26

Computers

A. M. MacLeod
University of Abertay
Dundee, Scotland

P. F. Martin
University of Abertay
Dundee, Scotland

W. A. Gillespie
University of Abertay
Dundee, Scotland

26.1 Introduction

Computers are an essential feature of most instrumentation systems because of their ability to supervise the collection of data and allow information to be processed, stored, and displayed. Many modern instruments are capable of providing a remote user with access to measurement information via standard computer networks.

26.2 Computer-Based Instrumentation Systems

The main features of a computer-based instrumentation system are shown in Figure 26.1. The actual implementation of such systems will depend on the application. Many commercially produced instruments such as spectrophotometers or digital storage oscilloscopes are themselves integrated computerbased measurement systems. These "stand-alone" instruments may be fitted with interfaces such as IEEE- 488 or RS-232 to allow them to be controlled from personal computers (PCs) and also to support the transfer of data to the PC for further processing and display. Alternatively, the instrumentation system may be based around a PC/workstation or an industrial bus system such as a VME or Multibus, allowing the user the ability to customize the computer to suit the application by selecting an appropriate set of add-on cards. Recently, the introduction of laptop and notebook PCs fitted with PCMCIA interfaces with input/output capability has provided opportunities for the development of highly portable instrumentation systems.

The Single-Board Computer

The simplest form of a computer is based around the single-board computer (SBC) which contains a microprocessor, memory, and interfaces for communicating with other electronic systems. The earliest

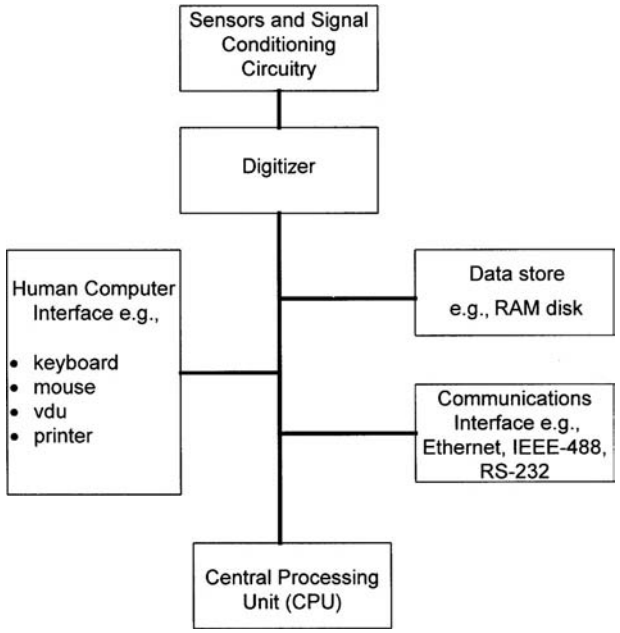

FIGURE 26.1 Elements of a computer-based instrumentation system.

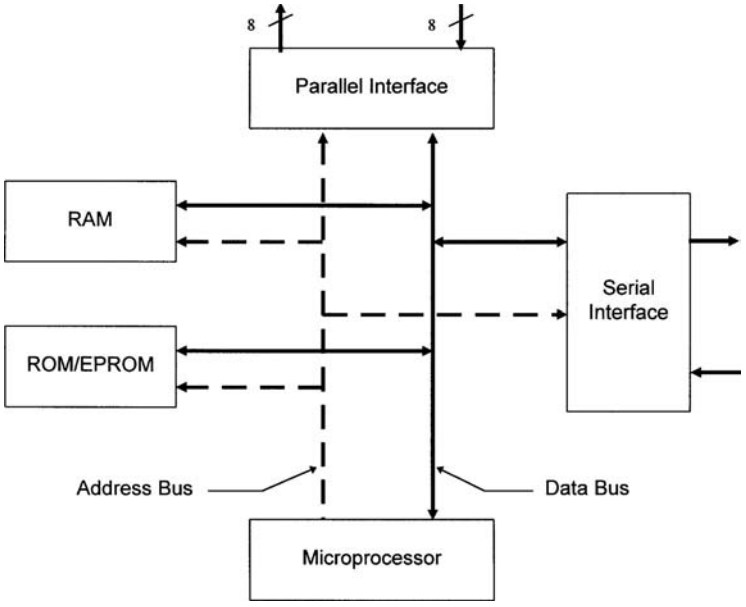

FIGURE 26.2 An overview of a single-board computer.

form of personal computers simply comprised an SBC together with a keyboard, display, disk drives, and a power supply unit. Today, the SBC still offers a solution for the addition of limited intelligence to instrumentation systems as well as forming an element of most computer systems, e.g., as the motherboard of a PC.

An overview of a simple SBC is shown in Figure 26.2. The microprocessor, which contains the central processor unit, is responsible for executing the computer program and for controlling the flow of data around the SBC. The random access memory (RAM) acts as a temporary (i.e., volatile) storage area for

both program code and data. On the motherboard of a PC the read only memory (ROM) is largely used for storing the low-level code used to access the input/output hardware, e.g., the BIOS (basic input output system). The operating system and applications software are loaded into RAM from the disk unit. In small, dedicated systems such as an oscilloscope the ROM may be used to store the program code to avoid the need for a disk drive. For small production runs or development work erasable programmable ROM (EPROM) is used as an alternative to ROM, allowing the code to be upgraded without the high cost of commissioning a new ROM chip from the manufacturers.

Data are transferred around the SBC on its data bus, which will be typically 8, 16, or 32 bits wide (corresponding to the number of bits that can be transmitted at the same time). SBCs with 8-bit data buses are less complex and consequently of lower cost than 32-bit systems and may well be the optimum solution for those process control applications which require minimal processing of 8-bit data, e.g., from temperature and position sensors. However, the 32-bit bus width of most modern PCs and workstations is essential to ensure the fast operation of Windows-based applications software.

The address bus is used to identify the destination of the data on the data bus. Data transfer is usually between the microprocessor and the memory or interfaces. However, some SBCs support DMA (direct memory access) which allows data to be transferred directly between interfaces and memory without the need for the information to pass through the processor. DMA is inherently faster than program-driven data transfer and is used for moving large blocks of data, e.g., loading programs from disk or the transfer of digitized video images.

SBCs are fitted with interfaces to allow them to communicate with other circuits. Interfaces carry out two main functions. First, they serve to ensure that the signals on the internal buses of the SBC are not affected by the connection of peripheral devices. Second, they ensure that signals can pass into and out of the computer and that appropriate voltage levels and current loading conditions are met. A well-designed interface should also provide adequate electrical protection for the SBC from transients introduced via the external connection. Parallel interfaces allow data to be transferred, usually 8 or 16 bits at a time, and contain registers that act as temporary data stores. Serial interfaces must carry out the conversion of data from the parallel format of the internal SBC data bus to and from the serial format used by the interface standard (e.g., RS-232 or RS-422). Some SBCs may contain additional interfaces to support communication with a VDU, local area network (LAN), or a disk drive. Interfaces can range in complexity from a single parallel interface chip to a LAN controller which may require a significant fraction of the SBC board area.

Computer Bus Architectures

All but the simplest computer systems contain several circuit board cards which plug into a printed circuit board backplane. The cards will include at least one SBC and a number of "add-on" cards providing functions such as interfaces to peripherals (e.g., a LAN, graphics display, disk unit) or additional memory. The actual structure of bus architectures is quite variable but the simple model shown in Figure 26.3 contains the essential features. A group of tracks will carry the data and address information with a second group of tracks being used to control the flow of data and to ensure its reliable transfer. Other

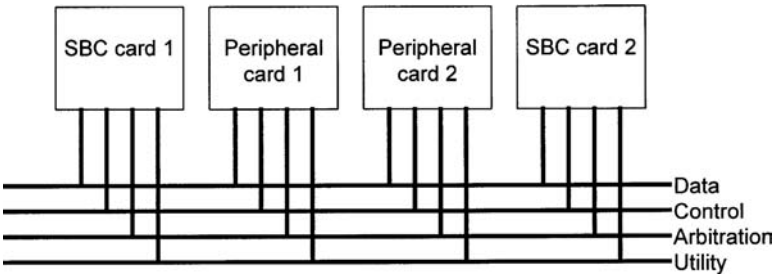

FIGURE 26.3 A simplified model of a computer bus.

TABLE 26.1 Typical Microprocessor Bus Standards

Bus Standard	STE	G96	VME	Multibus II
Data width (bits)	8	8/16	8/16/32	8/16/32
Max address (bytes)	1 M	32 M	4 G	4 G
Synchronous/asynchronous	Async	Async	Async	Sync
Connectors	64 pin	96 pin	96 pin	96 pin + 32 pin

tracks are reserved for the signals which provide arbitration between SBC cards to ensure that only one such card has control of the bus at any given moment. There will also be tracks which provide utility functions such as clock signals and the transmission of event or interrupt signals.

The main advantage of bus-based systems is that they help one build a complex computerized system using standard cards. By conforming to agreed standard buses (see Table 26.1 for typical examples), the system integrator can minimize problems of incompatibility and keep system costs to a minimum. The complexity of many bus systems is such that a significant fraction of the area of each card is dedicated to providing the logic required to interface to the bus, adding appreciably to the cost. In addition, data transfer between cards, even using DMA, is relatively slow compared with transfers between chips on the same SBC. There is, therefore, a trade-off between integrating functions on a single SBC and a lack of flexibility in customizing the system. An alternative to bus-based systems is to utilize processors such as transputers which are designed to be connected together directly into multiprocessor structures.

Most modern computerized systems adopt some form of distributed intelligence strategy in order to free the central processor unit to run the main application. Graphics controller cards generally employ a dedicated graphics processor able to handle the creation of the basic features such as line drawing or scaling the size of objects. LAN controller cards require intelligence to handle the network protocols and buffer the data flow to and from the network. Disk controller cards relieve the central processor of the need to handle the processes of reading and writing to the disk.

The use of a bus system with more than one SBC allows the designer to dedicate an SBC to the process of data acquisition and temporary data storage in order to ensure that the main SBC can concentrate on data processing and communication with the user. Such a strategy enables the data acquisition SBC to maintain a real-time response to making measurements while allowing the user access to processed data. In most systems one of the SBCs acts as a system controller, which has ultimate control over the use of the bus and will normally contain the bus arbitration hardware. Only one SBC at a time can obtain the authority of the controller to act as a bus master and initiate data communication. The other cards will be configured as slaves allowing them to respond to requests for information from the master. In some systems, e.g., Multibus II, bus arbitration is distributed throughout the intelligent cards.

Industrial Computers

Many instrumentation systems utilize standard PCs or workstations either fitted with add-on cards or linked to intelligent instruments via standard protocols such as IEEE-488 or RS-232. In many industrial environments there is a need to provide protection against hazards such as dust, damage by impact or vibration, and unauthorized access. The systems may have to fit more stringent electromagnetic compatibility (EMC) requirements. Industrial or ruggedized versions of desktop computers are available to meet this market. Cardframes designed to accommodate the standard industrial bus systems such as VME and Multibus can be readily customized to meet the demands of the industrial environment and ruggedized versions of PCs are also available. In designing industrial computers, care must be paid to the specification of adequate power supplies, both to provide sufficient current for the add-on cards and also to provide protection against fluctuations in the mains supply. It may also be necessary in safety-critical applications to use an uninterruptable power supply that will guarantee the operation of the system during short failures in the mains supply.

Software

All but the simplest computer systems require an operating system to support the operation of applications software. The operating system will allocate areas of memory for use by the applications programs and provide mechanisms to access system resources such as printers or displays. Some operating systems, such as MS DOS, are single-tasking systems, meaning that they will only support the operation of one program or task at a time. Instrumentation systems which simultaneously take measurements, process data, and allow the user to access information generally require a multitasking operating system, such as OS-9, UNIX, or Windows 95. In such instrumentation systems the applications software may well comprise a single program, but it may generate software processes or tasks each with their own local data and code. The multitasking operating system will allow the actual execution of these tasks to be scheduled and also provide mechanisms for communication of information between tasks and for the synchronization of tasks.

Instrumentation systems are real-time environments, i.e., their operation requires that tasks be carried out within specified time intervals; for example, the capture of data must occur at specified moments. Operating systems adopt a range of strategies for determining the scheduling of software tasks. Round-robin scheduling, for example, provides each task with execution time on a rota basis, with the amount of time being determined by the priority of the task. Operating systems which support preemptive scheduling allow high-priority tasks to become active if a specified event such as trigger signal occurs. The speed at which an operating system can switch from one task to another (i.e., context switch) is an important metric for real-time systems.

The extent to which an operating system can respond to a large number of events is extremely limited, as low-priority tasks will have a poor response time. If an instrumentation system needs to make a large number of measurements and also support other activities, such as information display, the solution is generally to use a distributed system with dedicated SBCs operating as microcontrollers to capture the data. These microcontrollers will not usually require an operating system and will each operate a single program which will be either downloaded from the master SBC or located in ROM.

Commercial packages (for examples, see Table 26.2) are readily available to support PC/workstation-based instrumentation systems. A typical system will provide a Windows-type environment to control measurements and store, process, and display data and increasingly make use of the virtual-instrument concept which allows the user to configure instrumentation systems from an on-screen menu. Most packages will allow stand-alone instruments to be controlled using interface cards supporting standards such as IEEE-488 or RS-232. Some packages also support the use of add-on cards for analog-to-digital conversion or a range of control functions such as channel switching and waveform generation; however, their main restriction is the limited range of hardware configurations supported by the software supplier. Each stand-alone instrument and add-on card requires a piece of code called a device driver so that the operating system can access the hardware resources of the card and instrument. Therefore, the development of device drivers requires an intimate knowledge of both the hardware and the operating system.

System Development

The software component of any computerized instrumentation system can form a significant percentage of its total cost. This is especially true of on-off systems which require software to be developed for a specific application, where the cost of the implementation of the software and its maintenance can considerably exceed the hardware costs. In such circumstances the ability to reuse existing code and the access to powerful development systems and debugging tools are of crucial importance. Some debug software only provides support for stepping through the operation of code written at the assembly code level. When developing device drivers, for example, access to a source-level debugger which can link the execution of code written in a high-level language to the contents of processor registers is useful.

Many SBCs that are intended for operation as stand-alone microcontrollers, instead of in bus-based systems, are often supported by a high-level language such as C or FORTH and a development environment. A common practice is to utilize a PC to develop the code which can be downloaded via a serial

TABLE 26.2 Typical Data Acquisition and Display Software Packages for PC-Based Instrumentation Systems

Product	Platforms	Manufacturer
DTV	PC (Windows 3.1 or 95)	Data Translation Inc. 101 Locke Drive Marlboro, MA 01752-1192 Tel: (U.S. and Canada) (800) 525-8528 @em@em(worldwide) +1 508-481-3700
Labtech Notebook/Notebook Pro	PC (DOS, Windows 3.1 or 95)	Laboratory Technologies Corporation 400 Research Drive Wilmington, MA 01887 Tel: (U.S. and Canada) 800-8799-5228 @em@em(worldwide) +1 508-658-9972
LabVIEW	PC, Mac, Sun, Power PC	National Instruments 6504 Bridge Point Parkway Austin, TX 78730-5039 Tel: +1 512-794-0100
LabWindows	PC (DOS)	National Instruments 6504 Bridge Point Parkway Austin, TX 78730-5039 Tel: +1 512-794-0100
LabWindows/CVI	PC (Windows 3.1 or 95) Sun	National Instruments 6504 Bridge Point Parkway Austin, TX 78730-5039 Tel: +1 512-794-0100
ORIGIN	PC (Windows 3.1 or 95)	MicroCal Software Inc. One Roundhouse Plaza Northampton, MA 01060 Tel: +1 413-586-2013

link to the microcontroller. Some debugging support for the execution of the code on the microcontroller is also provided. The development of powerful compact Intel 486 or Pentium microcontroller cards running operating systems such as MS DOS can greatly reduce software development time because of the ready access to PC-based software. The implementation of systems based on dedicated microcontrollers may require the use of a logic analyzer to view data on system buses.

26.3 Computer Buses

The VMEbus (IEEE P1014)

The VMEbus [1–3], utilizes a backplane fitted with two 96-pin connectors (called P1 and P2). The P1 connector provides access to all the bus signals with the exception of bits 16 to 31 of the data bus and bits 24 to 31 of the address bus, which are provided on the P2 connector. The location of cards in a VME system is important, as slot 1 must contain the system controller and the priority of the SBCs is determined by their proximity to the system controller SBC. As only 32 of the pins on the P2 connector are defined, the remaining 64 tracks on the P2 backplane may be specified by the user. VME cards may be single height (fitted with a P1 connector only) or double height (fitted with both P1 and P2 connectors). Single-height boards are 100 mm high and 160 mm deep and fit 3U height cardframes. Double-height boards are 233.35 mm high and 160 mm deep and fit 6U height cardframes.

VMEbus Signals

The VMEbus can be described as having four distinct groups of signals.

Data Transfer Bus (DTB).
The DTB provides 32-bit data transfer with 32-bit address information using a nonmultiplexed asynchronous approach. Transfer is supported by a simple handshake mechanism with the bus master

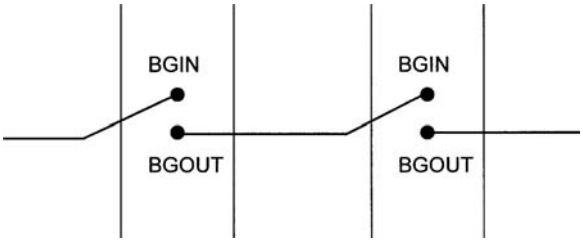

FIGURE 26.4 VMEbus arbitration daisy chain.

initiating the transfer providing an address strobe (AS) signal to tell the destination card to read the address information. The master also controls two data strobe (DS1 and DS2) lines which in a write operation indicate the presence of valid data on the data bus and in a read operation that the master is ready to receive data from the slave. The slave card uses the data acknowledge signal (DTACK) in a read operation to indicate that it has placed valid data on the bus and in a write operation to confirm that it has read the data. There is clearly a danger that the bus will hang up if the slave does not respond to a request for a data transfer, so in most VME systems a watchdog timer is provided to produce a signal on the bus error (BERR) line if a DTACK signal is not detected within the time-out period.

Revision C of the VME bus specification will support 8-,16-, and 32-bit data transfers, whereas revision D supports 64-bit transfers by making use of the address bus to transfer the additional 32 bits. Data may also be transferred in blocks of up to 256 bytes without the need to retransmit the address information. Most SBCs have both master and slave interfaces allowing their memory to be accessed from other SBCs in the system. The VME bus also supports three types of addressing mode, short (16 bit), standard (24 bit), and extended (32 bit), allowing the memory decoding circuitry of cards to be minimized for small systems or for input/output cards. Six address modifier lines (AM0 to AM5) are used to indicate the addressing mode and also to give information about the mode of transfer (e.g., block transfer and whether the transfer is in privileged, supervisor, or nonprivileged mode).

Arbitration Bus.
The arbitration bus provides a mechanism for deciding which master is allowed to gain control of the bus. The arbitration bus provides four bus request lines (BR0 to BR3) and four bus grant lines. The SBC in slot 1 of the VME bus acts as a bus controller which provides arbitration using a number of scheduling algorithms. In a simple priority-based algorithm each of the bus request lines is assigned a priority, whereas in a round-robin system access to the bus cycles around the bus request lines. Each of the four bus grant lines is in fact allocated two pins on the P1 connector (see Figure 26.4). The bus grant signal therefore passes through each card (called daisy chaining); thus, in Figure 26.4 the signal enters via BG1IN and exits via BG1OUT. This scheme allows a card to intercept the bus grant signal, giving cards nearer the system controller a higher priority. A consequence of daisy chaining is that any unused card slots must be fitted with jumpers between the bus grant pins.

Priority Interrupt Bus.
The priority interrupt bus provides seven interrupt request lines (IRQ1 to IRQ7) with IRQ7 having the highest priority. The interrupt requests may be handled by several SBCs, provided that all interrupts of the same level are handled by the same master. The interrupt acknowledge line is daisy chained through the cards in a similar way to the bus grant lines (the pins are called IACKIN and IACKOUT). Again, any unused card slots must be fitted with jumpers between these pins.

Utility Bus.
The utility bus provides the power supply tracks, the system reset, a 16 MHz clock and two system failure lines, SYSFAIL, which allows a card to indicate that it has suffered a failure, and ACFAIL, which is generated by the power supply monitor. Both failure lines are handled by the system controller SBC.

Associated with the VMEbus is VXI (*V*MEbus e*x*tensions for *i*nstrumentation). This blends the IEEE 1014 VMEbus standard with the IEEE-488 instrumentation bus using the uncommitted pins on the P2 connector to form the IEEE P1155 standard.

Multibus II (IEEE 1296)

Multibus II [3] is a synchronous bus system which is implemented on a single 96-track backplane. Cards conform to the Eurocard format of 220 × 233.65 mm; however, the bus requires only a single 96-pin DIN 41612-type connector. A second connector may be used to support a local bus or merely provide mechanical stability for a double-height card.

Unlike VME, the Multibus II standard has the following features:

1. A synchronous bus; i.e., data transfer is linked to the bus clock rather than the strobe/acknowledge control lines of the asynchronous VME system;
2. A 32-bit multiplexed address and data bus;
3. A message-passing strategy as an alternative to interrupt request lines;
4. Distributed bus arbitration rather than a central system controller (each card contains a message-passing controller, MPC, chip which participates in implementing the message-passing algorithm);
5. Only intelligent cards (i.e., fitted with microprocessors) that can interface to the main bus; simple input/output (I/O) cards should make use of a local bus (e.g., the iLBX bus) using the second DIN connector.

Multibus Signals

Multibus II may be described as having five distinct groups of signals.

Central Control.
The central services module (CSM) in slot 0 is responsible for the generation of this group of signals. The CSM may be implemented on a dedicated card or incorporated onto a SBC. The CSM produces the 10 MHz bus clock as well as a range of reset signals to support both a cold and warm start as well as recovery from power supply failure.

Address/Data.
Multibus II supports a 32-bit multiplexed address/data bus (AD0 to AD31) with four parity lines (PAR0 to PAR3) providing parity information for each data byte. As in all synchronous systems, data are only sampled on an edge of the system clock, a feature that enhances noise immunity.

System Control.
There are ten system control lines (SC0 to SC9) which provide basic handshake information to assist data transfer, e.g., requester is ready or replier is not ready. In addition, these lines are used to convey information such as data width, data error indication, and whether the bus is in the request or reply phase.

Exception Signals.
Two exception signals, bus error (BUSERR) and time-out (TIMOUT), are provided. Any card detecting a data integrity problem must report this to all other cards using BUSERR. The CSM generates a TIMOUT when it detects a data communications hang up on the bus.

Arbitration Group.
The arbitration signals, which determine which card gains control of the bus, consist of a single bus request line (BREQ) and six bus arbitration lines (ARB0 to ARB5). To request exclusive use of the bus, a card asserts the BREQ line and provides it arbitration ID on ARB0 to ARB4. It also uses ARB5 to indicate whether the request has high priority or whether "fairness mode" is acceptable. In this latter mode the card will not make another request until after all other requesters have used the bus. Each card has the same bus arbitration logic within its MPC chip. If several cards make high-priority requests, the order of access is determined by the numerical value of the arbitration ID.

Message Passing on Multibus II

Multibus II uses message passing to implement block data transfers and interrupt requests. Each MPC chip contains a first-in first-out (FIFO) buffer, which ensures that the full bandwidth of the bus can be utilized by storing data immediately before or after transfer. In "solicited" transfers, the MPCs cooperate by warning each other that a message is to be sent. These messages may be sent as 32-byte packets in a manner that is analogous to the block transfer mechanism in VME. "Unsolicited" packets not expected by the receiving MPC are used to set up a block transfer or to act as the equivalent of interrupt request signals.

System Configuration

Multibus II employs a software configuration approach in which information such as the base memory address and arbitration ID are sent down the bus rather than by the use of jumpers or dip switches. Some information such as card type and serial number are coded on the cards themselves.

Other System Buses for Small-Scale Systems

The G64/G96 and STE standards [3] are examples of buses well suited to small industrial real-time instrumentation and control systems because of their relatively simple and hence lower-cost bus interfaces, compared with VME and Multibus. Both buses support DMA, have multiprocessor bus arbitration, and use single-height Eurocard (100×160 mm) cards. Prototyping cards fitted with bus interfaces are readily available and may be used to develop custom designed I/O cards. Power is supplied on the buses at +5 and $\pm$12 V. In addition, the real-time multitasking operating system OS-9 has been ported onto SBCs which operate with these buses. Development systems that support the use of MS-DOS are also available, thus providing access to a wide range of PC-based software especially for graphics applications.

The G64 bus, which was defined by the Swiss company Gespac in 1979, specifies 64 bus lines which are mapped to rows of a DIN41612 connector. The bus has a 16-bit nonmultiplexed data bus and a 16-bit address bus; 32-bit transfers may be achieved by multiplexing the upper 16 bits of the data bus with the address bus. The G96 standard adds a further 32 bus lines by making use of the third row of a DIN41612 connector to extend the address bus width to 24 bits and provide additional interrupt request lines. The STE bus provides an unmultiplexed 8-bit data and 20-bit address bus using 64-pin DIN41612 connectors.

26.4 Personal Computer Buses

There are three main bus standards for personal computers — industry standard architecture (ISA), extended ISA (EISA), and the microchannel architecture (MCA). In addition, the Personal Computer Memory Card International Association (PCMCIA) architecture has been developed primarily for use in laptop and notebook computers.

ISA and EISA are pin compatible and are both synchronous buses with a clock rate of 8 MHz regardless of the clock rate of the main processor, whereas the MCA bus is an asynchronous bus. Slow slave add-on cards can utilize the ISA and EISA buses by using an appropriate number of wait states. The MCA architecture will not be covered in this chapter. Further details of these bus architectures are given in References 4 through 6.

ISA Bus

The original IBM PC and its clones used the standard PC bus which supported 8 data bus and 20 address bus lines and employed a 62-pin printed circuit card edge connector. When IBM introduced the PC-AT, a second 36-pin connector was added to the motherboard backplane slots to provide a 16-bit data bus and increase the number of address lines to 24. This new bus subsequently became known as the ISA bus and is capable of supporting both cards designed for the original PC bus, and cards with two connectors providing full 16-bit data access. The bus master line allows a card to take over control of

the bus; however, this is generally only suited to long-term takeovers. The ISA bus supports 8-bit and 16-bit DMA transfers allowing efficient transfer of blocks of data, e.g., while loading software from disk into RAM. The ISA bus supports the I/O addressing mode of the Intel 80 × 86 range of processors with an I/O address space of 768 locations in addition to the 256 locations reserved for the motherboard.

EISA Bus

The EISA bus provides full 32-bit data and 32-bit address bus lines. ISA cards can fit into EISA bus connectors; however, cards designed for the EISA standard have bilevel edge connectors, providing double the number of contacts as an ISA card. The presence of a notch in the EISA cards allows them to be inserted farther into the motherboard socket than the ISA cards and thus mate with the additional contacts. The EISA standard increases the maximum ISA data width in DMA transfer to 32 bits and also provides a much-enhanced bus master. The following features of EISA are worthy of note.

Bus Arbitration

All EISA systems have a central arbitration control (CAC) device on the motherboard. The CAC uses a multilevel rotating priority arbitration scheme. The top-priority level rotates around three customers, the DMA controller, the dynamic memory refresh controller, and, alternately, either the main CPU or one of the bus master cards. A device that does not make a request is skipped over in the rotation process. The bus masters take it in turns to gain access to the top-priority level. Whereas ISA supported a single bus request line, the EISA standard provides a dedicated pair of request lines (MREQ0 to 14) and acknowledge lines (MAK0 to 14) for each bus master. (*Note:* Although this allows 15 bus masters to be used, in many systems the chip set which implements the CAC supports bus masters in a limited number of the EISA sockets.) The CAC supports preemption, i.e., it allows a device making a request to capture control from another device if it is next in turn. A bus master card must release the bus within 64 bus clock cycles, whereas a DMA controller has 32 clock cycles to surrender the bus.

Input/Output

The EISA bus provides 768 additional I/O locations for each slot in addition to the 768 ISA I/O locations that may be accessed from any slot. EISA cards contain nonvolatile memory to store configuration information (see below), and a minimum of 340 bytes of the I/O address space of each slot is reserved for this purpose.

System Configuration

The EISA system provides support for automatic system configuration to replace the use of jumpers and dip switches to specify parameters such as the base memory address, interrupt request line number, or DMA channel number used by each add-on card. Information on the product type and manufacturer are stored on each EISA add-on card and is read by the CPU during system start-up, making it possible to identify the slots that are fitted with full EISA cards. Manufacturers of both ISA and EISA cards should supply a configuration file containing information on the programmable options that are available. When a system configuration program is run, an optimum configuration of the boards will be determined, and the configuration information written into the I/O space of each EISA card. For ISA cards, the user can be informed of the required jumper settings.

PCMCIA

The PCMCIA architecture was developed by the Personal Computer Memory Card International Association and the Japan Electronics Industry Development Association for removable add-on cards for laptop and notebook computers. Each PCMCIA socket has its own host bus adapter (HBA), which acts as an interface to the main computer bus. Cards may be plugged into PCMCIA sockets either before or after the computer has been powered up. There are three types of PC cards all with the same planar

dimensions (54.00 × 85.6 mm) but with differing thicknesses, namely, 3.3 mm for type I, 5.0 mm for type II, and 10.5 mm for type III. Cards and sockets are keyed to prevent them from being inserted the wrong way around. The PCMCIA standard supports cards that operate at several possible voltage levels, i.e., 5.0 V cards, 3.3 V cards, or dual-voltage 5.0 V/3.3 V cards.

Configuration

PC cards contain configuration information called the card information structure (CIS) which is stored in nonvolatile memory. Cards may be configured either on system power-up or on insertion of the card into the socket (i.e., plug and play). Configuration is carried out using a form of device driver called an enabler. The enabler may make use of two additional software services called card services and socket services. Socket services, which may be contained in ROM on the PC or loaded from disk during power-up, provide function calls to allow the HBA to be configured to cooperate with the PC card. Card services, which may be an extension to the operating system or an installable device driver, act as a server for the enabler, which performs as a client. Card services provide a range of functions such as accessing the CIS of the card, requesting system resources required by the card, and telling the enabler that a card has been inserted or removed from the socket. Enablers may be classified as dedicated to one particular card or generic, i.e., designed for a range of cards. Note that early PCMCIA cards were not designed for use with card services.

The PCMCIA Socket Interface

The PCMCIA socket comprises a 68-pin connector with 26 address lines providing access to 64 MB of memory space and a 16-bit data bus. Two V_{cc} pins and four ground pins are supplied. The maximum current that can be supplied to the card is 1.0 A with a maximum of 0.5 A from each of the two power supply pins. Release 2.x sockets apply 5.0 V to the V_{cc} pins on power-up and reduce this to 3.3 V if the card has dual-voltage capability. Cards that operate at 3.3 V only are keyed so they cannot fit into this type of socket and only into low-voltage sockets. The supply voltage provided by low-voltage sockets depends on the logic state of its voltage sense inputs. A PCMCIA socket can be configured either as a *memory only socket* or as a *memory or I/O socket.* Initially, the socket acts as a memory only socket but is converted by the enabler to a memory or I/O socket if the card is required to support I/O functions. In this mode the card can generate an interrupt request via a single IRQ pin and support both 8-bit and 16-bit I/O data transfers. DMA may be supported but not by Release 2.x systems.

PC/104

The enormous popularity of PC architecture resulted in its use in embedded systems. A need then arose for a more compact implementation of the ISA bus that could accommodate the reduced space and power requirements of embedded applications. The PC/104 specification (1992) was adopted as the base for an IEEE draft standard called the P996.1 Standard for Compact Embedded PC Modules. The key features of the PC/104 are

- Size reduced to 90 × 96 mm (3.550 × 3.775 in.);
- Self-stacking bus allowing modules to be "piggy-backed" and eliminating backplanes or cardframes;
- Rugged 64-contact and 40-contact male and female headers replacing the PC edge connectors (64 + 40 = 104, hence PC/**104**);
- Lower power consumption (<2 W per module).

PC/104 CPU modules range from a basic 9.6 MHz, 8088 compatible XT with one serial port and a keyboard connector to a 100 MHz, 80486DX4 with four serial ports, parallel port, IDE disk controller, display controller, Ethernet adapter, keyboard port, and up to 64 MB of on-board RAM. Pentium-based systems are also available. Systems can be customized from a wide range of modules, including data acquisition boards, solid-state disk modules, and LAN support, all in the same 3.6 by 3.8 in. stackable format.

TABLE 26.3 Manufacturers/Suppliers of Bus-Based Systems and Industrial PCs

System	Manufacturer/Supplier	System	Manufacturer/Supplier
VME	Wordsworth Technology Ltd. 6 Enterprise Way Edenbridge, Kent TN8 6HF U.K. Tel: +44 (0) 1732 866988	STE	Arcom Control Systems Units 8-10 Clifton Road Cambridge CB1 4WH, U.K. Tel +44 (0) 1223 411200
	PEP Modular Computers, Inc. 750 Holiday Drive, Building 9 Pittsburg, PA 15220 Tel: (412) 921-3322	G64	Gespac SA 18 Chemin des Aulx 1228 Geneva, Switzerland Tel: +41 (22) 794 34 00
	BVM Ltd, Hobb Lane Hedge End Southampton, SO30 0GH, U.K. Tel: +44 (0) 1489 780144		Altek Microcomponents Ltd. Lifetrend House Heyes Lane Alderley Edge, Cheshire SK9 7LW, U.K. Tel: +44 (0) 1625 584804
	Motorola, Inc. Computer Group 2900 S Diablo Way Tempe, AZ 85282 Tel: (800) 759-1107	Industrial PC	Blue Chip Technology Ltd. Chowley Oak, Tattenhall Chester, Cheshire CH3 9EX, U.K. Tel: +44 (0) 1829 772 000
VXI	National Instruments 6504 Bridge Point Parkway Austin, TX 78730-5039 Tel: (512) 794-0100		Capax Industrial PC Systems Ltd. Airport House, Purley Way Croydon, Surrey, CR0 0XZ, U.K. Tel +44 (0) 181 667 9000
Multibus	Tadpole Technology, Inc. 2001 Gateway Place, Suite 550 West, San Jose, CA 95110 Tel: (408)441-7920	PC/104	ComputerBoards, Inc. 125 High Street Mansfield, MA 02048 Tel: (508) 261-1123
	Intel Corp. 3065 Bowers Avenue Santa Clara, CA 95051		Diamond Point International (Europe) Ltd. Unit 9, North Point Business Estate, Enterprise Close Rochester, Kent ME2 4LY, U.K. Tel +44 (0)1634 718100
	Syntel Microsystems Queens Mill Road Huddersfield, HD1 3PG, U.K. Tel: +44 (0) 1484 535101		

The wide availability of software development tools for the PC, the large number of software developers familiar with the PC environment, and the ease of transferring software developed on a conventional PC to the PC/104 make this an increasingly popular format. Table 26.3 lists some manufacturers and suppliers of bus-based systems and industrial computers.

26.5 Peripherals

Computer peripherals fall conveniently into two categories. The first category may be considered to be internal to the computer system and comprises cards plugged directly into a computer bus slot. The second category comprises instruments external to the computer but controlled by it. These external instruments are usually themselves "intelligent," being controlled by their own CPU, and are linked to the main computer by a serial (RS-232) line or the IEEE-488 bus (GPIB). Such instruments may normally be operated in a stand-alone mode in response to their front panel controls without the requirement of an external computer.

TABLE 26.4 Typical Internal Peripherals

Card Type	Facilities Offered
Display adapter	Provides video and graphics facilities
Serial communications adapter	Serial communication to a similar device using the RS232, RS422, or RS485 standard
IEEE-488 Adapter	Communication with intelligent instruments using the IEEE-488 bus (GPIB)
Digital input/output (I/O)	Individual I/O lines, normally grouped as an 8-bit byte, which may be used to provide logic high and low signals (output) or sense logic high or low signals (input); inputs and outputs may be optically isolated; outputs may drive relays
Counter/timer	Hardware counter timer allowing external pulses to be counted, digital waveforms generated or pulse widths measured; counter/timers are often available as an additional facility on digital I/O cards
Analog input	Converts an analog input voltage to an integer value which may be read by the computer; resolution typically 8, 10, 12, or 16 bits; input voltage range may be fixed or may be user selectable; conversion times vary from several seconds to tens of nanoseconds
Analog output	Produces an analog output voltage proportional to a digital input; resolution typically be 8, 10, 12, or 16 bits

Internal Cards

Internal cards are available to perform a wide range of functions. Table 26.4 provides a brief list of representative types. Note that both the serial interface and the IEEE-488 adapter required for the control of external "intelligent" peripherals will be fitted as internal cards. Almost all internal peripherals need to be configured before use to set up such parameters as the base address, the interrupts, and/or DMA channels used. This may involve setting jumpers or switches on the card or may be accomplished under software control using a configuration file supplied by the manufacturer. To operate the cards, bytes are written or read from appropriate addresses on the cards by the controlling SBC. The mechanism for doing this is discussed further in the section concerned with software for data acquisition. Further detail on some of the card types is given below.

Display Adapter

While graphics support is normally available as standard on a PC-based system, this is not the case with other bus-based systems such as VME or Multibus. These systems are provided with a serial port that may be connected to a terminal to provide a text-only dialogue with the operating system running on the SBC. In such circumstances the choice of display adapter will determined by the user requirements, taking into account the support for the device provided by any software packages that are to be used. If the user intends to write custom graphics software, it is essential to ensure that a graphics library is available from the vendor providing as a minimum line drawing, arc drawing, and block color fill facilities.

IEEE-488 Adapter

This device provides support for communications across the IEEE-488 bus or GPIB (general purpose-interface bus). The bus itself comprises eight data lines, five interface management lines and three handshake lines. Transfers are parallel, synchronous, and at rates up to 1.5 MB/s. The IEEE-488 bus and its applications are discussed further in the section on external instruments.

Serial Communications Adapter

This device provides support for serial communications using the RS-232, RS-422, or RS-485 standards. It is used to provide a text-based terminal for systems based on Multibus or VME and to communicate with "intelligent" instruments such as position controllers, multimeters, or storage oscilloscopes fitted with similar interfaces.

Serial adapters convert parallel data to and from a bit stream in which each data byte (5, 7, or 8 bits) is framed by a start bit, an optional parity bit, and one or more stop bits. The bit stream may be sent via a circuit consisting of only two wires at rates of up to 115,200 bits per second (commonly referred to as 115200 baud). Common bit rates are 300, 600, 1200, 2400, 4800, 9600, 19200, 28800, 38400, 57600,

115200 baud. Both the transmitting and receiving adapters must be configured, normally by software, for the same baud rate, number of data bits, stop bits, and parity. Some form of flow control to prevent a receiver from being overloaded with incoming data is essential. This is accomplished either by separate handshake lines (e.g., those denoted by RTS and CTS in the standard) or by software where the receiver sends a special control byte (XOFF) back to the transmitter telling it to stop sending until it receives a second control byte (XON) to reenable it. For historical reasons the RS-232 standard does not define a bidirectional handshake procedure, and manufacturers have been forced to implement their own schemes which are not always compatible with each other.

Serial communication between devices may be

Full duplex, where either device may transmit or receive data at any time;
Half duplex, where both devices are capable of transmission or reception but only one may transmit at any instant;
Simplex, where one device is a transmitter, one is a receiver, and data can only flow in a single direction.

RS-232, developed by the Electronics Industries Association (EIA), is the oldest standard, originally developed in the early 1960s, to allow mainframe computers to communicate with terminals via modems and telephone lines. This is the origin of the names of some of the connections (e.g., RI ring indicator, DCD data carrier detect), which have no relevance in the applications considered here. A related problem is that the standard expects that the devices being connected are data terminal equipment (DTE), at one end of the link, and data communication equipment (DCE) at the other. Computers and terminals are DTE, while modems are DCE. The most commonly used revision of the standard, RS-232-C (revisions D and E also exist), was made in 1969 and is still widely used. Serial communication is made using voltage levels in the region of ±12 V, over distances up to 15 m (50 ft) at speeds up to 20000 baud.

RS-422, also developed by the EIA, is an enhancement of the RS-232 standard. Differential transmitters and receivers are employed which allow one transmitter to drive up to ten receivers, using a twisted-pair connection for each circuit, at bit rates up to 10 MBaud at distances up to 12 m (40 ft) or 100 kBaud at distances up to 1200 m (4000 ft). The RTS and CTS lines (defined in the standard) are used for flow control, while the RXD and TXD lines are used to transmit and receive data. Thus, a two-twisted-pair cable is required for duplex connection without hardware handshaking. A four-twisted-pair cable is required if hardware handshaking is used.

RS-485 is based on RS-422 and allows up to 32 driver/receiver pairs to be connected to a common data bus (two twisted pairs). Clearly, only one device can be allowed to transmit at any one time. The RTS circuit is used to disable the other transmitters connected to the bus if a device is required to transmit data. Handshaking is performed using software.

The serial interfaces on instruments are usually configured as DTE devices. We are faced with the problem of connecting one DTE device (the computer serial interface) to another (the instrument), which is not what the RS-232 standard was designed for. Furthermore, since the standard does not define a bidirectional handshake to control data flow, several incompatible handshaking schemes exist. A comprehensive survey of these is presented in Reference 7. A common solution to the DTE to DTE connection problem is the so-called *null modem,* which is nothing more than a specially wired cable. Figure 26.5 shows two such connection schemes. One requires the software handshaking procedure and the other implements a bidirectional hardware handshake. The reader should refer to Reference 7 for details of other schemes and for the definitions of the mnemonics used to label the connections.

Some common problems encountered in practice are

1. The received data are garbage. This is almost always due to the baud rate, parity, and number of stop bits not being the same at both ends of the link.
2. Data initially correct, but parts in the middle are missing. This is probably a handshake problem. The transmitting device is sending data faster than the receiver can process it.
3. No communications at all. Probably a handshake problem where the transmitter does not sense that the receiver is ready.

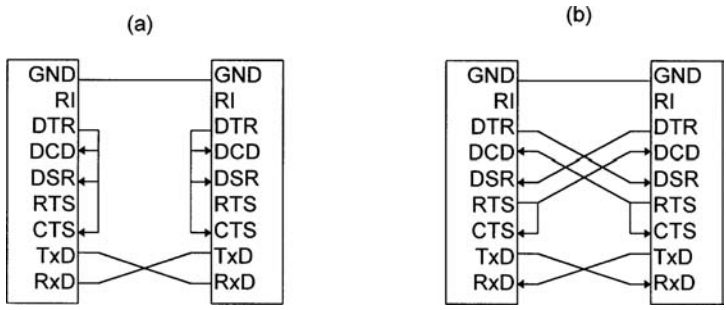

FIGURE 26.5 Two null modems for connecting DTE to DTE. In (a) all handshaking must be in software. The DTR line "fools" the serial interface that it is connected to the handshake lines of another device. Scheme (b) implements a hardware handshake. The DTR–DSR connection shows each device that the other is present. The RTS lines, connected to the DCD of the other device, which it can monitor, are used to control the flow of data in either direction.

Digital Input and Output

These cards provide I/O lines, normally in groups of eight, which may be used to sense or generate digital signals for devices outside the computer. A group of eight input lines is referred to as an (8-bit) input port and a group of eight output lines as an (8-bit) output port. Input and output levels vary from card to card and it is best to consult the appropriate data sheet. Typically, voltages between 2.5 and 5.0 V are considered as high logic levels, whereas voltages between 0.0 and 0.5 are considered as low logic levels. These levels are sometimes referred to loosely as TTL (transistor transistor logic) levels. Note that the actual logic levels used by the various TTL families differ from these slightly. The "high" and "low" ranges may be slightly different for input and output lines. Output lines often have limited current sourcing and sinking abilities compared with TTL, and it is therefore often necessary to buffer them. It is important that voltages exceeding the maximum rated values do not appear on inputs or outputs (e.g., attempting to switch an inductive load might produce a dangerously high transient voltage at an output); otherwise, the device may be damaged. Where this is likely to be a problem, inputs and outputs should be suitably buffered or even optically isolated, which provides protection up to a few kilovolts. Outputs may also drive appropriately connected relays. I/O cards with these facilities on board are readily available.

As a minimum, an I/O card may be expected to support a control register, two I/O ports each with an associated data register, and some handshake lines. Handshake lines may sometimes be used to generate interrupts on the controlling SBC. A byte written to the control register is used to configure the I/O ports, i.e., to determine if they are to behave as input or output ports as well as to select the function, if any, of the handshake lines. It may not be possible to select the direction of optically isolated or buffered ports. Writing a byte to an output port causes a pattern of high and low voltages to appear on the lines reflecting the pattern of zeros and ones in the binary representation of the byte written. Similarly, when a byte is read from an input port, the number read is specified in binary representation by the pattern of high and low logic levels on the input lines. The following example illustrates this.

The Intel 8255 Programmable Peripheral Interface (PPI) is commonly used in digital I/O cards for the PC. Data for this device are readily available [8]. The 8255 provides three ports, denoted A, B, and C, and a control register. Ports A and B may be designated as an 8-bit input port or an 8-bit output port. Port C may be considered as two independent 4-bit ports, which may be chosen independently as input or output. Port C may also provide handshake functions. There are three modes in which the chip may operate. The simplest, mode 0, which provides basic input and output without automatic handshaking, is used in our example. Note that the 8 bits of an I/O port are conventionally labeled bits 0 to 7. This is because bit 0 is weighted 2^0, bit 1 weighted 2^1, etc. in the binary representation of the number read from or written to the port.

To configure the PPI to operate in mode 0 with port A as an input port and port B as an output port the bit pattern 10011001 must be written to the control register. This number is equivalent to 99 in

hexadecimal or 153 in decimal. Suppose now that switches connected to port A hold bits 2 and 7 high and the remaining bits low. The resulting binary pattern will be 10000100, which is equivalent to hexadecimal 84 or decimal 132. When port A is read, the number 132 (decimal) will therefore be obtained. To hold the lines connected to bits 2, 3, and 5 of port B high while leaving the remaining lines low, we see that the binary pattern 00101100 must appear at port B. 00101100 binary is equivalent to 2C hexadecimal or 44 decimal. We therefore write the decimal number 44 to port B.

Counter/Timers

These devices typically provide software-programmable event counting, pulse, and frequency measurement. As output devices, they may generate a single pulse (one-shot) when a programmable number of input pulses have been counted and produce square waves of arbitrary frequency and complex duty cycles. Frequencies generated are normally based on an on-board crystal clock to provide independence from the internal clock speed of the computer. Counter/timer cards commonly support at least three independent 16-bit counters.

Common applications include:

1. Alarms. The counter is in one-shot mode and generates a single pulse on timeout. This is connected to interrupt the computer and alert the user in the middle of the currently executing task.
2. Watchdog timer. This is used to detect problems, particularly in systems which are intended to operate without operator intervention. It is similar to the Alarm described earlier except that the interrupt is used to reset the computer. In normal operation this will never occur as all software tasks executing are designed to update the counter constantly so that it never reaches its terminal count. Only if a problem develops, e.g., a software "crash," will the counter time out and the system be reset.
3. The generation of complex waveforms, e.g., for pulse-width modulation. This application uses two counters in cascade, one (T1) to provide regular pulses at the carrier frequency triggering another (T2), in one-shot mode, to provide the variable duty cycle as shown in Figure 26.6.

Analog Input

An analog input card uses an analog-to-digital converter (ADC) that accepts an input voltage and supplies an integer proportional to that voltage to the computer. Many cards now are produced with on-board signal conditioning circuits that provide for variable gain either by means of switches or under program control. Cards with specialized signal conditioning circuits for common applications such as thermocouple linearization or interfacing to strain gages and other bridge sensors, are available. Signal conditioning to protect cards destined to be used in hostile electrical environments is also available. Cheaper cards may provide fixed gain and require additional signal conditioning circuits to be provided external to the computer. Many cards also feature multiplexed inputs where one of several inputs may be selected under program control to be fed to the ADC. Some important parameters to consider in selecting a analog input board are given in Table 26.5.

At rates above a few tens of kilohertz, interrupt-driven data capture is essential to maintain speed. Faster data rates require on-board memory to avoid degrading the performance of the controlling SBC. In this case DMA may be used to transfer data to main memory and increase performance further.

A timer function is often incorporated to allow samples to be taken at regular intervals independently of what the controlling SBC is doing. An interrupt is generated when the conversion is complete, and an interrupt service routine is then activated to read the result of the conversion into memory. The writing and installation of interrupt service routines is not a trivial task and is best left to those with an intimate knowledge of the operating system running on the SBC. Fortunately, most manufacturers supply software (device drivers) for this purpose. The simpler analog input boards may be driven by writing values directly to registers on them in a similar manner to the example given for digital I/O cards. Manufacturers now commonly provide a software library which may be called from a variety of high-level languages to allow the user to access the card in a more intuitive way. This is discussed further in the section on software.

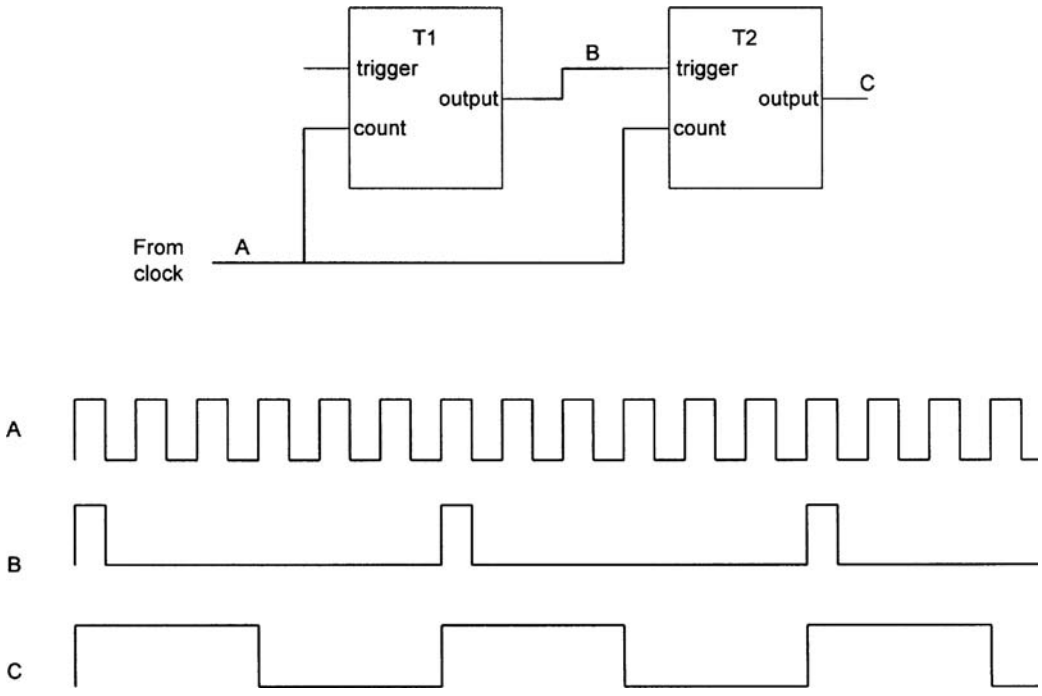

FIGURE 26.6 Using two timers to produce pulse-width modulation under software control. Both timers receive input pulses at a constant frequency from an external clock, as shown at A. Timer T1 operates in continuous mode. The trigger has no effect in this mode. The output of T1 is a single positive-going pulse when it has counted the specified (by software) number of input pulses, as shown at B. Timer T2 operates in one-shot mode. Each time it receives a trigger pulse, its output goes high for a specified (again by software) number of counts, as shown at C. In this way the frequency of the output at C is controlled by the count specified for T1 and the width of the positive-going part of C is controlled by the count specified for T2.

TABLE 26.5 Common ADC Parameters

Parameter	Description
Resolution	The smallest change in input detectable in the digital output; resolutions are commonly expressed in the number of significant bits in the digital output; hence, 8-bit resolution means 1 part in 256; 10-bit, 1 part in 1024, and 12-bit, 1 part in 4096
Linearity	The extent to which the output deviates from a linear relationship with the input; good devices will be linear to ±1 least significant bit; i.e., the output value is guaranteed to be within ±1 of an exactly linear conversion
Range	The maximum (and minimum) input voltages; inputs may be unipolar, e.g., 0–5 V or bipolar ±5 V; voltages are specified relative to ground unless the inputs are differential, e.g., those designed for bridge sensors
Conversion speed	The time taken to convert an input voltage into a digital output, typically 1 s to 1 μs; may also be quoted as a sample rate
Linearity	The extent to which the conversion is linear, e.g., a linearity of ±1 least-significant bit means that the output value is within ±1 of the ideal linear conversion
Input impedance	The impedance between the input terminal and ground or between differential inputs

Analog Output

Analog output cards are available as 8-, 10-, or 12-bit devices. Frequently, a card will support several channels of analog output with provision for delaying the updating of channels so that all can be updated simultaneously. Output voltages may be unipolar or bipolar and current outputs (4 to 20 mA) are also

available. Signal conditioning (buffering) is necessary to drive loads drawing currents of more than a few milliamps. Special care should be taken with inductive loads, e.g., motors to avoid damage to the device by transient voltages. Specially designed position-control modules incorporating suitably buffered analog and digital I/O are available for this purpose.

External Peripherals

These are usually "intelligent" devices which can operate via their front panel controls without another computer but are additionally capable of being controlled by a computer. Many common laboratory instruments are available with such facilities, including power supplies, signal generators, storage oscilloscopes, voltmeters, spectrophotometers, and position controllers. A computer can coordinate the actions of several such instruments to gather then manipulate and display data in a way which enhances the power of the instrumentation system. An almost trivial example is the use of a computer to control a signal generator and a voltmeter in order to generate the frequency response of an amplifier automatically. Such a system has an obvious role in automatic testing rigs.

Two common methods are used to control such devices: a serial link or the IEEE-488 bus. In both cases the devices are controlled by sending messages consisting of sequences of ASCII characters. Usually the sequences are chosen to have an obvious meaning, as in the example that follows, but this is not always the case — particularly with older devices where user friendliness was often sacrificed as a result of limited memory and processing power! The message sequence

"FREQ10kHz"
"SINE"
"1.0VOLTRMS"

might be used to set a signal generator to produce a 10 kHz sinusoidal signal at 1 V rms. There is little standardization in the form of device messages used although the IEEE standard 488.2 goes some way in this direction. Responses from instruments are sent in the same way, i.e., as ASCII characters so that a voltmeter might respond to a command to make a measurement with the data

"AC2.01mV"

to indicate that it was on an ac voltage range and measured 2.1 mV. Again, there is little standardization in the format of responses. Large blocks of data may be send in a binary format where possible (8-bit serial links or IEEE-488 bus) to minimize the amount of data to be transferred.

Serial Devices

Serial control of devices is accomplished using links conforming to one of the serial standards (RS-232, RS-422, or RS-485) described elsewhere in this chapter. This is a relatively simple method of control and has the advantage that much of the preliminary testing and debugging of a system can be done using a terminal or a terminal emulator program such as the public domain KERMIT available from Kermit Distribution (Columbia University Academic Information Systems, 612 West 115th Street, New York, NY 10025, Tel: 212-854-3703). The writing of custom software that accesses the serial interface of the controlling computer is relatively easy under common operating systems including DOS, Windows, Windows 95, UNIX, and OS-9 using languages such as C, Pascal, or BASIC. It is increasingly common, particularly for DOS and Windows applications, for manufacturers to provide software support for their devices.

Disadvantages of serial transfers are the relative slowness when large amounts of data are transferred, the lack of standardization in device messages, and the limited control facilities available. Advantages are the ease of testing, the simplicity of the controlling software, the relative simplicity of the interconnection scheme, and — for remote instrumentation systems — the fact that with the use of modems data can be transferred over large distances using standard telephone lines or even a radio link.

IEEE 488 Devices

The IEEE standard 488 was developed in the 1970s and rapidly became an industry standard for the interconnection and control of test equipment. This standard was modified slightly in 1987 (IEEE standard 488.1) to allow for the considerable enhancements of IEEE standard 488.2 which was introduced at the same time [9,10]. The original IEEE standard 488 specifies the electrical characteristics of the bus, the mechanical characteristics of its connectors, and a set of messages to be passed between interfaces. It does not attempt to provide any syntax or structure for communicating these messages, to specify commonly used commands, or to establish a standard for device-specific messages. These issues are addressed in IEEE standard 488.2.

The bus itself supports synchronous parallel transfers of data using three groups of lines,

A bidirectional 8-bit data bus,
Five interface management lines, and
Three handshake lines,

over distances of up to 20 m and at data transfer rates of up to 1 MB/s.

Devices on the bus are classed as *talkers, listeners,* or *controllers.* In general, the computer system is the bus controller which can also talk (send data) or listen (receive data). Most devices are both talkers and listeners: for example, a digital voltmeter will be a listener when receiving instructions to set the voltage range prior to making a measurement but will be a talker when returning the result of the measurement to the controller, which is itself acting as a listener. Each device on the bus must be assigned a unique address which is a number between 0 and 30. This may be done from the front panel of the device or, less conveniently, by setting switches elsewhere on the device.

It is a difficult, time-consuming, and error-prone process to write software to drive an IEEE-488 card. Purchasers of new IEEE-488 interfaces are strongly advised to obtain a device driver from the manufacturer. Such device drivers are now readily available and integrate the card into the filing system of the operating system running on the controlling computer. This allows the interface to be accessed in a natural way from high-level languages running on the controller.

26.6 Software for Instrumentation Systems

The difficulty involved in writing software for instrumentation systems depends largely on the support available from the manufacturers of the subsystems, on the operating system (if any), and on the development tools available. On one extreme, one may be working in a virtual instrument environment, such as that provided by the National Instrument LabVIEW, where software development is entirely graphical and, for small projects at least, is readily undertaken by users with little or no prior experience. On the other extreme, one is faced with the problem of developing software, which is at the very least interrupt driven and probably multitasking, for a target SBC with no resident operating system; this requires considerable expertise in software design and development together with the availability of development tools such as cross compilers and source-level debuggers.

Virtual Instruments

Figure 26.7 shows a layer model of the software for a generalized instrumentation system. The application layer handles the data acquisition, analysis, and presentation. The instrument drivers provide a mechanism

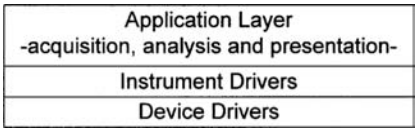

FIGURE 26.7 A layer model for instrumentation systems.

for communicating with the instruments in a standard way without requiring the user to know about the often cryptic data strings which need to be sent. For example, all digital multimeters will need the facility to chose a specific input voltage range. The range coding, resolution, etc. that have to be sent to the multimeter to achieve this will vary from instrument to instrument; however, the instrument driver allows the software writer programming in the application layer to call a procedure such as

SetVoltageRange(VoltageValue)

and this procedure call is the same for all multimeters. Although some manufacturers use the term slightly differently, the instrument driver is in effect the virtual instrument. Writing *instrument drivers* is a time-consuming but not too difficult task. Instrument drivers for proprietary instrumentation software design packages are readily available from instrument manufacturers. Device drivers integrate the controlling interface (e.g., IEEE-488, RS-232, or internal card) into the operating system of the computer. Writing a *device driver* requires a detailed knowledge of the device hardware and of the computer operating system. This is a difficult task and new interfaces should be purchased with a device driver appropriate for the operating system wherever possible.

A number of development environments which are based on the virtual instrument concept are now available. These free the user from the problems of writing conventional software to control instruments and handle the data produced. Instruments appear to the software developer as "front panels" drawn on the computer screen, complete with familiar buttons, knobs, and displays. Data flow is handled by linking instruments in a block diagram using a mouse in an environment that resembles an ordinary drawing package. The software developer is working only in the application layer.

While the graphical environment allows simple systems to be developed rapidly, experienced programmers may find it restrictive. There are software development systems available that give the programmer access libraries containing instrument drivers, data analysis routines, graphics functions, and data visualizations facilities in commonly used high-level languages such as C, Pascal, BASIC, and FORTRAN. Table 26.2 lists representative software packages.

Working Directly with Peripherals

It may occasionally be the case that the cost of software support for a virtual instrument development environment is not justified for a small application. Software must then be written to interface directly with the peripheral. The earlier section on digital input and output explained in principal what was necessary to program a simple interface chip. We now continue this example and show using the language C how this might be achieved.

The method of accessing the registers of peripheral cards depends on the microprocessor involved and may not even be a standard feature of the language being used. Where the peripheral forms part of the same address space as the computer memory, such as in the Motorola 680XX series, pointers can be used to read and write values in the registers. The Intel 80 × 86 series of processors often place peripherals in a separate address space which may not be accessed by pointers. In this case an extension to the language is required. Borland's Turbo C and C++ provide functions to read and write I/O mapped devices:

unsigned char inportb(int portid)
void outportb(int portid, unsigned char value)

These are used in the code fragment which implements the software for our earlier example. We assume that the 8255 PPI has base address 0 × 1b0 and that the program copies the value read from the input port (port A) directly to the output port (port B), until bit 0 of the input becomes zero

```
/ * define the addresses of the registers for the PPI */
#define BASE 0x1b0
#define PABASE
#define PBBASE+1
#define CONTROLBASE+3
```

```
unsigned char xin;/ * declare an 8-bit variable */
outportb(CONTROL, 153)/* set port A as input, port B as output */
do
{
          xin = inportb(PA);/ * read the input */
          outportb(PB,xin);/ * write the output */
}
while(xin & 1);/* loop if bit 0 is still 1 */
```

This fragment also illustrates that high-level languages generally only support input and output of bytes. Masking techniques must be used to access individual bits.

The Choice of Operating System

The software development support discussed so far is typically available under DOS, Windows 3.1, Windows 95, Windows NT, and UNIX. When a system is multitasking, it has to meet stringent real-time constraints; however, none of these operating systems is particularly appropriate. DOS does not support multitasking, and the others are not optimized for real-time systems which require speedy context switching and rapid interrupt response. The most fundamental requirement of real-time applications is that ability of the system to respond to external events with very short, bounded, and predictable delays. Table 26.6 lists some important real-time operating systems and kernels.

Real-time operating systems tend not to have the mature and powerful software development support available for conventional operating systems. It is not possible simply to develop the software on a familiar operating system and then transfer the working programs to the target system. Much of the debugging, testing, and system integration will have to be done on the target itself to access the hardware. A common solution is a development system in which a conventional workstation is linked to the target system. Software is developed on the workstation using familiar tools, e.g., a Windows-based editor, support for version control, and a powerful filing system. At any time, code can be cross-compiled (i.e., compiled for the processor on the target system) and downloaded to the target. The workstation may then monitor the execution of the software running on the target processor. Features that allow the user to single-step

TABLE 26.6 Some Important Real-Time Operating Systems and Kernels

System	OS/Kernel	Manufacturer
OS-9	OS	Microware Systems Corporation 1900 NW 114th Street, Des Moines, IA 50325
LynxOS	OS	Lynx Real-Time Systems, Inc. 16870 Lark Avenue Los Gatos, CA 95030-2315
VxWorks	OS	Wind River Systems 1010 Atlantic Avenue Alameda, CA 94501
VRTX/OS	OS	Microtec Research 2350 Mission College Blvd. Santa Clara, CA 95054
VRTX	Kernel	Microtec Research 2350 Mission College Blvd. Santa Clara, CA 95054
iRMX	Kernel	Intel Corp. 3065 Bowers Avenue Santa Clara, CA 95051

through the source code, seen in a workstation window, while viewing the status of key variables in another are available. It is also possible to set breakpoints and allow the processes to run until one is encountered. Manufacturers of real-time operating systems are often able to provide development support of this type for their product for a variety of workstations.

References

1. Micrology pbt, Inc., *The VMEbus Specification Manual: Revision C*1, PRINTEX, 1985.
2. S. Heath, *VMEbus: A Practical Companion*, Boston, MA: Butterworth Heinemann, 1993.
3. J. Di Giacomo, Ed., *Digital Bus Handbook*, New York: McGraw-Hill, 1990.
4. T. Shanley and D. Anderson, *ISA System Architecture,* 3rd ed., Reading, MA: Addison-Wesley, 1995.
5. T. Shanley and D. Anderson, *EISA System Architecture,* 2nd ed., Reading, MA: Addison-Wesley, 1995.
6. T. Shanley and D. Anderson, *PCMCIA System Architecture,* 2nd ed., Reading, MA: Addison-Wesley, 1995.
7. M.D. Seyer, *RS*232 *Made Easy*, Englewood Cliffs, NJ: Prentice-Hall, 1984.
8. Intel Corporation, *Intel Microsystems Component Handbook*, Vol. 2.
9. IEEE Std 488.1-1987, *IEEE Standard Digital Interface for Programmable Instrumentation.*
10. IEEE Std 488.2-1987, *IEEE Standard Codes, Formats, Protocols and Common Commands.*