14

Microcontrollers

Microcontrollers are hidden inside almost every product or device with which its user can interact. In fact, any device that has a remote control or has an LED/LCD screen and a keypad has an embedded microcontroller. Some common products where one is sure to find the use of a microcontroller include automobiles, microwave ovens, TVs, VCRs, high-end stereo systems, camcorders, digital cameras, washing machines, laser printers, telephone sets with caller ID facility, mobile phones, refrigerators and so on. This chapter focuses on microcontroller fundamentals and the application-related aspects of it. Beginning with an introductory description of the device, with particular reference to its comparison with a microprocessor, the chapter covers the general architecture and the criteria to be followed to choose the right device for a given application. This is followed by application-relevant information, such as salient features, pin configuration, internal architecture, etc., of popular brands of eight-bit, 16-bit, 32-bit and 64-bit microcontrollers from major international manufacturers. Intel's 8051 family of microcontrollers is described in more detail.

14.1 Introduction to the Microcontroller

The microcontroller may be considered as a specialized computer-on-a-chip or a single-chip computer. The word 'micro' suggests that the device is small, and the word 'controller' suggests that the device may be used to control one or more functions of objects, processes or events. It is also called an embedded controller as microcontrollers are often embedded in the device or system that they control.

The microcontroller contains a simplified processor, some memory (RAM and ROM), I/O ports and peripheral devices such as counters/timers, analogue-to-digital converters, etc., all integrated on a single chip. It is this feature of the processor and peripheral components available on a single chip that distinguishes it from a microprocessor-based system. A microprocessor is nothing but a processing

Digital Electronics: Principles, Devices and Applications Anil K. Maini
 ISBN: 978-0-470-03214-5

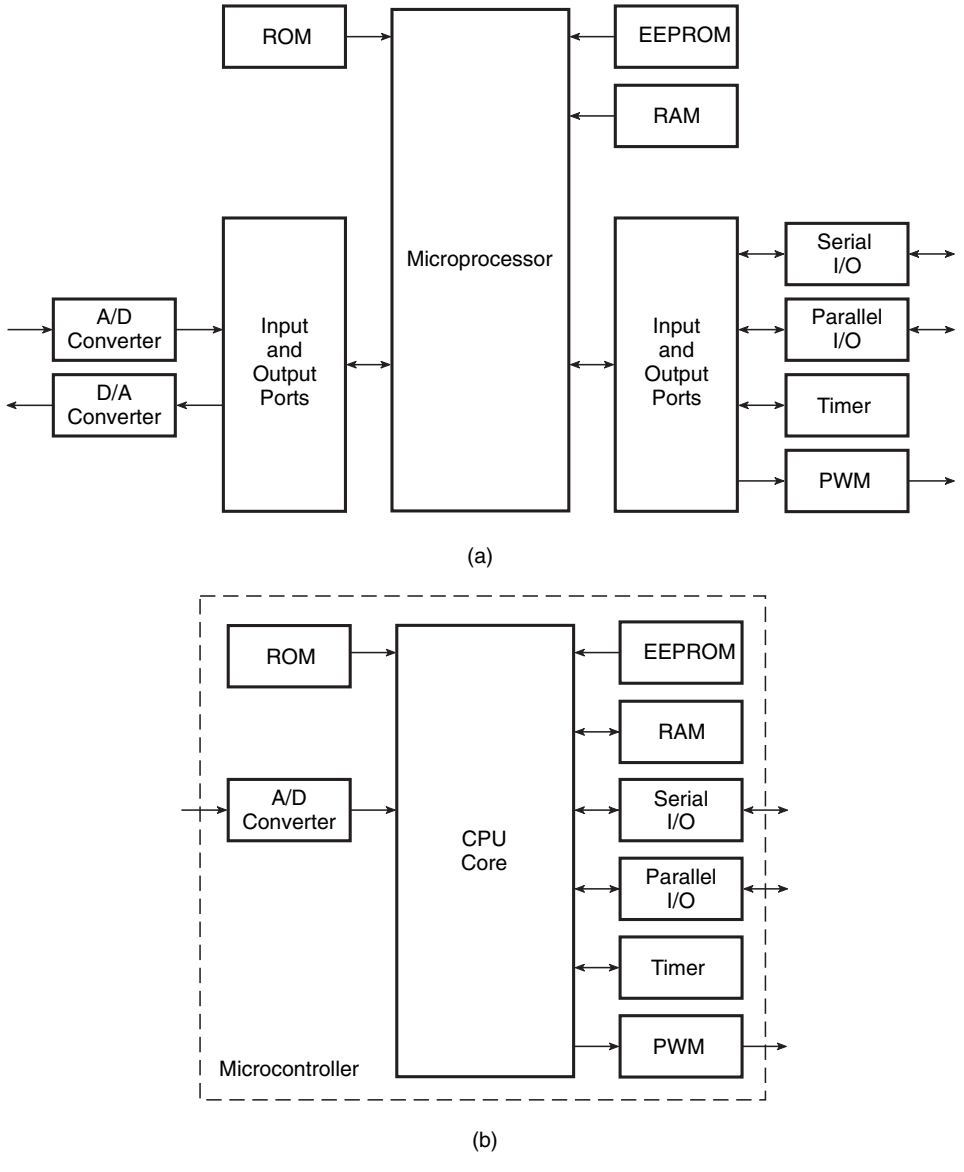

Figure 14.1 Microprocessor versus microcontroller: (a) microprocessor configuration; (b) microcontroller configuration.

unit with some general-purpose registers. A microprocessor-based system also has RAM, ROM, I/O ports and other peripheral devices to make it a complete functional unit, but all these components are external to the microprocessor chip. While a microprocessor-based system is a general-purpose system that may be programmed to do any of the large number of functions it is capable of doing,

microcontrollers are dedicated to one task and run one specific program. This program is stored in ROM and generally does not change.

Figure 14.1 further illustrates the basic difference between a microprocessor-based system and a microcontroller. As is evident from the two block schematics shown in the figure, while a microprocessor-based system needs additional chips to make it a functional unit, in a microcontroller the functions of all these additional chips are integrated on the same chip.

14.1.1 Applications

Microcontrollers are embedded inside a surprisingly large number of product categories including automobiles, entertainment and consumer products, test and measurement equipment and desktop computers, to name some prominent ones.

Any device or system that measures, stores, controls, calculates or displays information is sure to have an embedded microcontroller as a part of the device or system. In automobiles, one or more microcontrollers may be used for engine control, car cruise control (Fig. 14.2), antilock brakes and so on. Test and measurement equipment such as signal generators, multimeters, frequency counters, oscilloscopes, etc., make use of microcontrollers to add features such as the ability to store measurements, to display messages and waveforms and to create and store user routines. In desktop computers, microcontrollers are used in peripheral devices such as keyboards, printers, modems, etc. Consumer and entertainment products such as TVs, video recorders, camcorders, microwave ovens, washing machines, telephones with caller ID facility, cellular phones, air conditioners, refrigerators and many more products make extensive use of microcontrollers to add new control and functional features.

14.2 Inside the Microcontroller

Figure 14.3 shows the block schematic arrangement of various components of a microcontroller. As outlined earlier, a microcontroller is an integrated chip with an on-chip CPU, memory, I/O ports and some peripheral devices to make a complete functional unit. A typical microcontroller as depicted in Fig. 14.4 has the following components: a central processing unit (CPU), a random access memory (RAM), a read only memory (ROM), special-function registers and peripheral components including serial and/or parallel ports, timers and counters, analogue-to-digital (A/D) converters and digital-to-analogue (D/A) converters.

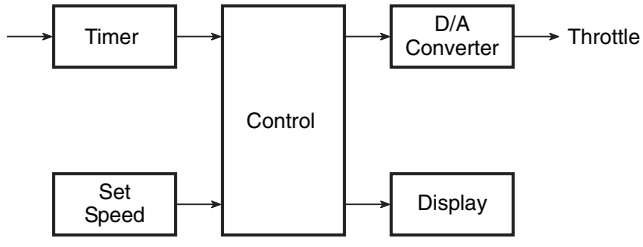

Figure 14.2 Microcontroller-based car cruise control.

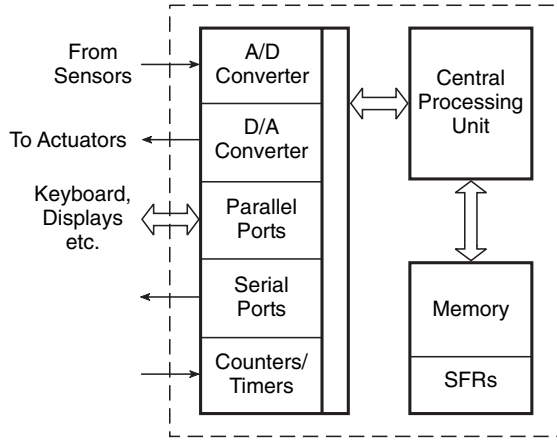

Figure 14.3 Inside the microcontroller.

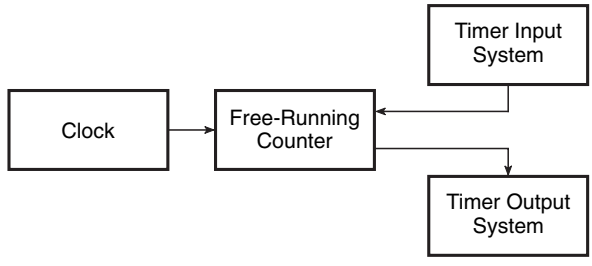

Figure 14.4 Timer subsystem.

14.2.1 Central Processing Unit (CPU)

The central processing unit processes the program. It executes the instructions stored in the program memory pointed to by the program counter in synchronization with the clock signal. The processor complexity could vary from simple eight-bit processors to sophisticated 32-bit or even 64-bit processors. Some common microcontrollers using eight-bit processors include 68HC11 (Freescale Semiconductor – earlier part of Motorola), the 80C51 family of microcontrollers (Intel and Dallas Semiconductor), Zilog-eZ8 and Zilog-eZ80 (Zilog) and XC800 (Infineon). Examples of microcontrollers using 16-bit processors include the 8096 family (Intel), 68HC12 and 68HC16 (Freescale Semiconductor), the F2MC family (Fujitsu) and the XC166 family (Infineon). Examples of microcontrollers using 32-bit processors include 683XX, MPC 860 (PowerQUICC), MPC 8240/8250 (PowerQUICC-II) and MPC 8540/8555/8560 (PowerQUICC-III) (all from Freescale Semiconductor), the TRICORE family (Infineon) and the FR/FR-V family (Fujitsu).

14.2.2 Random Access Memory (RAM)

RAM is used to hold intermediate results and other temporary data during the execution of the program. Typically, microcontrollers have a few hundreds of bytes of RAM. As an example, microcontroller type numbers 8XC51/80C31, 8XC52/80C32 and 68HC12 respectively have 128, 256 and 1024 bytes of RAM.

14.2.3 Read Only Memory (ROM)

ROM holds the program instructions and the constant data. Microcontrollers use one or more of the following memory types for this purpose: ROM (mask-programmed ROM), PROM (one-time programmable ROM, which is not field programmable), EPROM (field programmable and usually UV erasable), EEPROM (field programmable, electrically erasable, byte erasable) and flash (similar to EEPROM technology). Microcontroller type numbers 8XC51, 8XC51FA and 8XC52 have 4K, 8K and 16K of ROM. As another example, the 68HC12 16-bit microcontroller has 32K of flash EEPROM, 768 bytes of EEPROM and 2K of erase-protected boot block.

14.2.4 Special-Function Registers

Special-function registers control various functions of a microcontroller. There are two categories of these registers. The first type includes those registers that are wired into the CPU and do not necessarily form part of addressable memory. These registers are used to control program flow and arithmetic functions. Examples include status register, program counter, stack pointer, etc. These registers are, however, taken care of by compilers of high-level languages, and therefore programmers of high-level languages such as C, Pascal, etc., do not need to worry about them. The other category of registers is the one that is required by peripheral components. The contents of these registers could, for instance, set a timer or enable serial communication and so on. As an example, special-function registers available on the 80C51 family of microcontrollers (80C51, 87C51, 80C31) include a program counter, stack pointer, RAM address register, program address register and PC incrementer.

14.2.5 Peripheral Components

Peripheral components such as analogue-to-digital converters, I/O ports, timers and counters, etc., are available on the majority of microcontrollers. These components perform functions as suggested by their respective names. In addition to these, microcontrollers intended for some specific or relatively more complex functions come with many more on-chip peripherals. Some of the common ones include the pulse width modulator, serial communication interface (SCI), serial peripheral interface (SPI), interintegrated circuit (I^2C) two-wire communication interface, RS 232 (UART) port, infrared port (IrDA), USB port, controller area network (CAN) and local interconnect network (LIN). These peripheral devices are briefly described in the following paragraphs.

14.2.5.1 Analogue-to-Digital Converters

Analogue-to-digital and digital-to-analogue converters provide an interface with analogue devices. For example, the analogue-to-digital converter provides an interface between the microcontroller and the sensors that produce analogue electrical equivalents of the actual physical parameters to be controlled.

The digital-to-analogue converter, on the other hand, provides an interface between the microcontroller and the actuators that provide the control function. As an example, both 68HC11 and 68HC12 from Freescale Semiconductor have eight-channel, eight-bit analogue-to-digital converters. The digital-to-analogue converter function in microcontrollers is provided by a combination of pulse width modulator (PWM) followed by a filter. As an example, 68HC12 has an on-chip 16-bit/two-channel PWM. Analogue-to-digital and digital-to-analogue converters are discussed at length in Chapter 12.

14.2.5.2 I/O Ports

I/O ports provide an interface between the microcontroller and the peripheral I/O devices such as the keyboard, display, etc. The 80C51 family of microcontrollers has four eight-bit I/O ports. Microcontroller 68HC11 offers 38 general-purpose I/O pins including 16 bidirectional I/O pins, 11 input-only pins and 11 output-only pins.

14.2.5.3 Counters/Timers

Counters/timers usually perform the following three functions. They are used to keep time and/or measure the time interval between events, count the number of events and generate baud rates for the serial ports. Microcontroller 68HC11 has a 16-bit timer system comprising three input capture channels, four output compare channels and one additional channel that can be configured as either an input or an output channel. Another popular microcontroller type number, PIC 16F84, has an eight-bit timer/counter with an eight-bit prescaler.

Figure 14.4 shows a generalized block schematic representation of the timer subsystem of a microcontroller. The clock signal controls all timing activities of the microcontroller. The counter is used both to capture external timing events (accomplished by the timer input block) and to generate timing events for external devices (accomplished by the timer output block). While the former process is typically used to measure the frequency and time interval of periodic signals, the latter generates control signals for external devices.

It may be mentioned here that a timing event to be captured or generated is nothing but a change in logic status on one of the microcontroller I/O pins configured as an input pin if the event is to be captured and as an output pin if it is to be generated. Figure 14.5 shows a block schematic arrangement of the timer input block of Fig. 14.4. As shown in the figure, the counter captures the input time event in the form of its contents at the time of occurrence of the event. In fact, the counter captures the relative time of the event as the counter is free running. Absolute timing values can be computed from the relative system clock values. As an example, consider a microcontroller with a 10 MHz clock and a 16-bit counter/timer subsystem. This counter will take 6.5536 ms to count from 0000 to FFFF (hex notation). Let us assume that it is desired to find the frequency of a periodic signal whose successive rising or falling edges are observed to occur at 0010 and 0150. 0010 and 0150 respectively correspond to 16 and 336 in decimal. Therefore, the time interval between two successive edges equals $320 \times 0.1 = 32$ μs. The signal frequency is therefore (1/32) MHz = 31.25 kHz.

Figure 14.6 shows a block schematic arrangement of the timer output block of Fig. 14.4. The diagram is self-explanatory. Again, free-running counter values can be used to synchronize the time of the desired logic state changes on the output pin. This feature can also be used to generate an aperiodic pulse or a periodic signal of any desired duty cycle.

For timer input and output operations, the microcontroller needs to set up some special registers. For timer input operation, as shown in Fig. 14.5, registers are required to program the event (logic HIGH or logic LOW), configure the physical I/O pin as an input pin and also set up parameters for the

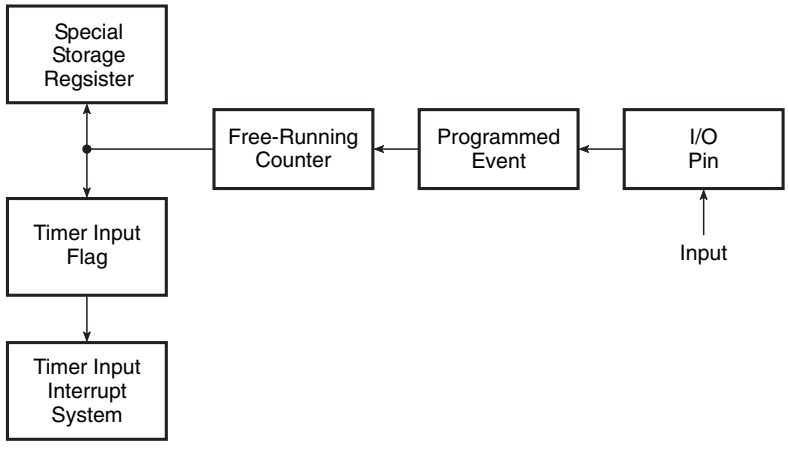

Figure 14.5 Timer input subsystem.

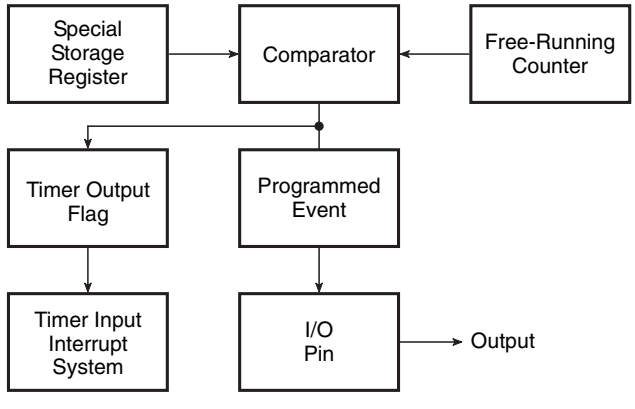

Figure 14.6 Timer output subsystem.

related interrupt, if used. Another register is used to capture the counter value at the time of occurrence of the event. For time output operation, as shown in Fig. 14.6, the physical I/O pin is to be configured as an output pin, the event is to be programmed and the timing value is to be set in the special register to tell when the programmed event should appear on the output pin. The output timer system also has an associated interrupt that can be utilized if needed.

14.2.5.4 Serial Communication Interfaces

There are two types of serial communication interface, namely the asynchronous communication interface and the synchronous communication interface. The asynchronous communication interface uses a start and stop bit protocol to synchronize the transmitter and receiver. Start and stop bits

are embedded in each data byte. Compared with the synchronous communication interface, it offers lower data transmission rates. It is also referred to as the universal asynchronous receiver/transmitter (UART) or the serial communication interface (SCI). The synchronous communication interface uses a synchronized clock to transmit and receive each bit. Synchronization of transmitter and receiver clocks is usually accomplished by using an additional clock line linking the transmitter and the receiver. It is not recommended for long distance communication. It is also referred to as the serial peripheral interface (SPI). Microcontroller 68HC11 offers an asynchronous non-return-to-zero serial communication interface and also a synchronous serial peripheral interface. The 80C51 family of microcontrollers offers a full duplex-enhanced UART interface.

Since a large number of peripheral devices are equipped to communicate with an RS-232-compatible interface, which is a serial interface standard that specifies the different aspects, including electrical, mechanical, functional and procedural specifications, a variety of chips are available to translate microcontroller signals to RS-232-compatible signals. These chips are equipped to provide interfacing for a two-way communication system.

14.2.5.5 Interintegrated Circuit (I^2C) Bus

The interintegrated circuit (I^2C) bus is a two-wire, low- to-medium-speed serial communication interface developed by Philips Semiconductors in the early 1980s for chip-to-chip communications. The two wires in the I^2C bus are called clock (SCL) and data (SDA). The SDA wire carries data, while the SCL wire synchronizes the transmitter and receiver during data transfer.

It is a proven industry-standard communication protocol used in a variety of electronic products, which is particularly facilitated by its low cost and powerful features. It is supported by a large number of semiconductor and system manufacturers who offer a variety of electronic products including input and output devices, different types of sensor, memory devices, displays, data entry devices, etc. Some of the important features offered by I^2C devices are briefly described in the following paragraphs.

I^2C devices offer master–slave hierarchy. These are classified as either master (the device that initiates the message) or slave (the device that responds to the message). The device can be either master only or slave only or can be switched between master and slave depending upon the application requirement. One possible master–slave configuration is the one where one master (e.g. a microcontroller) is connected to many chips configured as slaves, as shown in Fig. 14.7. Each of the I^2C slave devices is

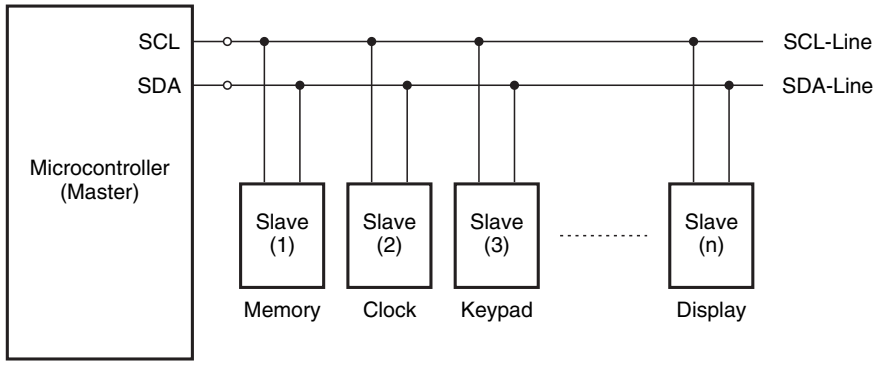

Figure 14.7 Master–slave configuration – one I^2C master and multiple slaves.

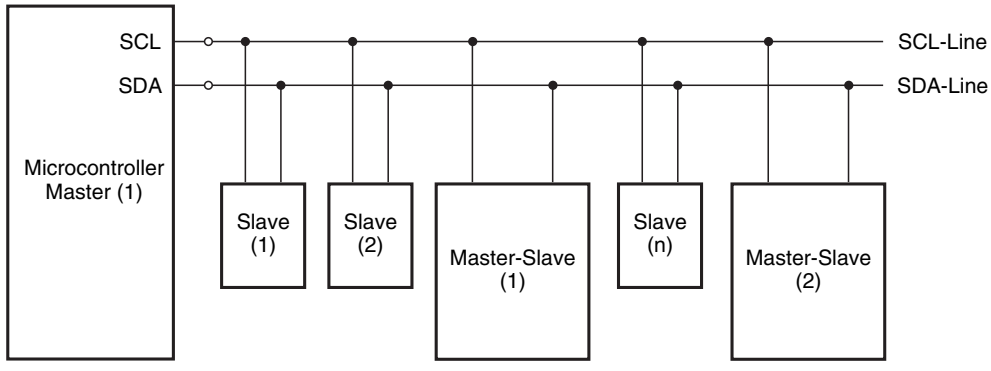

Figure 14.8 Master–slave configuration – multiple-master support arrangement.

identifiable by a unique address. When the master device sends a message, it includes the address of the intended slave device at the beginning of the message.

The I^2C interface also supports multiple master devices at the same time. The bus has a special feature that allows it to resolve signal conflicts should two or more master devices try to talk on the bus at the same time. A master I^2C device that detects the conflict, called arbitration loss, terminates its use of the bus, thus allowing the message sent by another master to cross the bus unharmed. Figure 14.8 shows one such multimaster support arrangement.

14.2.5.6 Controller Area Network (CAN) Bus

The controller area network (CAN) bus is a rugged serial communication interface used in a broad range of embedded as well as automation control applications. It was introduced by Bosch in 1986 for in-vehicle networks in automobiles. The CAN protocol was internationally standardized in 1993 as ISO-11898-1 and comprises the data link layer of the seven-layer ISO/OSI reference model. The protocol provides two communication services, namely data frame transmission (sending of a message) and remote transmission request (requesting of a message). All other services such as error signalling, automatic retransmission of erroneous frames, etc., are performed by CAN chips. Some of the important features of the CAN protocol include the following. It provides a multimaster hierarchy. This allows the user to build intelligent and redundant systems. It uses the broadcast communication method. The sender of a message transmits to all devices connected to the bus. All devices read the message and decode it if it is intended for them. This feature guarantees data integrity. Data integrity is also ensured by error detection mechanisms and automatic retransmission of faulty messages.

CAN protocol provides low-speed fault-tolerant transmission at a rate of 125 kbps up to a distance of 40 m, which can function over one wire if there is a short. Transmission without fault tolerance is provided at a rate of 1 Mbps up to a distance of 40 m. Transmission rates of 50 kbps are achievable up to a distance of 1 km.

14.2.5.7 Local Interconnect Network (LIN) Bus

The local interconnect network (LIN) bus is a broadcast serial network that is used as a low-cost subnetwork of a CAN bus to integrate intelligent sensors or actuators in modern automobiles. It

comprises one master (typically a moderately powerful microcontroller) and up to 16 slaves (less powerful, cheaper microcontrollers or ASICs). It does not offer a collision detection feature and therefore all messages are initiated by the master with at the most one slave replying to a given message identifier. Multiple such LIN networks may all be linked to a CAN upper layer network through their respective masters.

Example 14.1

A certain microcontroller has an on-chip 16-bit counter/timer system. It is used to measure the width of an input pulse. The microcontroller has been programmed to measure the time of occurrence of rising and falling edges of an input pulse on a certain I/O pin. If the microcontroller uses an 8 MHz clock and the count values observed at the time of occurrence of rising and falling edges of the input pulse are 001F and 00F1 (in hex), determine the pulse width as measured by the microcontroller.

Solution

- Since the microcontroller uses a 16-bit counter, it counts from 0000 to FFFF (in hex) or 0 to 65536 in decimal.
- The rising edge of the input pulse occurs at 001F, the decimal equivalent of which is 31.
- The falling edge occurs at 00F1, the decimal equivalent of which is 241.
- Therefore, the input pulse width accounts for $241 - 31 = 210$ clock cycles.
- The clock signal time period $= 1/8 = 0.125$ μs.
- Therefore, the time period corresponding to 210 cycles $= 210 \times 0.125 = 26.25$ μs.
- The pulse width measured by the microcontroller $= 26.25$ μs.

Example 14.2

It is desired to design a microcontroller-based periodic signal generator with minimum and maximum time period specifications of 125 ns and 100 ms. What should the system clock frequency be?

Solution

- The minimum time period that can be generated by the microcontroller equals the time period corresponding to one clock cycle.
- Therefore, one clock cycle time period $= 125$ ns.
- The clock frequency $= 1/125$ GHz $= 1000/125$ MHz $= 8$ MHz.

14.3 Microcontroller Architecture

Microcontroller architecture may be defined in several ways. These include architecture used by the processor to access memory, architecture used for mapping special-function registers into memory space and the processor architecture itself.

14.3.1 Architecture to Access Memory

There are two fundamental architectures used by the processing units to access memory, namely Von Neumann architecture and Harvard architecture.

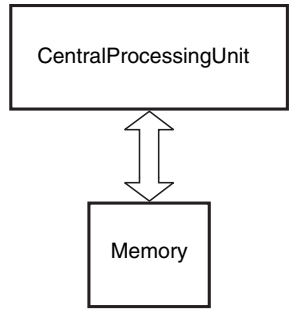

Figure 14.9 Von Neumann architecture.

Von Neumann architecture uses a single memory to hold both program instructions and data. There is one common data and address bus between processor and memory (Fig. 14.9). Instructions and data are fetched in sequential order, thus limiting the operation data transfer rate or the throughput. This phenomenon is commonly referred to as the Von Neumann bottleneck. The throughput is very small compared with the size of the memory. In present-day machines, the throughput is also very small compared with the rate at which the processor itself can work. In the condition where the processor is required to perform minimal processing on large amounts of data, the processor is forced to wait for vital data to be transferred from or to memory. Microcontroller type number 68HC11 uses Von Neumann architecture.

Harvard architecture uses physically separate memories for program instructions and data. It therefore requires separate buses for program and data, as shown in Fig. 14.10. In such architecture, instructions and operands can be fetched simultaneously, which makes microcontrollers using this architecture much faster compared with the ones using Von Neumann architecture. Also, different data and program bus widths are possible, which allows the program and data memory to be better optimized to architectural requirements. In fact, the word width, timing, implementation technology and memory address structure can be different in the two cases. Program memory is usually much larger than data memory, which implies that the address bus for the program memory is wider than the address bus for the data memory.

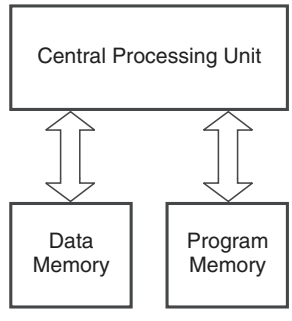

Figure 14.10 Harvard architecture.

14.3.2 Mapping Special-Function Registers into Memory Space

There are again two fundamental architectures used for mapping special-function registers into memory space. In the first type of arrangement, I/O space and memory space are separated as shown in Fig. 14.11(a). I/O devices have a separate address space, which is accomplished by either an extra I/O pin on the CPU physical interface or through a dedicated I/O bus. As a result of this, access to I/O control registers requires special instructions. It is particularly attractive in CPUs having a limited addressing capability. It is generally found on Intel microprocessors.

In the second arrangement, called the memory-mapped I/O, I/O control registers are mapped into memory address space as shown in Fig. 14.11(b). Read and write operations to the control registers are done via absolute memory addresses, which could be variables at absolute addresses or pointers to absolute addresses in high-level languages. In this case, no special instructions are needed to access I/O control registers. The memory-mapped I/O uses the same bus to address both memory and I/O devices. CPU instructions used to read from or write to memory are also used in accessing I/O devices.

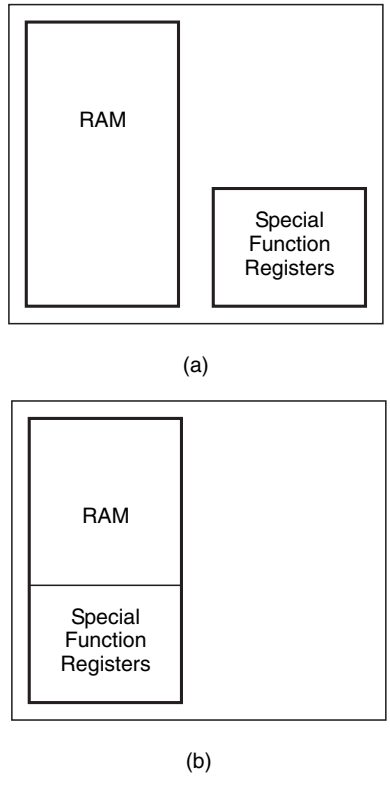

Figure 14.11 (a) Separate I/O and memory spaces and (b) memory-mapped I/O.

14.3.3 Processor Architecture

Processor architecture can be categorized as one of the following four architectures: accumulator-based architecture, register-based architecture, stack-based architecture and pipeline architecture.

14.3.3.1 Accumulator-based Architecture

In accumulator-based architecture, as shown in Fig. 14.12, instructions begin and end in accumulators (Acc A and Acc B in Fig. 14.12), which are specially designated registers. In a typical operation, one of the operands is found in the accumulator and the other is to be fetched from memory. The result of the operation is placed in the accumulator. As one of the operands needs to be continually fetched from memory, this architecture is slower than the register-based and stack-based architectures. However, accumulator-based architecture has the ability to run fairly complicated instructions.

14.3.3.2 Register-based Architecture

In register-based architecture, as shown in Fig. 14.13, both operands are stored in registers and the result of operation is also stored in a register. The registers are typically colocated with the processor. Since the processor and registers operate at the same speed, this architecture is much faster than the previously discussed accumulator-based architecture. The contents of the register are read from and written to memory using background operation.

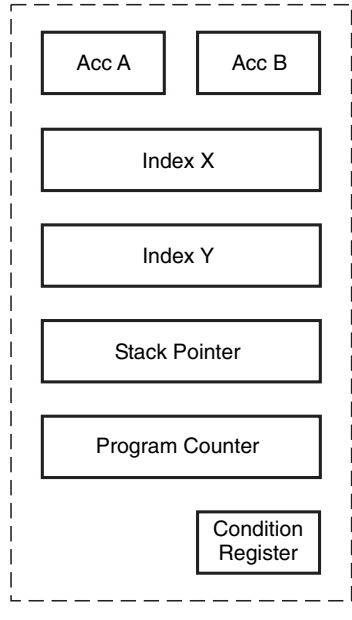

Figure 14.12 Accumulator-based processor architecture.

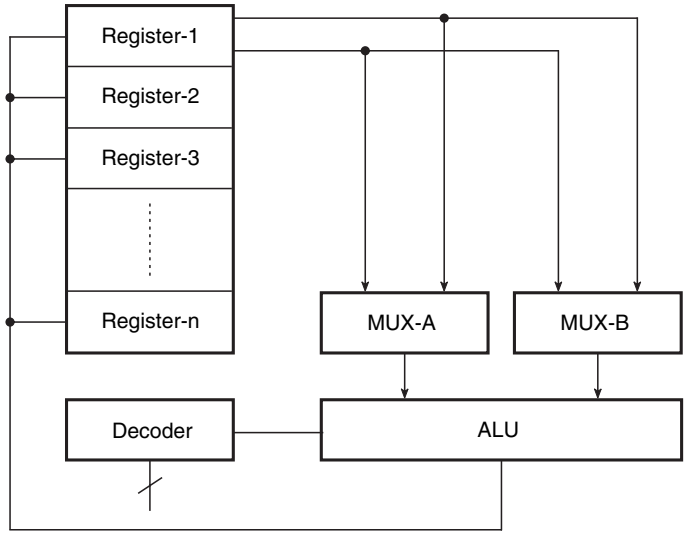

Figure 14.13 Register-based processor architecture.

14.3.3.3 Stack-based Architecture

In stack-based architecture, both operands and the operation to be performed are stored on the stack, which could be configured around dedicated registers or a special portion of RAM. The result of operation is placed back on the stack. Figure 14.14 shows typical block schematic arrangement of this type of architecture.

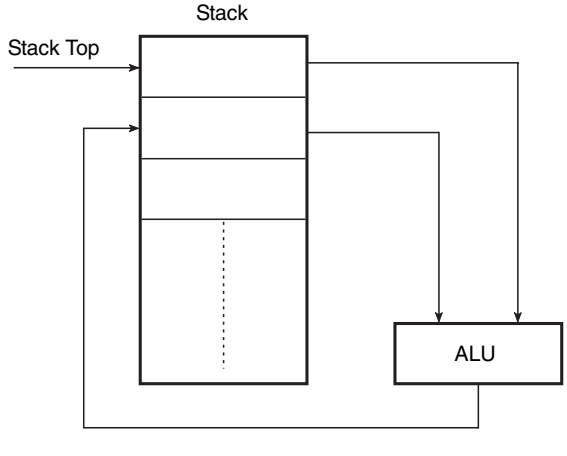

Figure 14.14 Stack-based processor architecture.

Figure 14.15 Pipelined architecture.

14.3.3.4 Pipeline Architecture

In pipelined architecture, as shown in Fig. 14.15, there are separate hardware stages for execution of different steps involved in execution of an instruction. These different steps include fetching an instruction from memory, decoding the instruction, fetching instruction operands from memory or registers, executing the instruction and then finally placing the result back on the memory. Pipelining allows these stages to overlap and perform with parallelism. The overall result is that the execution of an instruction is completed on every clock cycle. Instructions in a pipelined architecture are usually simple instructions that can be implemented within a single stage. These simple instructions act as building blocks for more complex instructions.

14.4 Power-Saving Modes

Power consumption is one of the important issues in battery-powered devices. Most microcontrollers come with various power-saving features. For a given application requirement, designers use these features to keep the power consumption down to an optimum value without compromising the operational requirements of the device. It may be mentioned here that not all modes are for power saving. Some microcontrollers support in-circuit debugging. As an example, some of the power-saving modes available with the 80C51 family of microcontrollers are briefly outlined in the following paragraphs.

The *stop clock mode* allows the clock oscillator to be stopped or the clock speed to be reduced to as low as 0 MHz. When the oscillator is stopped, the special-function registers and RAM retain their values. This mode allows reduced power consumption by lowering the clock frequency to any value.

The *idle mode* is another power-saving mode available with the 80C51 family of microcontrollers. In this mode, the processor puts itself to sleep while all on-chip peripheral components stay active. The processor contents, the on-chip RAM and all special-function registers remain intact during the idle mode. The instruction that invokes this mode is the last instruction executed in the normal operating mode before the idle mode is activated. The idle mode can be terminated by either an enabled interrupt or by a hardware reset. By an enabled interrupt, the process is picked up at the interrupt service routine and continued. Hardware reset starts the processor in the same manner as it does on a power-on reset.

The *power down mode* is recommended for the lowest power consumption. When this mode is enabled, the oscillator stops and the instruction that invokes the power down mode is the last instruction executed. Special-function registers and on-chip RAM retain their values down to a V_{CC} amplitude of 2.0 V. V_{CC} must be brought to the minimum specified operating voltage before this mode is deactivated. Either a hardware reset or an external interrupt can be used to terminate the power down mode. While a hardware reset redefines all the special-function registers and retains on-chip RAM values, an external interrupt allows both special-function registers and the on-chip RAM to retain their values. For proper termination of the power-down mode, a reset or external interrupt should not be executed unless $V_{CC}$is restored to its normal operating level and also has been held active long enough for the oscillator to start and stabilize.

Yet another mode available with the 80C51 family of microcontrollers that helps in power saving is the LPEP. The EPROM array contains some analogue circuits that are not required for a V_{CC} of less than 4.0 V. This feature can be used to save power by setting the LPEP bit, resulting in reduced supply current. This mode should be used only for applications that require a V_{CC} of less than 4.0 V.

14.5 Application-Relevant Information

This section briefly presents application-relevant information in terms of general specifications, microcontroller-related features and peripheral features on some of the common types of microcontroller from well-known international manufacturers including Intel, Freescale Semiconductor, Microchip Technology, Altera, Atmel, Zilog, Lattice Semiconductor, National Semiconductor, Applied Micro Circuits Corporation (AMCC), Fujitsu, Infineon, Dallas Semiconductor, Philips Semiconductors, Texas Instruments, Xilinx, NEC, Toshiba and so on. Some of the more widely used type numbers, including the 80C51 family of microcontrollers (Intel and many more manufacturers), the 89C51 microcontroller (Intel and many more manufacturers), the 68HC11 family of microcontrollers (Freescale Semiconductor) and the PIC 16X84 family of microcontrollers (Microchip Technology), are discussed in a little more detail. For these type numbers, information such as architecture, pin connection diagrams, functional description of different pins, addressing modes, etc., is also presented.

14.5.1 Eight-Bit Microcontrollers

This subsection outlines salient features of popular eight-bit microcontrollers. For most of the type numbers, the information is contained under two headings, namely microcontroller-related features and peripheral-related features.

14.5.1.1 80C51/87C51/80C31 (Dallas Semiconductor and Other Manufacturers)

Microcontroller-related Features

MCS-51 architecture, CMOS technology, 4K × 8 ROM (no ROM in 80C31), 128 × 8 RAM, memory addressing capability of 64K (ROM and RAM), special-function registers, six interrupt sources, three power control modes including STOP CLOCK, IDLE and POWER DOWN modes, two clock speed ranges of 0–16 MHz and 0–33 MHz, low EMI (inhibit ALE) and three package style options (40-pin dual in-line, 44-pin plastic leaded chip carrier and 44-pin plastic quad flat pack).

Peripheral-related Features

Two 16-bit counters/timers, four eight-bit I/O ports and full duplex-enhanced UART.

Architecture and Pin Connection Diagram

Figure 14.16 shows the architecture and Fig. 14.17 shows the pin connection diagram in the 40-pin dual in-line package.

Registers

Registers are categorized as general-purpose registers and special-function registers. The 80C51 family of microcontrollers has an accumulator, B-register and four register banks, each having eight-bit wide registers R0 to R7. Registers R0 through R7 are used as scratch-pad registers. In addition, there is

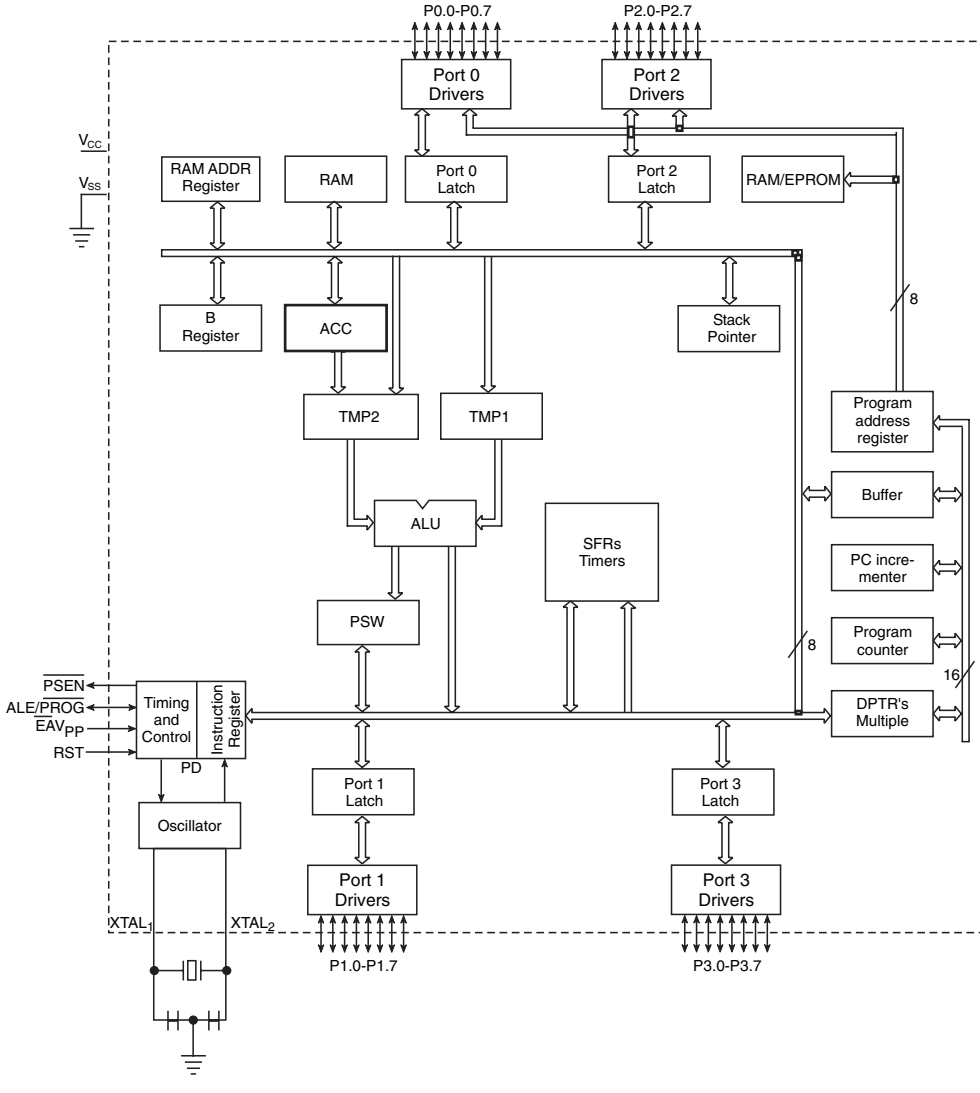

Figure 14.16 Architecture of the 80C51 microcontroller family.

an eight-bit wide stack pointer and a 16-bit wide program counter. Special-function registers include program status word (PSW), data pointer (DPTR), timer registers, control registers and capture registers.

Addressing Modes

The 80C51 family of microcontrollers supports five addressing modes including register addressing, direct addressing, register indirect addressing, immediate addressing and base register plus index register addressing.

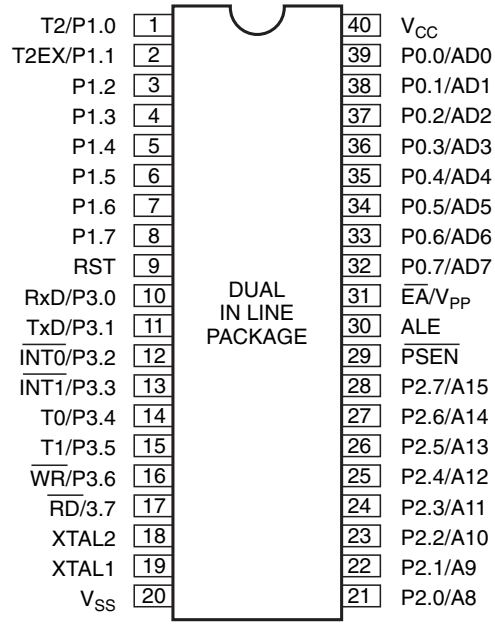

Figure 14.17 PIN connection diagram in the 40-pin DIP package.

Instruction Set

The instruction set of the 80C51 family of microcontrollers consists of 111 instructions divided into five categories, namely data transfer instructions, arithmetic instructions, logical instructions, Boolean variable manipulation instructions and control transfer instructions.

Interrupts

The 80C51 family of microcontrollers supports five vectored interrupts. These include external interrupt 0, external interrupt 1, timer/counter 0 interrupt, timer/counter 1 interrupt and serial port interrupts.

Power Modes

The 80C51 family of microcontrollers offers various operational modes that can be used to reduce power consumption. These include STOP CLOCK MODE which enables the clock speed to be reduced down to 0 MHz, IDLE MODE when the CPU puts itself to sleep while all of the on-chip peripherals stay active and POWER DOWN MODE in which the oscillator is stopped. In addition to the power-saving operational modes, it also offers ONCE™ (On-Circuit Emulation) MODE which facilitates in-circuit testing and debugging.

14.5.1.2 80C31FA/8XC51FA/FB/FC (Dallas Semiconductor and Other Manufacturers)

The same as 80C51 except for the size of ROM and RAM, which is 0K/8K/16K/32K (ROM) and 256 bytes (RAM).

14.5.1.3 80C31RA$^+$/8XC51RA$^+$/RB$^+$/RC$^+$(Dallas Semiconductor and Other manufacturers)

The same as 80C51 except for the size of ROM and RAM, which is 0K/8K/16K/32K (ROM) and 512 bytes (RAM).

14.5.1.4 8XC51RD$^+$(Dallas Semiconductor and Other Manufacturers)

The same as 80C51 except for the size of ROM and RAM, which is 64K (ROM) and 1024 bytes (RAM).

14.5.1.5 80C32/8XC52/54/58 (Dallas Semiconductor and Other Manufacturers)

The same as 80C51 except for the size of ROM and RAM, which is 0K/8K/16K/32K (ROM) and 256 bytes (RAM).

14.5.1.6 89C51 (ATMEL and Other Manufacturers)

Microcontroller-related Features

MCS-51 architecture, CMOS technology, 4K $\times$ 8 of in-system reprogrammable ROM, 128 $\times$ 8 internal RAM, memory addressing capability of 64K (ROM and RAM), special-function registers, six interrupt sources, two power-saving modes (IDLE and POWER DOWN modes), a clock speed range of 0–24 MHz, low EMI (inhibit ALE), three package style options (40-pin dual in-line, 44-pin plastic leaded chip carrier and 44-pin plastic quad flat pack) and compatible with the industry-standard MCS-51 instruction set and pin-out.

Peripheral-related Features

Two 16-bit counters/timers, 32 programmable I/O lines and a programmable serial channel.

Architecture and Pin Connection Diagram

The architecture and pin connection diagram are the same as those given earlier for the case of the 80C51 family of microcontrollers in Fig. 14.16 (architecture) and Fig. 14.17 (pin connection diagram).

14.5.1.7 68HC05 Family of Microcontrollers (Freescale Semiconductor)

Microcontroller-related Features

Fully static chip design using a standard eight-bit M68HC05 core, a clock speed of 4 MHz, 920 bytes of on-chip RAM, 32K of ROM, 7932 bytes of EEPROM (maximum values across the family of devices), power-saving WAIT mode and available in 40-pin DIP and 42-pin SDIP package styles.

Peripheral-related Features

Two serial interface channels, a multifunction timer with periodic interrupt, eight A/D converter channels, three PWM channels and 80 I/O lines (maximum values across the family of devices).

14.5.1.8 68HC11 Family of Microcontrollers (Freescale Semiconductor)

Microcontroller-related Features

Fully static chip design using an eight-bit M68HC11 core, a clock speed of 5 MHz, 0/256/512/768/1024 bytes of on-chip RAM (in different variants), 0/12/20 kB of on-chip ROM or EPROM (in different variants), 0/512/2048 bytes of on-chip EEPROM (in different variants), power-saving STOP and WAIT modes and available in six different package styles.

Peripheral-related Features

Asynchronous non-return-to-zero (NRZ) serial communication interface (SCI), synchronous serial peripheral interface (SPI), eight-channel, eight-bit analogue-to-digital converter, 16-bit timer system including three input capture channels, four output compare channels and an additional channel configurable as an input or an output channel, eight-bit pulse accumulator and 38 general-purpose I/O pins including 16 bidirectional I/O pins, 11 input-only pins and 11 output-only pins.

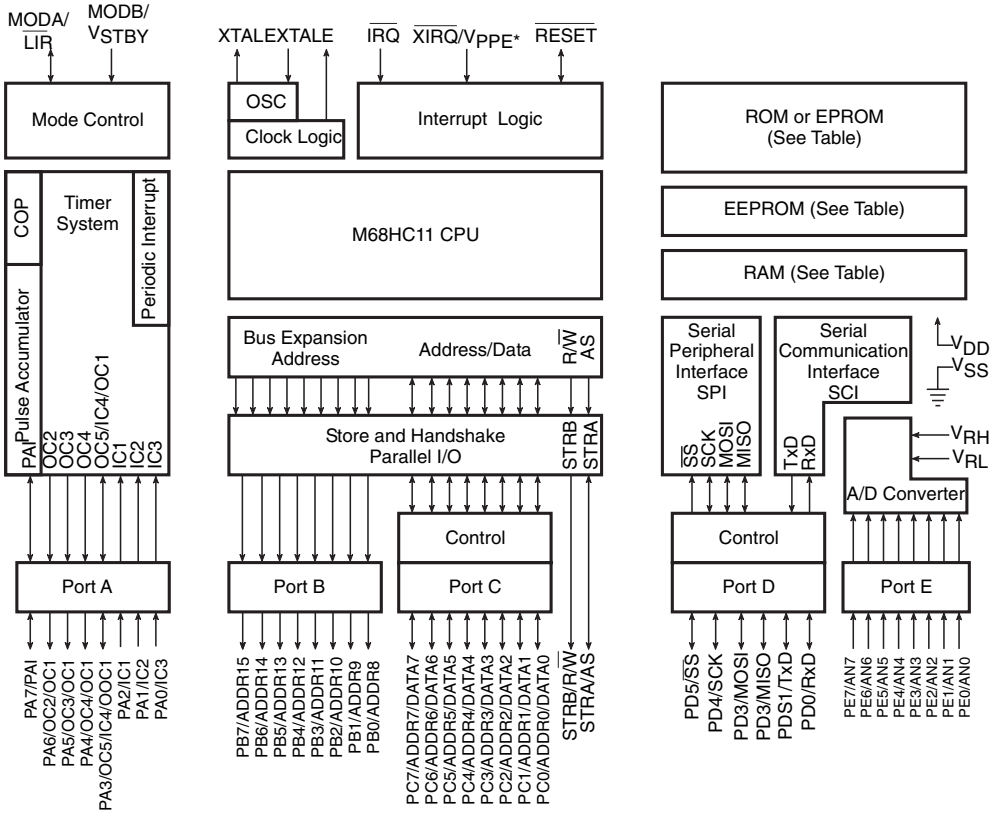

Figure 14.18 Architecture of the 68HC11 family of microcontrollers.

Architecture and Pin Connection Diagram

Figure 14.18 shows the architecture of the 68HC11 family of microcontrollers. Pin connection diagrams are shown in Fig. 14.19 (56-pin SDIP package) and Fig. 14.20 (48-pin DIP package). DIP and SDIP respectively stand for dual in-line package and shrink dual in-line package.

14.5.1.9 PIC 16X84 Family of Microcontrollers (Microchip Technology)

PIC 16C84 and PIC 16F84 are the two microcontrollers in the PIC 16X84 family of microcontrollers from Microchip Technology. PIC 16F84 is an improved version of PIC 16C84.

Microcontroller-related Features

High-performance RISC CPU, 14-bit wide instructions, eight-bit wide data path, 1024 × 14 EEPROM program memory, 64 bytes of on-chip data EEPROM, 36 × 8 general-purpose registers (16C84), 68 bytes of data RAM (16F84), 15 special-function hardware registers (16F84), a clock speed of 10/20

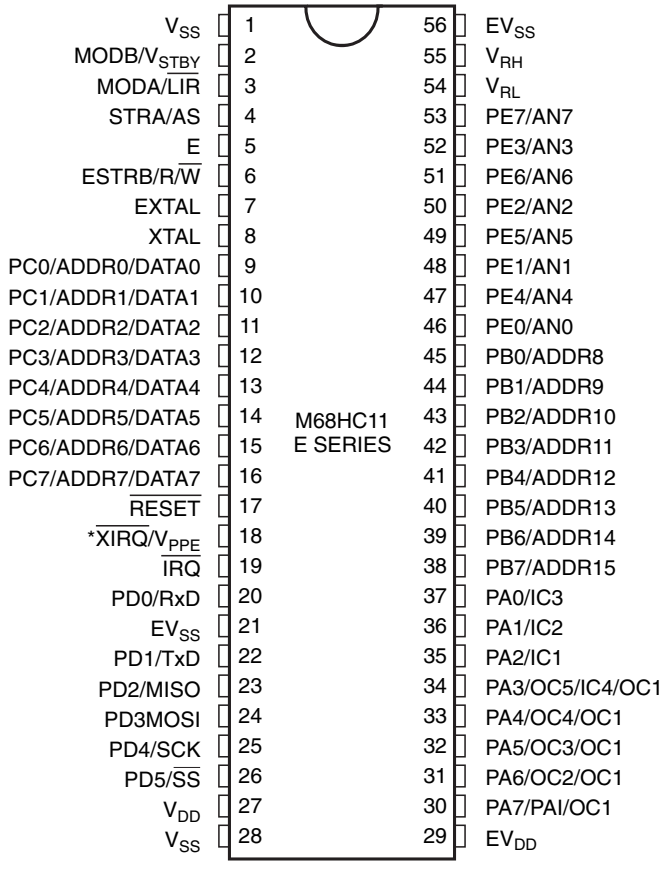

Figure 14.19 68HC11 in the 56-pin SDIP package.

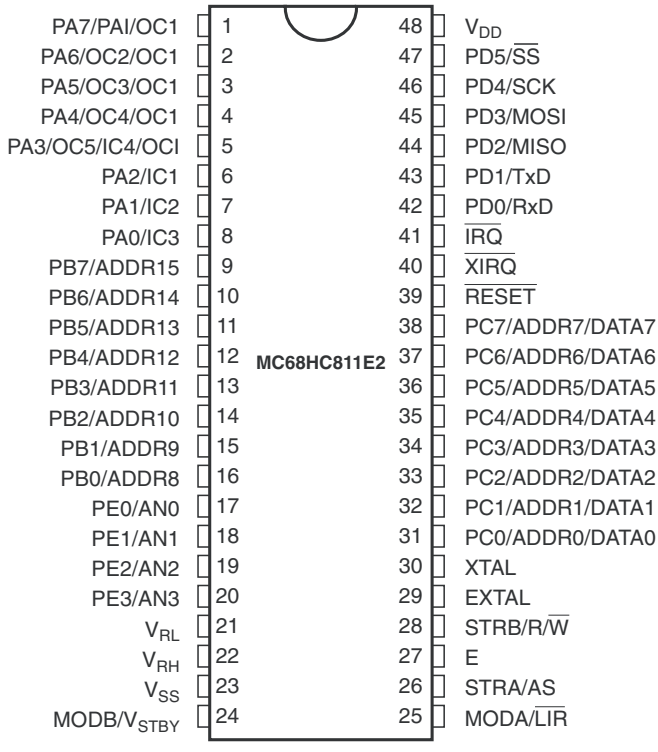

Figure 14.20 68HC11 in the 48-pin DIP package.

MHz (16C84/16F84), direct, indirect and relative addressing modes, power-saving SLEEP mode and four interrupt sources.

Peripheral-related Features

Thirteen I/O pins with individual direction control, high current sink/source for direct LED drive and eight-bit timer/counter with an eight-bit programmable prescaler.

Architecture and Pin Connection Diagram

Figure 14.21 shows the architecture. Figure 14.22 shows the pin connection diagram in the 18-pin DIP package.

14.5.1.10 XC-800 Family of Microcontrollers (Infineon)

The XC-800 family of microcontrollers offers high-performance eight-bit microcontrollers, with some of the members providing advanced networking capabilities by integrating both a CAN controller and LIN support on a single chip. Salient features of two of its members, i.e. XC-886/888 and XC-866, are briefly outlined in the following paragraphs.

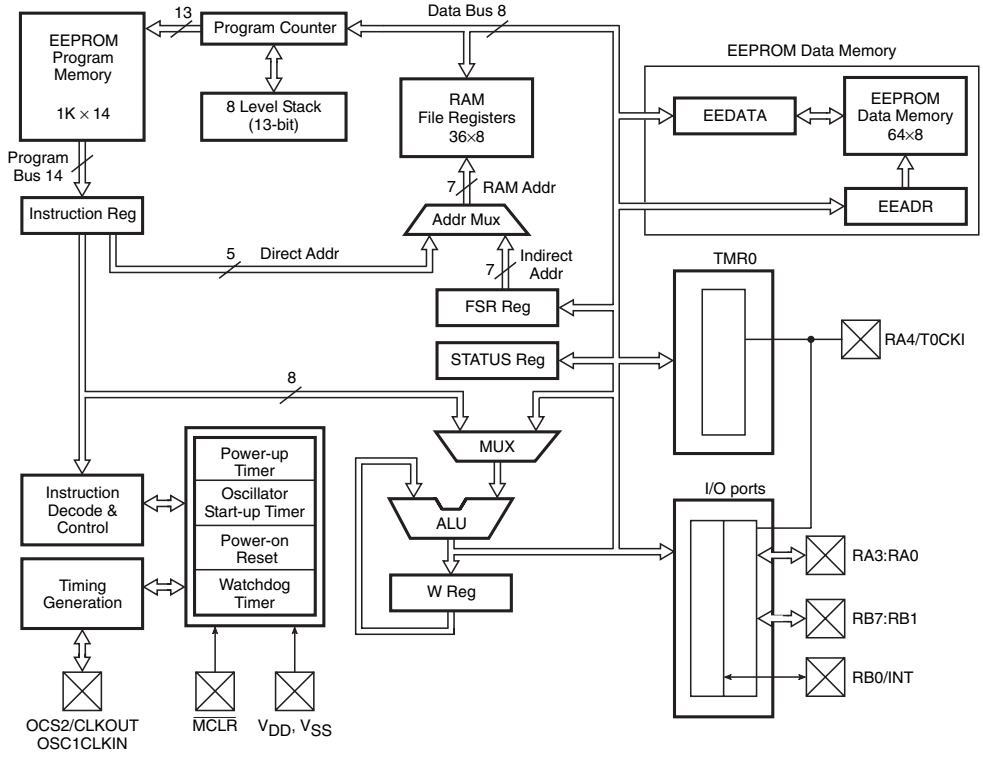

Figure 14.21 Architecture of the PIC 16X84 microcontroller family

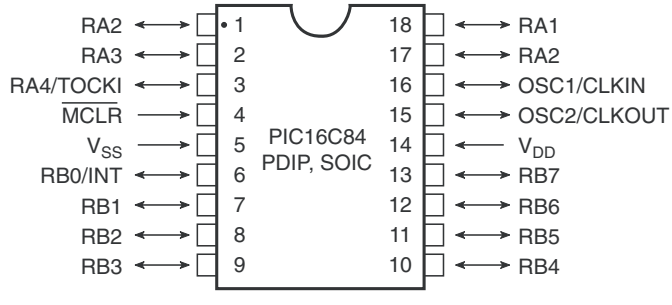

Figure 14.22 Pin connection diagram in the 18-pin DIP package.

Microcontroller-related Features

High-performance XC-800 core based on industry-standard 8051 architecture, a clock speed of 24 MHz, 24K or 32K of flash memory (XC-886/888), 256 bytes of RAM and 1536 bytes of XRAM (XC-886/888), 256 bytes of RAM and 512 bytes of XRAM (XC-866) and four power-saving modes including SLOW DOWN mode, IDLE mode, POWER DOWN mode and clock gating control.

Peripheral-related Features
Total of 34/48 general-purpose I/O ports, including eight analogue ports (XC-886/888) and 27 general-purpose I/O ports (XC-866), eight-channel, 10-bit analogue-to-digital converter, four 16-bit general-purpose timers (XC-886/888) and three 16-bit timers (XC-866), programmable 16-bit watchdog timer (WDT), two UARTs, including one for LIN simulation (XC-886/888) and one for LIN simulation, and one serial peripheral interface (XC-866).

14.5.2 16-Bit Microcontrollers

This subsection outlines salient features of some of the popular 16-bit microcontrollers. Again, the information is mainly contained under the headings microcontroller-related features and peripheral-related features.

14.5.2.1 68HC12 Family of Microcontrollers (Freescale Semiconductor)

Microcontroller-related Features
High-performance 16-bit CPU12 core having a 20-bit ALU, upward compatibility with the 68HC11 microcontroller instruction set, enhanced indexed addressing and fuzzy logic instructions, 1024 bytes of RAM, 32K of flash EEPROM and 768 bytes of EEPROM, a clock speed of 8 MHz, slow-mode clock divider, computer operating properly (COP) watchdog timer and available in 80-pin QFP and 112-pin TQFP packages.

Peripheral-related Features
Eight-channel, 10-bit analogue-to-digital converter, eight-channel, 16-bit input capture or output compare channels, up to 63 I/O lines, 16-bit pulse accumulator, eight-bit/four-channel or 16-bit/two-channel pulse width modulator, asynchronous serial communication interface (SCI) and synchronous serial peripheral interface (SPI).

Architecture and Pin Connection Diagram
Figure 14.23 shows the architecture of the 68HC12 family of microcontrollers. The pin connection diagram is shown in Fig. 14.24 (112-pin TQFP).

14.5.2.2 68HC16 Family of Microcontrollers (Freescale Semiconductor)

The 68HC16 family of microcontrollers is the 16-bit enhancement of the eight-bit 68HC11 family of microcontrollers. This family of microcontrollers has been designed to provide many powerful features without the need for CPU intervention.

Microcontroller-related Features
8K of ROM, 4K of RAM, clock speeds of 16, 20 and 25 MHz and available in 132-pin PQFP and 144-pin LQFP packages.

Peripheral-related Features
Twenty-four I/O lines, general-purpose timer, asynchronous serial communication interface (SCI) and synchronous serial peripheral interface (SPI).

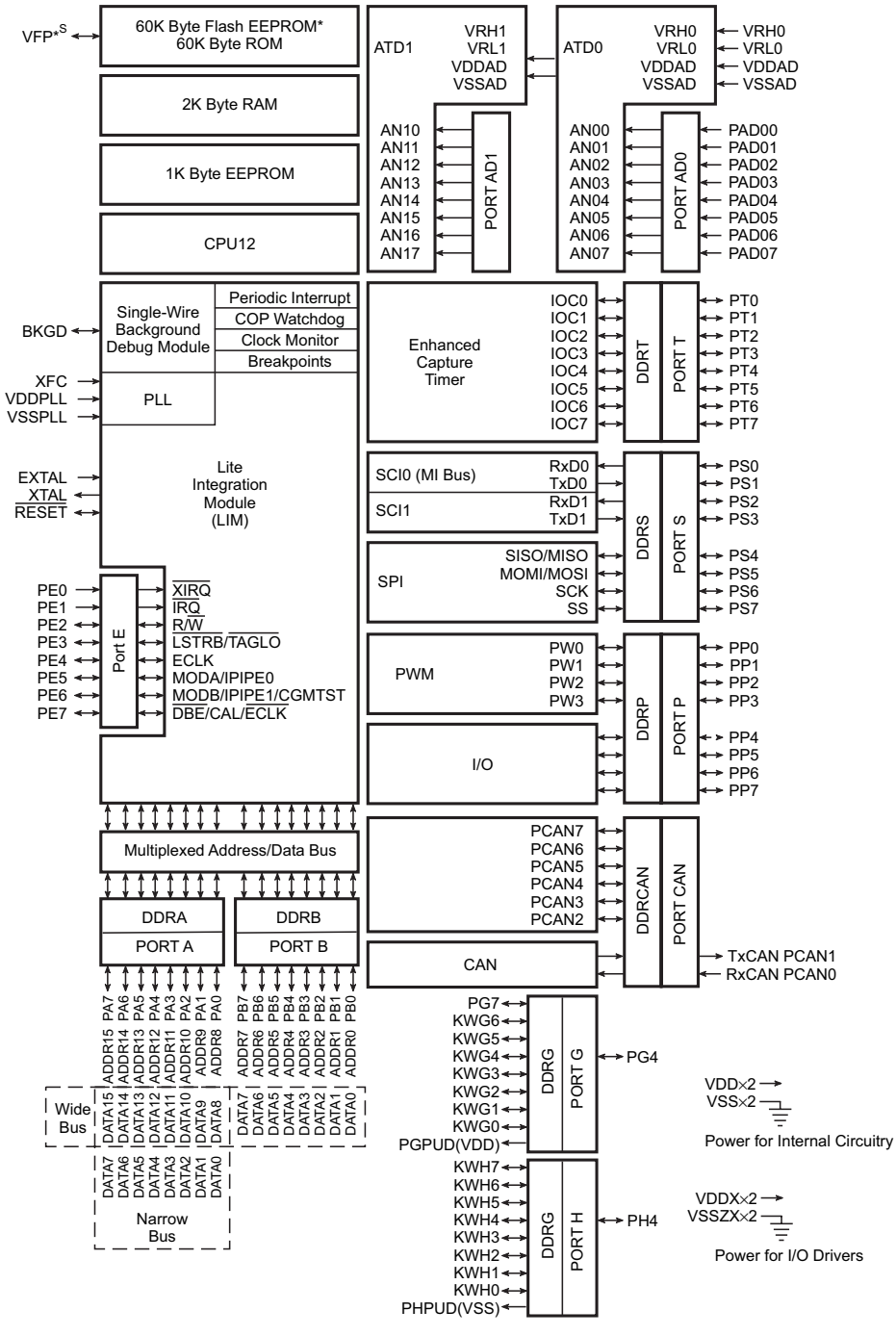

Figure 14.23 Architecture of the 68HC12 family of microcontrollers.

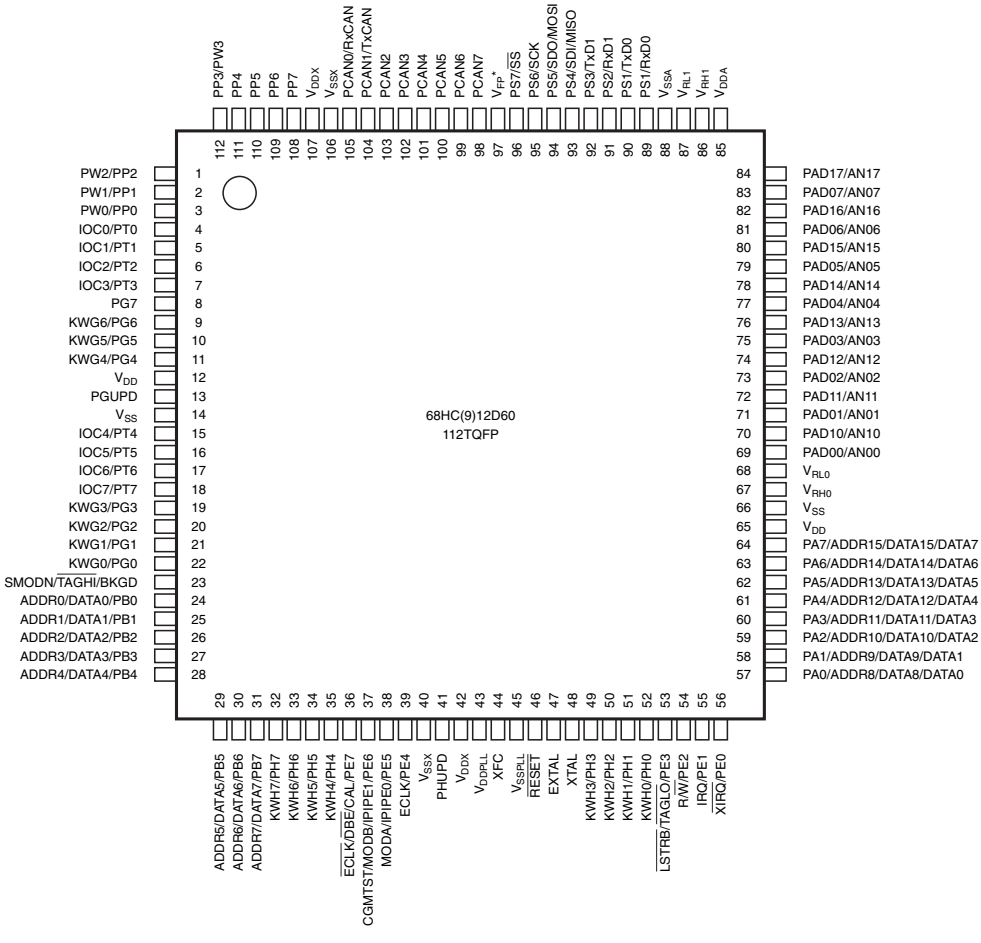

Figure 14.24 Pin connection diagram in the 112-pin TQFP package.

14.5.3 32-Bit Microcontrollers

This subsection outlines salient features of some of the popular 32-bit microcontrollers. The families of microcontrollers that are briefly described in the following paragraphs include 683XX, MCORE, MPC500 and MCFXXX families (Freescale), the LPC-3000 family (Philips Semiconductors) and the TRICORE family (Infineon).

14.5.3.1 683XX Family of Microcontrollers (Freescale Semiconductors)

Different members of this family include 68302, 68306, 68331/332/336, 68340, 68360 and 68375/376. 68302 uses an integrated multiprotocol processor. All other members of the family use a CPU32 core. The CPU32 core is a 32-bit processing unit based on the 68000 software model and instruction set with

some additional features from 68010 and 68020. It also has some new features added to the core for control operations. Salient features of this family of microcontrollers are as follows. The family offers 10K of RAM, 256K of flash, a clock speed of 33 MHz, 48 I/O lines, a 16-bit timer, a 16-channel/10-bit analogue-to-digital converter and four serial communication channels. It may be mentioned here that the above-mentioned values are the maximum available ones across the family of devices.

14.5.3.2 MCORE Family of Microcontrollers (Freescale Semiconductors)

This family of microcontrollers is built around a processing core known as the MCORE microRISC engine. The design of the core combines high performance with low power consumption, which makes the MCORE family of microcontrollers particularly suitable for battery-operated and mobile applications. Salient features of this family of microcontrollers are as follows. The family offers 32-bit wide load/store architecture, 16-bit wide instructions for fast instruction throughput between the core and the memory, 32 general-purpose registers and a four-stage instruction pipeline that facilitates most instructions to be completed in one clock cycle. Other features include 32K of RAM, 256K of flash, 33 MHz of clock speed, two serial communication channels, 104 I/O lines, an eight-channel analogue-to-digital converter and two timers. Again, the above-mentioned values are the maximum available ones across the family of devices.

14.5.3.3 MPC500 Family of Microcontrollers (Freescale Semiconductors)

The MPC500 family of microcontrollers is configured around a 32-bit PowerPC core. Different members of the family include MPC555, MPC556, MPC561, MPC562, MPC563, MPC564, MPC565 and MPC566. PowerPC architecture based design provides compatibility with the PowerPC instruction set, including floating-point operations. Salient features include 36K of RAM, 1024K of flash, a 66 MHz clock, three serial communication channels, 101 I/O lines, 40 channels of analogue-to-digital conversion and 70 timer channels. These microcontrollers are particularly suitable for scientific applications requiring complex operations.

14.5.3.4 MCFXXX Family of Microcontrollers (Freescale Semiconductors)

The MCFXXX family of microcontrollers is configured around a ColdFire Version 2 core. Different members of the family include MCF5206, MPC5207, MPC5208, MPC5211, MPC5212, MPC5213, MPC5214, MPC5216, MPC5232, MPC5233, MPC5234, MPC5235, MPC5249, MPC5270, MPC5271, MPC5272, MPC5274, MPC5275, MPC5280, MPC5281, MPC5282, MPC5327, MPC5328 and MPC5329. The core uses variable-instruction-length RISC architecture. ColdFire instructions, which are similar to those in the 680X0 instruction set, are processed in a pipelined architecture of fetch and decode/execute units. The core also contains an enhanced multiply-and-accumulate (eMAC) unit, which has been designed to support DSP applications. Other features include 64K of RAM, 66 MHz of clock, 5 serial communication channels, including an I^2C bus and CAN support, 150 I/O lines and four timer channels. Again, the above-mentioned values are the maximum available ones across the family of devices.

14.5.3.5 LPC3000 Family of Microcontrollers (Philips Semiconductors)

The LPC-3000 family of 32-bit microcontrollers is based on Philips' Nexperia platform. It is configured around an ARM926EJ core with the VFP9 floating-point coprocessor. The family offers enhanced

signal-processing performance with the 926EJ core equipped with features such as single-cycle multiply-accumulate packed data and saturating arithmetic. The vector coprocessor is a high-speed floating-point unit and is IEEE754 compliant.

The LPC3000 family of microcontrollers incorporates 32K of instruction cache and 32K of data cache, which operate concurrently owing to the use of Harvard architecture. The family combines high performance with low power dissipation, which is made possible by its low-voltage operation at 1.2 V. It operates at clock speeds in excess of 200 MHz and supports a wide range of peripherals. As an example, LPC3180 (the first member of the LPC3000 family of microcontrollers) has multiple serial interfaces including seven UARTs, two single master I^2C interfaces and two SPI controllers, USB on-the-go, a 32-bit general-purpose timer with a 16-bit prescaler with capture and compare capability, a watchdog timer, PWM blocks with an output rate of up to 50 kHz and up to 55 general-purpose I/O pins.

14.5.3.6 TRICORE™ Family of Microcontrollers (Infineon)

The TRICORE family of 32-bit microcontrollers uses a unified, single-core 32-bit microcontroller–DSP architecture optimized for real-time embedded systems. The architecture combines the real-time capability of a microcontroller with the computational power of a DSP and the high performance features of RISC load/store architecture. The TRICORE family of microcontrollers offers various subfamilies, which include the AUDO-NextGeneration family, the AUDO1 family, the TC116X family and the TC1130 family. The family offers clock speeds ranging from 40 MHz (AUDO1 family) to 150 MHz (AUDO NextGeneration family) and is equipped with almost every microcontroller-related and peripheral-related features in terms of on-chip memory, power-saving modes, serial interfaces, counters/timers, PWM blocks, I/O ports, A/D converters and so on.

14.6 Interfacing Peripheral Devices with a Microcontroller

This section briefly describes the interfacing of some common external peripheral devices with the microcontroller. The peripheral devices discussed in this section include LEDs, electromechanical relays, seven-segment displays, keypads, LCD displays and analogue-to-digital and digital-to-analogue converters. Only the basic fundamentals are discussed here. A detailed description of the software routines is beyond the scope of this book.

14.6.1 Interfacing LEDs

The commonly used configuration to connect an LED to a microcontroller is shown in Fig. 14.25(a). The LED glows when the microcontroller pin is driven LOW and is OFF when the pin is set HIGH. The LEDs are connected in this fashion as the current-sinking capability of microcontrollers is of the order of a few tens of milliamperes and the current-sourcing capability is of the order of microamperes. The resistor is used to limit the current through the LED.

The value of the resistance is chosen according to the equation

$$R = (V_{CC} - V_{LED})/I \tag{14.1}$$

where V_{LED} is the voltage across the LED and I is the current.

Typical values of V_{LED} and I are 1.5 V and 20 mA respectively. If the current-sourcing capability of the microcontroller is sufficient to drive the LED directly, then the LED is connected to the

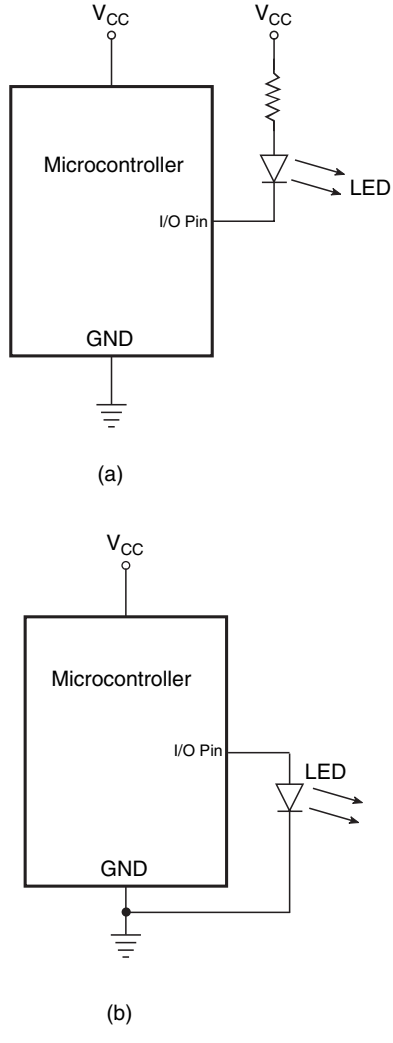

Figure 14.25 Interfacing LEDs to a microcontroller.

microcontroller as shown in Fig.14.25(b). The LED in this case glows when the microcontroller pin is set HIGH.

14.6.2 Interfacing Electromechanical Relays

Figure 14.26 shows the typical connection diagram for interfacing an electromechanical relay to a microcontroller. The NPN transistor is used to provide the desired current to the relay coil as the microcontroller cannot drive the relay directly. The freewheeling diode is required as the current

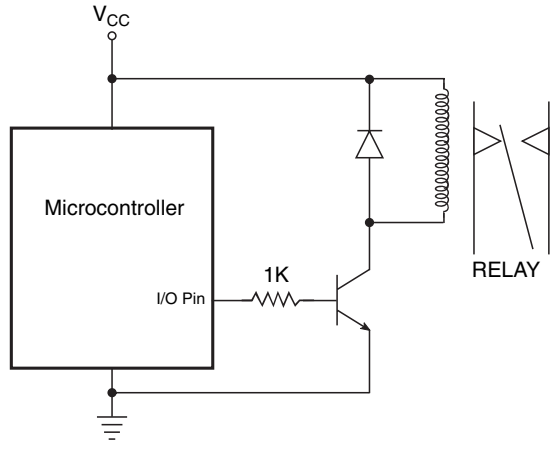

Figure 14.26 Interfacing an electromechanical relay to a microcontroller.

through the inductor cannot be instantaneously reduced to zero. When the microcontroller pin is set HIGH, the transistor is switched on. Current flows through the relay coil and the contact is closed. When the microcontroller pin is LOW, the transistor is switched off and the inductor current now flows through the freewheeling diode and slowly decays to zero value.

14.6.3 Interfacing Keyboards

Keyboards are used to enter data, values, etc., into the microcontroller system. They are generally available in three configurations, namely the lead-per-key keyboard, the matrix keyboard and the coded keyboard. Lead-per-key or linear keyboards are used when very few keys have to be sensed. Coded keypads are generally used in telephonic applications. They are high-quality durable keyboards and permit a multiple key press to be detected easily. They are used when the number of keys is 16 or less, as they are very expensive. The most commonly used keyboard is the matrix keyboard where the keys are arranged in a matrix, with keys in the same row and column sharing the same access lines. When the number of keys exceeds 10, more often than not matrix keyboards are used. Interfacing a matrix keyboard with the microcontroller is discussed in the following paragraphs.

When the keyboards are connected to a microcontroller, following factors must be considered:

1. *Contact bounce.* Contact bounce refers to multiple 'make' and 'break' oscillations of contact during the key-pressing operation (Fig. 14.27). Good-quality keyboards have bounce periods of 1–5 ms, whereas low-cost keyboards have bounce periods of tens of milliseconds. If the bounce is not taken into consideration, the microprocessor responds as if the key has been pressed and released several times when in fact it has been pressed only once. Contact debouncing through either a hardware or a software routine is done to avoid the undesirable multiple-contact effects during a key closure, so that it appears as a single ON or OFF operation. Hardware debouncing is done using an RC circuit [Fig. 14.28(a)] or a Schmitt trigger circuit [Fig. 14.28(b)]. If debouncing is done by a software routine, a delay of 20–50 ms is given after a key press before the routine for that key is executed.

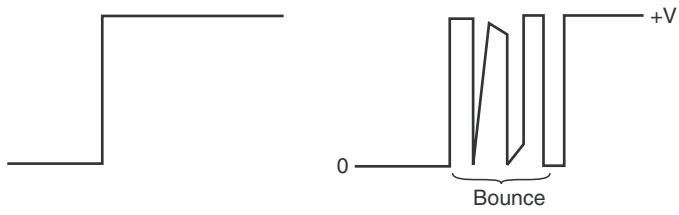

Figure 14.27 Contact bounce.

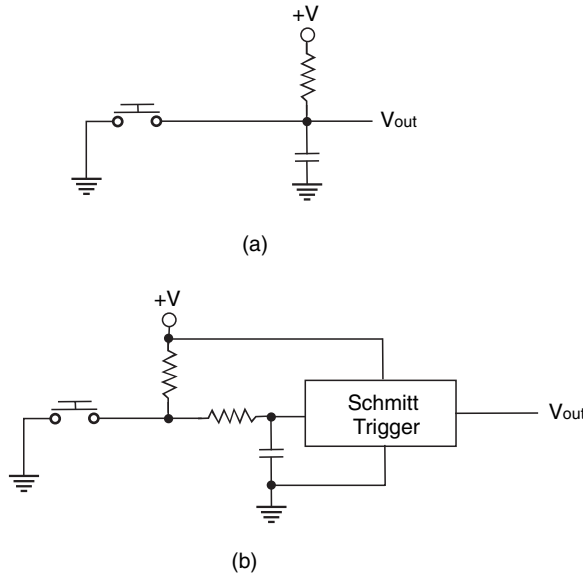

Figure 14.28 (a) Hardware debouncing using an RC circuit and (b) hardware debouncing using a Schmitt trigger circuit.

2. *Multiple keys.* If more than one key is pressed, then only routines corresponding to valid multiple-key presses should be executed. Also, the first valid key press pattern is executed.
3. *Key hold.* There are two types of keyboard actuation, namely the two-key lock-out and the *N*-key rollover. The two-key lock-out takes into account only one key pressed. An additional key pressed and released does not generate any codes. The system is simple to implement and most often used. The *N*-key rollover will ignore all keys pressed until only one remains down.

Figure 14.29 shows the connection of a 16-key matrix keypad with a microcontroller. Here, each column and row access line is connected to the microcontroller pin. The columns are generally at a HIGH level. The row lines are configured as output lines and the column lines are used as scan lines. The key actuation is sensed by sending a LOW to each row one at a time through a software routine via the row 1, row 2, row 3 and row 4 lines. The column lines are checked for each row to see

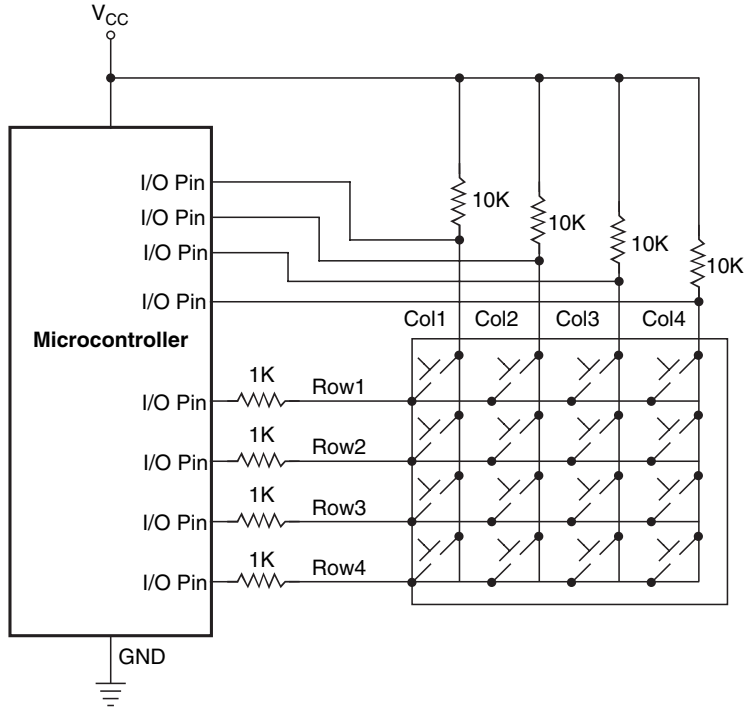

Figure 14.29 Connection of a 16-key matrix keypad with a microcontroller.

whether any of the normally HIGH column lines are pulled LOW. If a column is driven LOW, then, by determining which row and column line is LOW, the key is identified and the routine corresponding to that key press is executed.

14.6.4 Interfacing Seven-Segment Displays

Seven-segment displays commonly contain LED segments arranged as a figure-of-eight pattern, with one common lead (anode or cathode) and seven individual leads for each segment. When the common lead is the anode it is referred to as the common anode (CA), and when the common lead is the cathode it is referred to as the common cathode (CC). Figure 14.30 shows one of the possible configurations of interfacing a CC display with the microcontroller. The IC CD4511 is a BCD to seven-segment decoder/driver. The microcontroller feeds the BCD equivalent of the digit to be displayed to the 4511 IC.

Seven-segment displays can also be connected directly without the use of a BCD to seven-segment decoder. In this case the seven-segment code of the digit is generated by the microcontroller program itself. Figure 14.31 shows the direct circuit connection for CA display.

If more than one display is to be used, the displays are time multiplexed. The human eye cannot detect the blinking display if each display is relit every 10 ms or so. The 10 ms time is divided by the number of displays used to find the interval between updating each display. In the case of CC displays

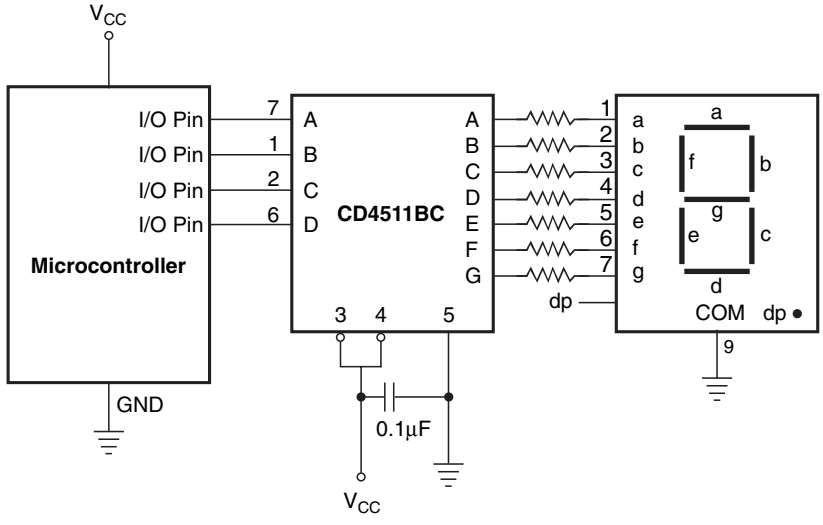

Figure 14.30 Possible configurations of interfacing a CC display with a microcontroller.

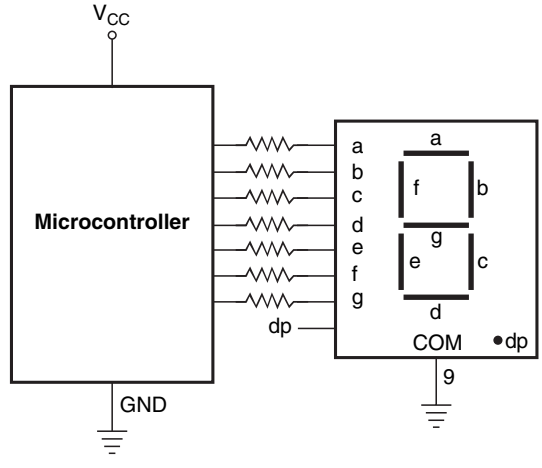

Figure 14.31 Direct circuit connection for CA display.

the display is selected by driving the common cathode to logic LOW, and in the case of CA displays the display is selected by driving the common anode to logic HIGH. Figure 14.32 shows the multiplexed circuit for two CC displays. The IC 74138 is a 3-to-8 line decoder used for selecting the display. Figure 14.33 shows the multiplexing in the case of CA displays for direct connection without the use of a BCD to seven-segment driver.

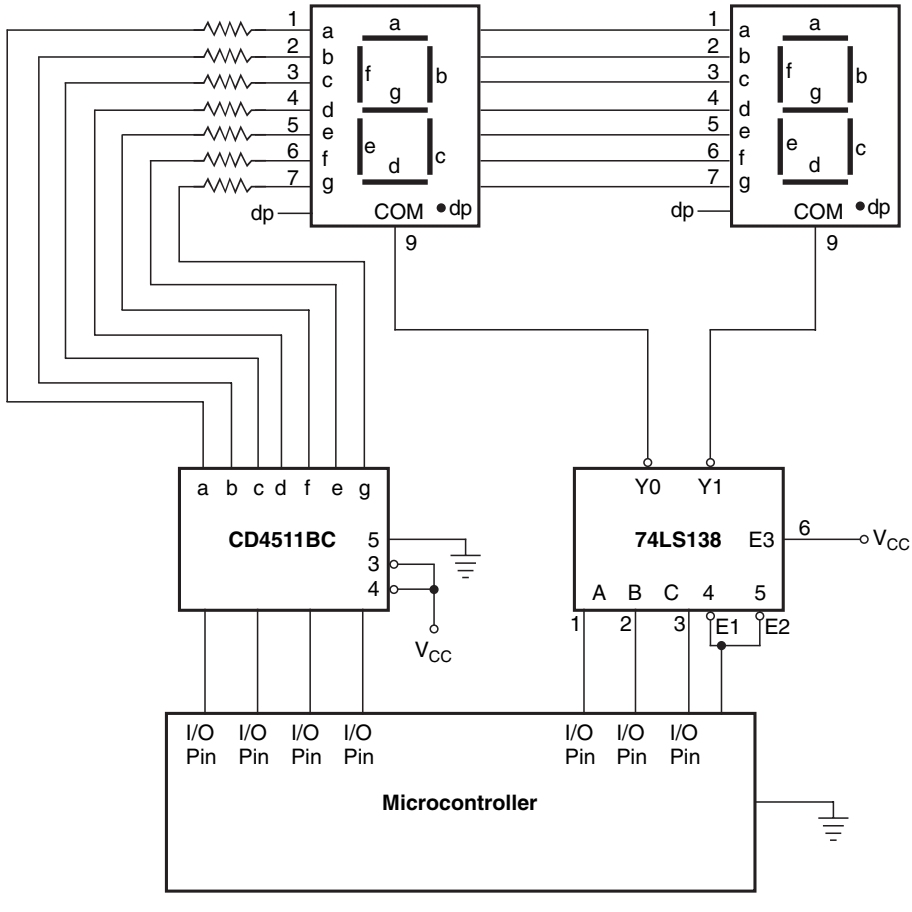

Figure 14.32 Multiplexed circuit for two CC displays.

14.6.5 Interfacing LCD Displays

Liquid crystal displays allow a better user interface compared with LED displays as it is much easier to display text messages in LCD displays. They also consume much less power than LED displays. However, LED displays have better intensity than LCD displays.

LCD displays are available typically in 8×2, 16×2, 20×2 or 20×4 formats. 20×2 means two lines of 20 characters each. These displays come with an LCD controller that drives the display. Figure 14.34 shows the interface of an LCD display with a microcontroller. There are three control lines, namely EN (enable), RS (register select) and RW (read/write). The EN line is used to instruct the LCD that the microcontroller is sending the data. When the RS line is HIGH, the data comprise text data to be displayed on the LCD. When the RS is LOW, the data are treated as a command or instruction to

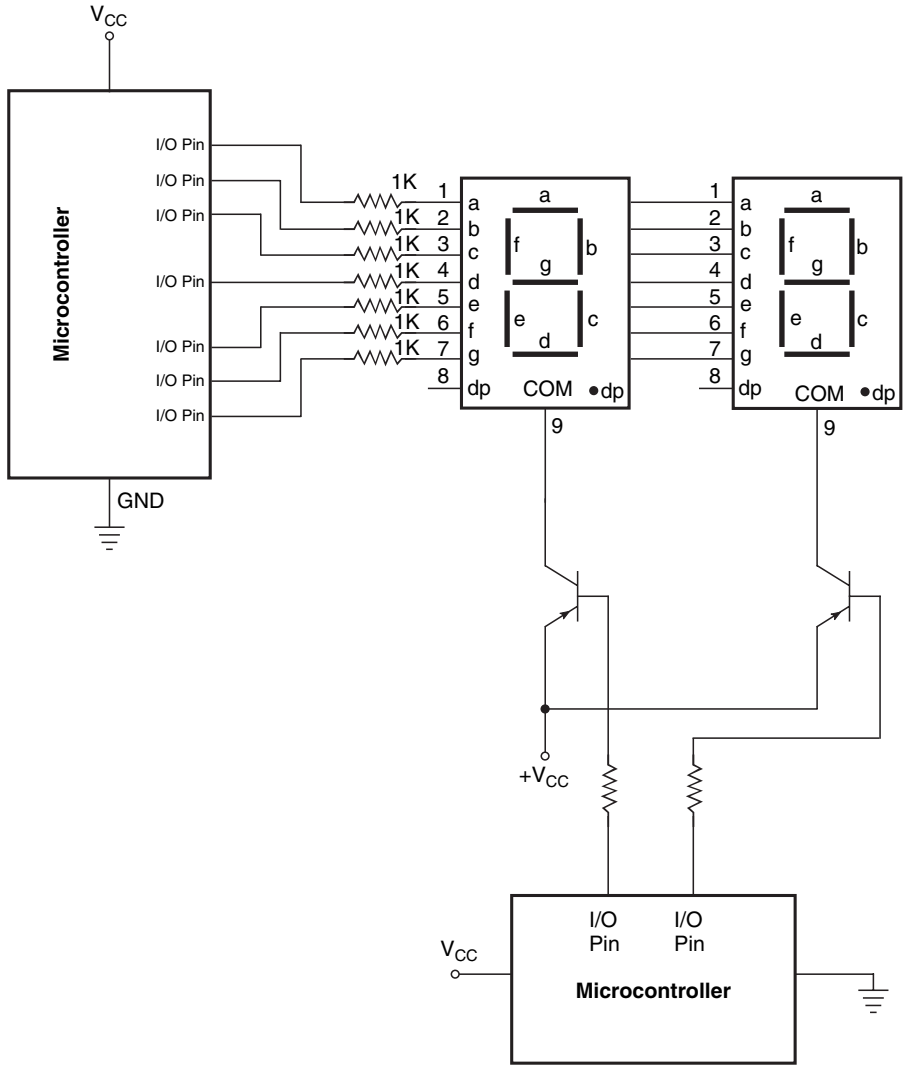

Figure 14.33 Multiplexed circuit for two CA displays for direct connection.

the LCD module. When the RW line is LOW, the instruction on the data bus is written on the LCD. When the RW line is HIGH, the data are being read from the LCD.

The software routine initializes the LCD firstly by setting the width of the data bus, selecting the character, font, etc., clearing the LCD, turning on the LCD module and the cursor, setting the cursor position and so on. Then the data to be displayed are sent on the data lines, and the three control signals are made use of to ensure proper LCD operation.

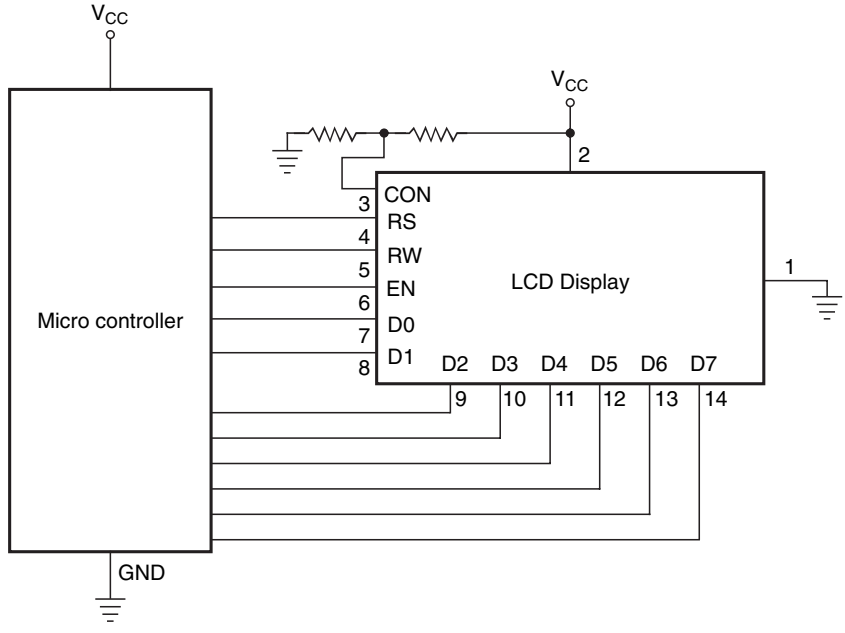

Figure 14.34 LCD display interface with a microcontroller.

14.6.6 Interfacing A/D Converters

A/D converters are used to interface the microcontroller with the analogue world. Figure 14.35 shows the interface of A/D converter type number AD571 with the microcontroller. AD571 is an eight-bit A/D converter. As can be seen from the figure, the data output lines and the control lines of the A/D converter are connected to the microcontroller I/O pins. The microcontroller sends commands such as the start of conversion, selection of the input channel if the A/D converter has more than one input channel, etc. It also senses signals from the A/D converter such as the end of conversion to store the digital bits. In the present case, the microcontroller sends a LOW on the $BLANK/\overline{DR}$ line to start the conversion process. It then waits for the data ready $(\overline{DR})$ signal to go to LOW. After that, the digital output bits are received by the microcontroller and processed according to the software routine.

14.6.7 Interfacing D/A Converters

When interfacing a D/A converter to the microcontroller, the digital data lines and the control lines, such as the start of conversion and chip select lines, are connected to the microcontroller I/O pins. The software routine generates the required signals to start the conversion process. Figure 14.36 shows the interface of D/A converter type number DAC 809 with the microcontroller. DAC-809 is a eight-bit D/A converter. Here, the output is current, so a current-to-voltage converter is required at the output.

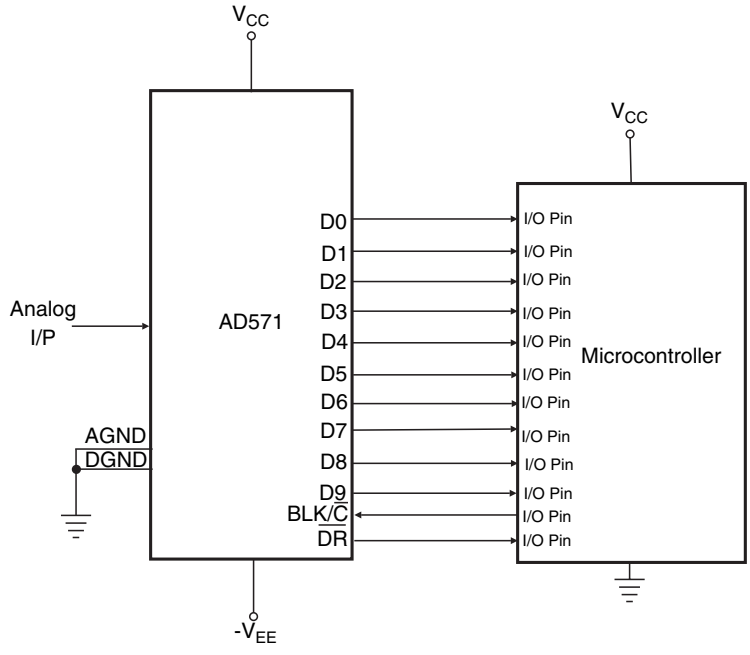

Figure 14.35 Interface of an A/D converter with a microcontroller.

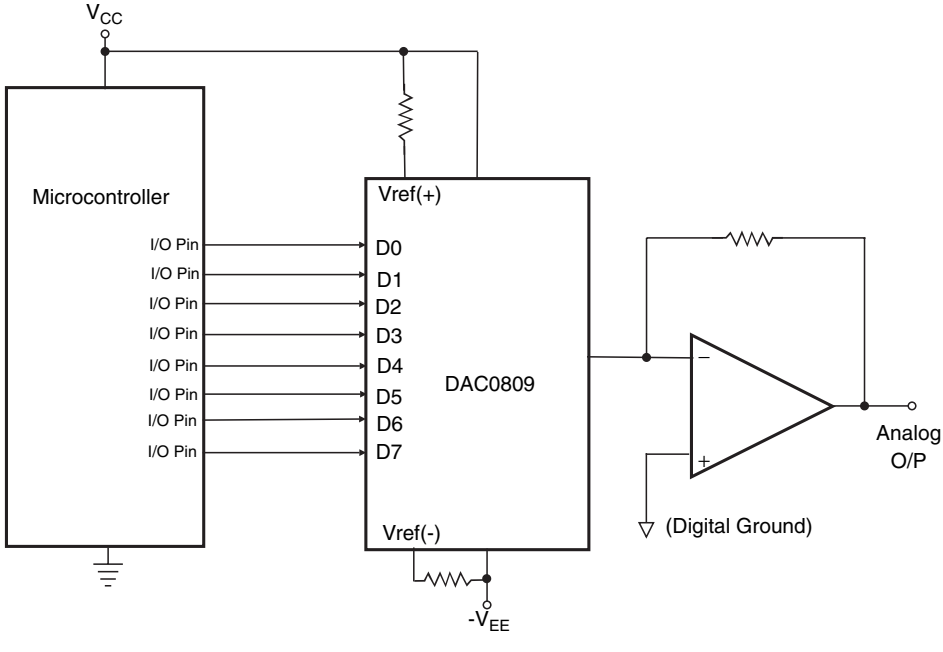

Figure 14.36 Interface of a D/A with a microcontroller.

Review Questions

1. What are the fundamental differences between a microprocessor and a microcontroller? Briefly describe some of the major application areas of microcontrollers.
2. What hardware components are likely to be found inside a typical microcontroller? Briefly describe the function of each one of them.
3. Name any three serial interfaces that are usually found on microcontrollers. Briefly describe where these are generally used.
4. What are the salient features of:

   (a) an interintegrated circuit (I^2C) bus;
   (b) Harvard architecture;
   (c) a memory-mapped I/O.

5. Briefly describe the salient features of the 80C51 family of eight-bit microcontrollers. Which microcontroller from Freescale Semiconductor does it closely resemble? Why and when would one like to choose a microcontroller other than 8051?
6. With reference to internal architecture, how do you compare eight-bit, 16-bit and 32-bit microcontrollers?
7. What are the basic differences between

   (a) the 80C51 and 89C51 families of microcontrollers;
   (b) the 68HC11 and 68HC16 families of microcontrollers;
   (c) the 80C51 and 16C84 families of microcontrollers.

8. With the help of relevant diagrams, briefly explain the difference between interfacing an LED type of display and an LCD type of display to a given microcontroller.
9. What are the interface requirements on the part of the microcontroller if it were to be interfaced with:

   (a) a keypad;
   (b) an LED;
   (c) another microcontroller.

Problems

1. A microcontroller with an eight-bit counter/timer system is used to measure the width of an input pulse. The microcontroller has been programmed to measure the time of occurrence of rising and falling edges of an input pulse on a certain I/O pin. If the microcontroller uses a 10 MHz clock and the count values observed at the time of occurrence of rising and falling edges of the input pulse are FE and 9A (in hex), determine the pulse width as measured by the microcontroller.

   10 μs

2. A microcontroller with a 16-bit counter/timer system is used to measure the frequency of an input pulse train. The microcontroller has been programmed to measure the time of occurrence of two successive leading edges of the input pulse signal on a certain I/O pin. If the microcontroller uses an 8 MHz clock and the count values observed at the time of occurrence of two successive rising

edges are observed to be FEEA and FE86 (in hex), determine the pulse width as measured by the microcontroller.

80 kHz

3. It is desired to design a microcontroller-based periodic signal generator with minimum and maximum time period specifications of 50 ns and 150 ms. Determine the minimum clock speed requirement of the microcontroller.

20 MHz

Further Reading

1. Susnea, L. and Mitescu, M. (2005) *Microcontrollers in Practice*, Springer Series, Springer, Germany.
2. Predko, M. (1999) *Programming and Customizing the 8051*, McGraw-Hill Professional, USA.
3. Van Sickle, T. (2000) *Programming Microcontrollers in C*, Elsevier Science, MA, USA.
4. Predko, M. (1998) *Handbook of Microcontrollers*, McGraw-Hill/Tab Electronics, USA.