floating-point representation, which can handle very small and very large numbers (for example, 80-bit extended precision can handle numbers from $\pm 10^{-4,932}$ to $\pm 10^{+4,932}$ with a precision of 19 decimal digits);
   (ii) the ability to write functions in assembly code for greater speed during data acquisition (some compilers permit intermixed assembly code and higher-level code);
   (iii) a built-in **editor** that displays lines causing compilation errors and permits immediate correction;
   (iv) a single command that compiles all changed program modules, links all necessary modules, and runs the result.
8. An analog input/output port (also called a data-acquisition and control circuit), with the required speed and number of A/D and D/A conversion circuits.
9. A parallel input/output port with sufficient speed, if item 8 is not available. In this case, it becomes necessary to design and build a data-acquisition circuit for connection to the parallel I/O port (this is demonstrated in Laboratory Exercise 9).
10. A counter/timer that can determine elapsed times to an accuracy of typically 1 μs, count input pulses, or produce output pulses of any desired width and period with an accuracy of typically 1 μs.

The microprocessor communicates with the other components of the microcomputer by an address bus, a data bus, and a number of control lines (Figure 1.2). The **address bus** allows the microprocessor to select particular components individually. Each component has a unique assigned **address** whether it is a RAM location, an I/O port register, or other peripheral circuit. An **address decoder** produces a **select** pulse whenever the assigned address appears on the address bus. For example, a 16-Mbit RAM chip has an internal address decoder with 24 input lines and 16 million select lines, one for each

memory bit that can be selected. The **data bus** is used to transmit data words to and from the microprocessor and its associated circuits.

Note: In some systems, memory locations and external devices are distinguished from each other by a special control bit. In others, a large block of memory address space is reserved for external devices.

Since many devices are attached to the data and address buses and at any instant only one can be sending data, control lines are used to indicate when the bus is busy, when a sending device requests use of the bus, when use is granted, etc. These details are beyond the scope of this book and are mentioned to outline the organization of the microcomputer.

Laboratory Exercise 1 is designed to familiarize the reader with the particular editor and compiler that will be used for the rest of the exercises as well as review 2's complement, hexadecimal, real, and integer interpretations of binary numbers.

1.3 Number systems

1.3.1 Binary number representations

Binary numbers can be interpreted in a variety of ways. Table 1.1 shows the interpretation of 8-bit binary patterns as unsigned decimal, hexadecimal, Gray, and 2's complement numbers. The 16-bit and 32-bit numbers are logical extensions.

A/D converters and counters/timers produce binary bit patterns that are to be interpreted as unsigned numbers. The binary sequence runs continuously from all bits = 0 to all bits = 1 and the leftmost bit is the most significant bit (**MSB**).

Angle and position encoders usually produce **Gray code** that runs from all bits = 0 to all bits = 1, but the binary sequence is not continuous because it has the special property that advancing from one number to the next involves changing the state of only one bit. Gray code is described further in the following section.

Binary numbers can also be represented in **hexadecimal** form (base 16) for efficient notation. Note that each 8-bit byte can be represented as two hexadecimal digits. Octal (base 8) is less frequently used.

Binary bit patterns can also be interpreted as signed numbers, to include negative numbers (<0) as well as 0 and positive numbers (>0). Some computers use signed binary representation, where the leftmost bit represents the sign. While this representation is closer to that of the printed page, it is seldom used in computers because arithmetic operations take longer due to the need to process the sign bit.

Most microcomputers use **2's complement representation** to deal more efficiently with negative and positive numbers. In 2's complement representation, the sign of a number is changed by complementing (reversing) all its bits and then adding one. This is called the **2's complement operation**. By using this operation, the subtraction process

Table 1.1 *Interpretations of 8-bit binary numbers*

Binary	Unsigned decimal	Hexadecimal	Gray	2's complement
0000 0000	0	00	0	0
0000 0001	1	01	1	1
0000 0010	2	02	3	2
0000 0011	3	03	2	3
0000 0100	4	04	7	4
0000 0101	5	05	6	5
0000 0110	6	06	4	6
0000 0111	7	07	5	7
0000 1000	8	08	15	8
0000 1001	9	09	14	9
0000 1010	10	0A	12	10
0000 1011	11	0B	13	11
0000 1100	12	0C	8	12
0000 1101	13	0D	9	13
0000 1110	14	0E	11	14
0000 1111	15	0F	10	15
0001 0000	16	10	31	16
...	...	...	...	...
0111 1110	126	7E	65	126
0111 1111	127	7F	64	127
1000 0000	128	80	192	−128
1000 0001	129	81	193	−127
...	...	...	...	...
1111 1110	254	FE	129	−2
1111 1111	255	FF	128	−1

$a - b$ can be performed by adding a to the 2's complement of b. For an 8-bit number, 2 is represented as binary 0000 0010 (hexadecimal 02) and −2 is represented as binary 1111 1110 (hexadecimal FE). For example, $5 - 2 = 3$ in 2's complement arithmetic is:

5	0000 0101	simply add, but ignore
−2	1111 1110	the most significant carry bit
3	0000 0011	

Note that in 2's complement notation, positive numbers have their MSB = 0 and negative numbers have their MSB = 1.

Warning: sign extension

As demonstrated in Laboratory Exercise 1, if the MSB of a number is zero, then conversion from 8 to 16 bits or from 16 to 32 bits occurs as expected. However, if the

Table 1.2 *Typical variable types, storage, and ranges of values*

Type	No. bits	Decimal digits	Range
Char	8		−128 to +127
Unsigned char*	8		0 to 255
Short	16		−32,768 to +32,767
Unsigned integer	16		0 to 65,535
Int and long	32		−2,147,483,648 to 2,147,483,647
Unsigned long*	32		0 to 4,294,967,295
Float	32	7	$\pm 1.2 \times 10^{-38}$ to $\pm 3.4 \times 10^{+38}$
Double	64	14	$\pm 2.3 \times 10^{-308}$ to $\pm 1.7 \times 10^{+308}$
Extended*	80	19	$\pm 1.7 \times 10^{-4932}$ to $\pm 1.1 \times 10^{+4932}$

*Standard in ANSI C, but not available on all C or Pascal compilers.

MSB is one, then the leftmost additional bits of the longer number will be filled with ones (**sign extension**). In this way, the converted number will have the same numerical value in 2's complement representation. For example, when transferred from char to int, 35 becomes 0035 and 8A becomes FF8A. Thus, if unsigned numbers are read from a counter/timer or A/D converter in blocks of eight bits, some precautions are necessary before they can be packed into 16- or 32-bit numbers. There are two approaches:

1. Mask the left half of the number with zeros (see Appendix C).
2. Declare all relevant variables to be "unsigned."

Table 1.2 shows the typical internal representations available on microcomputers. They are also explored in Laboratory Exercise 1. Each program variable is declared to be one of these types. The float, double, and extended have 8-, 11-, and 15-bit exponents and 23, 52, and 63 bits of precision, which correspond to 7, 15, and 19 decimal digits of precision, respectively.

1.3.2 Gray code

Gray code is used extensively in external devices such as digital position encoders because the transition from any number to the next involves a change of only one bit (Table 1.3). If binary code were used, erroneous values could result when more than one bit changed, since it is not possible to guarantee that all bits change simultaneously from one number to the next.

The exclusive-OR circuits shown in Figure 1.3 convert numbers from Gray code to binary code and from binary code to Gray code. See the following section for a review of the AND, inclusive-OR, and exclusive-OR logic circuits. It will be noted on the left-hand side of Figure 1.3 that bit 1, for example, cannot be determined until bit 2 is known, and bit 2 cannot be determined until bit 3 is known, etc. Thus the output is valid only after N gate propagation times. A "valid data" signal can be derived by connecting

Table 1.3 *Binary and Gray codes and their decimal equivalents*

Decimal	Binary	Gray	Decimal	Binary	Gray
0	00000	00000	16	10000	11000
1	00001	00001	17	10001	11001
2	00010	00011	18	10010	11011
3	00011	00010	19	10011	11010
4	00100	00110	20	10100	11110
5	00101	00111	21	10101	11111
6	00110	00101	22	10110	11101
7	00111	00100	23	10111	11100
8	01000	01100	24	11000	10100
9	01001	01101	25	11001	10101
10	01010	01111	26	11010	10111
11	01011	01110	27	11011	10110
12	01100	01010	28	11100	10010
13	01101	01011	29	11101	10011
14	01110	01001	30	11110	10001
15	01111	01000	31	11111	10000

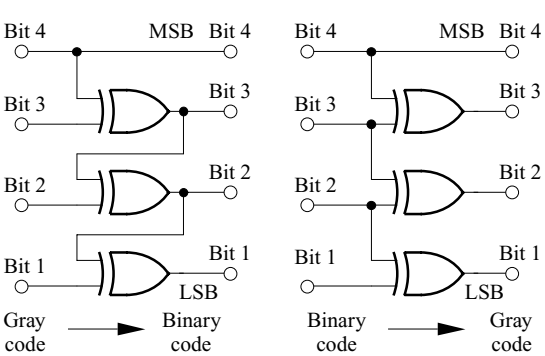

Figure 1.3 Circuits used to convert Gray code to binary and binary code to Gray code. Four bits are shown. The logic elements shown perform the exclusive OR, which has an output logic state that equals one only if the input logic states differ.

all input bits to an inclusive-OR circuit that is used as the input to a pulse generator. The output is read at the trailing edge of the pulse. Alternatively, a table lookup from computer memory or read-only memory (ROM) can be used to convert between Gray and binary codes.

1.4 Digital building blocks

This section describes the fundamental building blocks used to connect to a multiple-output bus, sample and store a logic state at a well-defined time, generate pulses,

Table 1.4 *Logic voltage ranges for TTL and ECL circuit families*

	TTL (V)	ECL (V)
Power supplies	0, +5 (±5%)	0, −5.2 (±5%)
Allowed "0" input range	−0.5 to +0.8	−5.0 to −1.4
Ambiguous input range	+0.8 to +2.0	−1.4 to −1.1
Allowed "1" input range	+2.0 to +5.5	−1.1 to +0.0
Nominal logic "0" output	+0.2	−1.75
Nominal logic "1" output	+3.2	−0.90
Allowed "0" output range	+0.0 to +0.4	−1.85 to −1.65
Ambiguous output range	+0.4 to +2.4	−1.65 to −0.96
Allowed "1" output range	+2.4 to +5.0	−0.96 to −0.81
Typical pulse risetime (10–90%)	10 ns*	1.5 ns†

*Low-power Schottky TTL.
†ECL 10,000.

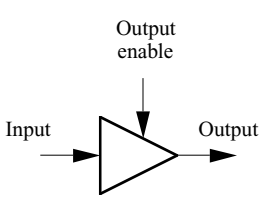

Figure 1.4 Tri-state buffer (see Table 1.5 for the function table and Figure 1.5 for a typical timing diagram).

and perform logical tests (AND, OR, etc.) of logic states. Table 1.4 lists the ranges of external voltages for the two most commonly used families of logic circuits, TTL (transistor–transistor logic) and ECL (emitter-coupled logic).

Note 1: To allow for voltage drop along conductors, the requirements for output are more stringent than for input.

Note 2: For both TTL and ECL, a logic 1 is always more positive in voltage than a logic 0.

1.4.1 Tri-state buffer

The **tri-state buffer** has three output states: asserted high, asserted low, and high impedance. In the high-impedance state, the output neither loads nor drives any circuit connected to it. This device has the usual logic input, but also has an additional enable input that determines whether the output follows the input or is put in the high-impedance state. The tri-state buffer is an essential component when several different outputs must be connected to form a common bus. See Figure 1.4 for the circuit schematic, Figure 1.5 for a typical timing diagram, and Table 1.5 for the function table.

Table 1.5 *Function table for tri-state buffer*

Input	Output enable	Tri-state output
H	L	H
L	L	L
X*	H	High impedance

*X = don't care.

Table 1.6 *Function table for edge-triggered* D*-type flip-flop*

Data D	Clock C	Flip-Flop output Q
H	↑†	H
L	↑	L
X*	H or ↓§ or L	Previous state

*X = don't care.
†↑ = low-to-high edge.
§↓ = high-to-low edge.

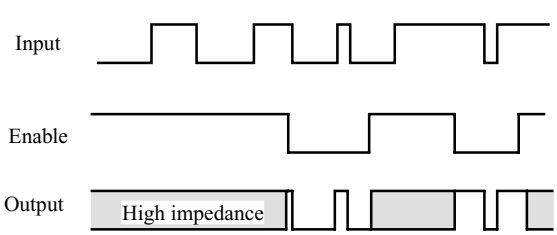

Figure 1.5 Typical timing diagram for the tri-state buffer (see Table 1.5 for the function table).

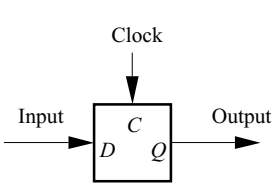

Figure 1.6 Edge-triggered D-type flip-flop (see Table 1.6 for the function table and Figure 1.7 for a typical timing diagram).

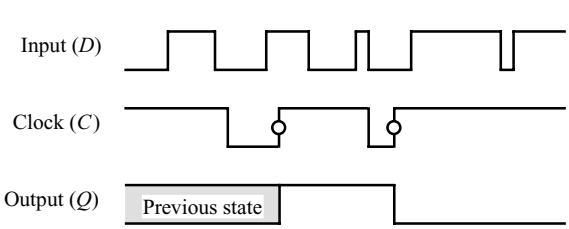

Figure 1.7 Typical timing diagram for the edge-triggered D-type flip-flop (see Table 1.6 for the function table).