INTERNATIONAL ATOMIC ENERGY AGENCY
UNITED NATIONS EDUCATIONAL, SCIENTIFIC AND CULTURAL ORGANIZATION
INTERNATIONAL CENTRE FOR THEORETICAL PHYSICS
I.C.T.P., P.O. BOX 586, 34100 TRIESTE, ITALY, CABLE: CENTRATOM TRIESTE

H4.SMR. 403/6

FIFTH COLLEGE ON MICROPROCESSORS: TECHNOLOGY AND APPLICATIONS IN PHYSICS

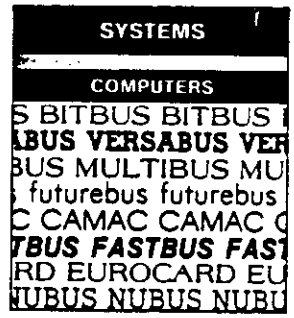
2- 27 October 1989

A framework for computer design

R.W. DOBINSON
CERN, EP Division, Geneva, Switzerland

A framework for computer design

Instead of starting off with a microprocessor, equipment designers must now select from among the many available bus standards before choosing hardware and software



All too often, engineers end up addressing the wrong issues when designing systems. In the selection of a bus, undue emphasis may be placed on largely irrelevant factors.

Here are two statements describing the best bus. Are they true or false?

(1) The best bus for a computer system transfers data at the highest possible rate and accommodates many microprocessors.

(2) The best bus for a microprocessor-based system is always the one designed by the manufacturer of the processor.

If you answered "false" in both cases, you are right. There is no single all-purpose bus that is "best" for every application. For a system implementor, the most important step in selecting a bus is to understand the needs to be met and the problems to be solved by the new product.

Performance considerations in particular are a potential minefield for the unwary. Usually the temptation is to overspecify. A modern high-speed bus with a 32-bit data path, designed to support multiple processors, is complete overkill for many applications. A cheaper, slower, 8-bit bus, optimized for a single processor, may be perfectly adequate.

The distance between the components that must be connected by a bus also affects the choice of a bus. Some instrumentation buses are designed to connect crates of test and measurement equipment hundreds of feet apart; others function only within a rack of equipment.

Computer systems consist of several distinct components: processors, memories, and peripherals. These must share and exchange instructions and data. Doing this requires a means of transferring the information, and a mutually accepted set of rules for transfers. The physical medium for electric signals and the protocols for transfer of information constitute a bus.

Buses are a crucial component of the technology used to interconnect computing equipment over a wide range of distances and speeds. In real-time data acquisition and control systems, for example, they provide the interface between the computer and the instruments measuring physical phenomena. They play a role complementary to longer-distance networks.

Today there are many different bus standards [see table, p. 50]. This proliferation indicates that no one bus is suitable for all applications. Not all of these standards, however, are motivated strictly by technical considerations. The marketing goals of major manufacturers, and the technical needs of special-interest groups, can lead to new bus designs that duplicate the functions of existing standards. The majority of designers will be best served by selecting an open standard independent of any given manufacturer.

For both users and manufacturers, bus standards help solve the problem of ever-increasing variety and complexity in electronic systems. They allow users to select among a large number of bus suppliers and products, and they let suppliers—especially small ones—concentrate on a few buses to serve a defined set of user needs.

Standards come about in three main ways. The first occurs when a particular manufacturer's bus gains widespread acceptance among manufacturers who build compatible equipment to attach to it. Digital Equipment Corp.'s Unibus and Qbus and the IBM Corp.'s PC bus fall into this category.

Second, the originators of proprietary buses sometimes propose that their buses be endorsed by national and international standards groups. Bus experts then embark on a technical review of the proposed bus, which often results in some technical changes, as well as an improved description. The General Purpose Instrumentation Bus (GPIB), the S-100 bus, and Intel's Multibus have all gone through this procedure. It guarantees the stability of a specification against sudden design changes.

Finally, special-interest groups, independent of any manufacturer, work under the auspices of standards organizations to develop and define bus standards to meet their needs. Manufacturers are then encouraged by users to make free use of such standards. Camac, Fastbus, and Futurebus all originated this way.

A crucial problem for each bus standard is assuring that components connected to each other can interoperate. Not only must each component conform to the protocol standard; it must also do so in a way that ensures that all components work together. Some early buses failed this test simply because their definition was ambiguous or not sufficiently comprehensive.

A good example is the S-100 bus, first used in a variety of microcomputer systems based on Intel's 8080 microprocessor during

W. Kenneth Dawson Triumf
Robert W. Dobinson European Organization for Nuclear Research

Defining terms

Backplane: a printed-circuit motherboard with connectors placed at intervals to allow connection and communication between daughter boards.
Bus: the physical channel over which electric signals are transferred between the components of a system, along with the protocol rules governing the transfer.
Component level bus: the set of input and output pins, with defined functions and timing, through which a microprocessor sends and receives signals.
Instrumentation bus: a dedicated bus tailored for the connection of measurement and control devices.
Memory bus: a dedicated bus through which a central processing unit accesses the memory of a computer system.
Peripheral bus: a dedicated bus tailored for high-speed transfer of blocks of data between a system and its peripherals, including tape and disk drives.
System bus: a general-purpose backbone used to connect processors, memory, and peripherals to form a computer system.

0018-9235/86/1000-0049$1.00©1986 IEEE

Present and proposed IEEE bus standards

IEEE number[1]	Popular name of bus	Brief description	Typical uses	Originator(s)
488 and P488.1	General Purpose Instrumentation Bus (GPIB)	Basic bus specification (electrical, mechanical, functional rules) for very popular instrument bus	Automation of general laboratory instrumentation	Hewlett-Packard
P488.2	GPIB	Limited set of syntactic rules for structuring messages exchanged among devices	Defines bus protocols and common commands	IEEE
583	Computer-Aided Measurement and Control (Camac)	Single-master backplane instrumentation bus housed in a standard instrument crate	Real-time data acquisition and control	Esone and NIM[2]
595	Camac	Serial connection scheme for up to 62 Camac crates	Dispersed large systems for data acquisition and control	NIM and Esone
596	Camac	Parallel connection scheme for up to 7 Camac crates	Compact large systems for data acquisition and control	Esone and NIM
675	Camac	Addition to IEEE 583 specification to allow multiple masters in one crate	Front-end intelligence in Camac data acquisition systems	NIM and Esone
683	Camac	Algorithms for implementing high-speed block transfers within a Camac crate	High-performance Camac automated laboratory instrumentation systems	NIM and Esone
696	S-100 bus	Popular microprocessor system bus, used with many different processors	Mid-range microprocessor systems	MITS Inc.
726	Camac	Extension to Basic language that introduces Camac as part of a real-time system	Interactive Camac automated laboratory instrumentation systems	Esone and NIM
728	GPIB	Set of syntactic rules for structuring messages exchanged among devices	Compatibility between different manufacturers' instrumentation products	IEEE
758	Camac	Set of standard subroutines to provide access to Camac from high-level languages	Portability of applications among processors and operating systems	NIM and Esone
796	Multibus	Widely used microprocessor system bus supporting Intel and other processors	Mid-range microprocessor systems	Intel
P896	Futurebus	Very high-performance system bus independent of processors and manufacturers	Top-end multiple microprocessor systems	IEEE
P959	Input/output expansion bus	Mounting specification for daughter boards on a single-board-computer mother board	Customization of input/output for single-board computers	Intel
960	Fastbus	Ultrahigh-performance instrumentation bus	Large data acquisition and control systems requiring high speeds and multiprocessing	NIM and Esone
P961	STD bus	Low-end robust microprocessor system bus with very wide support	Simple single-processor systems	Pro-Log
P970	Versabus	Parent to VME bus (IEEE P1014) that uses similar protocols with larger cards	High-end microprocessor systems	Motorola
P981	GPIB	Resource description for GPIB instruments; specifies how an instrument functions and how it is to be programmed	Transportable high-level language programs and interchangeable instruments performing similar functions	IEEE
P1000	STE bus	Low-end system bus independent of processors and manufacturers	Simple stand-alone microprocessor systems, and front-end to high-performance systems	IEEE
P1011	Eurocard	Mechanical specifications for a range of modular subracks and plug-in modules	All buses that use Eurocard mechanics	IEEE (IEC[2] specifications)
P1014	VME bus	Popular system bus initially conceived to support the Motorola 68000 processor family	High-end microprocessor systems	Motorola
P1118	Microcontroller serial bus	Serial bus for connecting up to 100 intelligent devices, spanning distances up to 1 kilometer, at moderate speed	Low-end industrial control and data acquisition systems	IEEE
P1196	Nubus	High-performance microprocessor bus with simple protocols independent of processors and manufacturers	Lean multiple microprocessor systems	MIT[2] and Texas Instruments
P1296	Multibus II	High-performance microprocessor system bus	High-end multiple processor systems	Intel

[1] Proposed standards are identified by a "P" prefix to the IEEE number
[2] Esone: European Standards on Nuclear Electronics Committee
IEC: International Electrotechnical Commission
MIT: Massachusetts Institute of Technology
NIM: National Instrumentation Methods committee of the U.S. Department of Energy

the mid-1970s. Prior to IEEE standardization, bus timing specifications between components varied among manufacturers, causing incompatibilities for users.

The IEEE has championed the cause of standard bus specifications for many years, with some success. There are now 12 full standards and approximately 14 proposed standards.

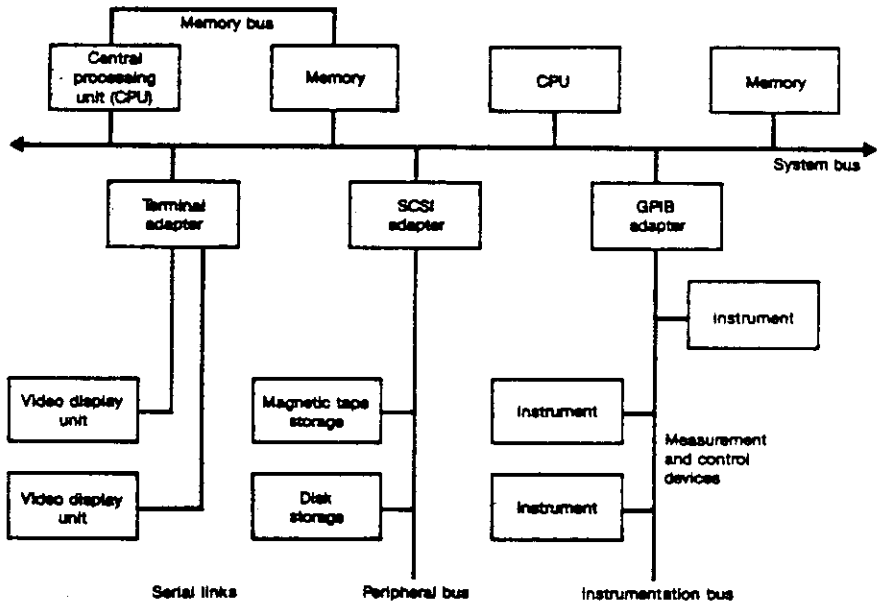
Catching the right bus

Faced with a wide choice of off-the-shelf bus standards and products, the design engineer must ask some basic questions to attain the goals set for the product. Is the goal to make a better mousetrap, or to acquire data on the movement of mice through the kitchen? The criteria used by a large manufacturer of electronic products to select a bus differ from those of the small end-user who simply wants to connect a selection of equipment. The large manufacturer may select a bus that will enable it to sell more proprietary products, while the end-user will be more concerned with selecting a bus specification that can easily be implemented in a particular application.

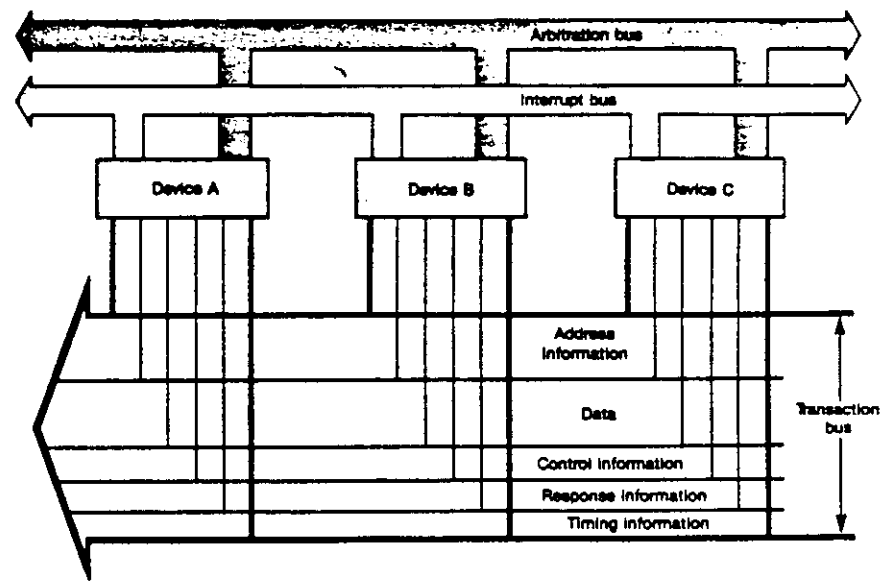
For connecting computing equipment of almost any kind, the trend today is to use international standards—those with full and freely available specifications, preferably endorsed by an appropriate standardization body—to avoid depending on any one manufacturer. The ability to "mix and match" product offerings, for example, has proved an overwhelming advantage in computer networking. This is evident in the success of such standards as X.25 and the Ethernet local-area network, which allow diverse equipment to communicate over a single network.

The case is strong for adopting a similar approach to synthesizing computing systems that use buses. But potential users of a bus standard must consider its "legal" state. It may be an accepted international standard like the S-100 bus (IEEE 696), a proposed standard like Futurebus (IEEE P896), an open proprietary standard like Digital Equipment Corp.'s Unibus, or a closed proprietary standard like DEC's VAXBI bus. Some proprietary standards may become international standards, as did Intel's Multibus.

Systems implementors will find the design process much easier if a wide selection of compatible devices is available from many manufacturers. But there is always a risk in mixing components from several sources. A classic example is the incompatibility of operating procedures adopted for many GPIB (IEEE 488) instruments. Work is now proceeding on further definitions to the standard to specify, for instance, formats to ask an instrument to identify itself to the bus. These interoperability problems will con-



[1] A complete computer system often includes a hierarchical bus structure: a backbone or system bus, and several subsystem buses tailored for specific functions. In the diagram, a high-performance central processor accesses memory over a private memory bus, and another processor is used as a front end for input/output. Adapters attached to the system bus allow access to a peripheral bus (SCSI) for disk and tape interfaces, and an instrumentation bus (GPIB) for connection of measurement and control equipment. The user interface to the computer consists of video display units attached to their own adapter over slow serial links.



[2] Data flows over a shared path—a bus—between devices according to the bus protocol, a well-defined set of rules. Five classes of signals—address, data, control, response, and timing—are used to transfer data. Together they form a transaction bus. Often address and data signals flow sequentially over the same set of physical lines. This multiplexing reduces the size of the bus and can be extended to other signal pairs, including command and response signals. An arbitration bus assures orderly access to the bus among competing devices. Requests for attention are made by devices over an interrupt bus. Arbitration and interrupts are sometimes carried out over the transaction bus.

tinue until independent conformance testing of components becomes available, and all manufacturers follow unambiguous and complete bus specifications to the letter.

Where in-house development is required, bus interface aids—bus transceivers and protocol chips, for instance—shorten the design time considerably. The compact size of such components increases the space available for application-specific circuitry.

Many designers are wary of identifying too closely with one manufacturer's product. A designer who is at the mercy of one company because of using a proprietary bus may find that the company is late in issuing specifications, or that the standard has changed significantly. Changes in accepted international bus standards are publicly disclosed during the proposal stage; manufacturers may or may not inform users of likely changes to their buses. Even when a bus is open in principle, in practice it may lack support—as well as announced products—from other manufacturers. A bus specification is not a true standard if no one uses it.

The mechanics associated with each bus system have a very real effect on the selection of a bus standard. The designer must evaluate the board size, the grouping and cooling of components, power distribution, and connector arrangements. The Eurocard system used by a number of bus systems is a promising approach that can allow several different buses to share the same mechanical specifications. It specifies a range of standard-sized interface cards and a particularly robust connector.

Designers must also bear in mind the reliability and environmental needs of the application. Products that implement a particular bus standard may vary in guaranteed reliability levels, shock and vibration resistance, susceptibility to electric noise, and operating temperature ranges.

The suitability of a bus should not be judged by data transfer rate and multiprocessing ability alone. Other fundamental bus attributes need to be considered. As noted, high performance should be avoided if it is not required; it inevitably leads to higher costs and greater complexity.

The distance spanned by a bus is particularly important in real-time data acquisition and control systems. Interconnections in such systems may extend beyond a single chassis or rack of instruments. Multiple, interconnected bus segments—as used in Fastbus, for example—offer one way of covering longer distances, although few bus specifications now incorporate such a possibility. Designers must consider the response characteristics of real-time data acquisition and control systems as system complexity and connection distances increase.

Many problems can be solved by the use of hierarchical buses [Fig. 1]. Such a hierarchical bus structure can be extended down to a single-board computer. Here, the designer can tailor a basic board with a processor and memory for specific I/O needs by adding small subboards, attached through a standardized bus.

Several major manufacturers offer families of related buses,

What a bus is and how it works: a primer

The function of a *bus* is to transfer information in the form of electric signals between the different parts of an electronic system. These range from processors on a circuit card to instruments in the rack of a test and measurement system. The generic term for system components connected by a bus is *devices*. A group of devices is connected to a *bus segment*.

Only one device, the *bus master* or *commander*, is allowed to control the bus and initiate data transfers at any one time. Several masters may be present on a bus, so contention between them must be resolved using *arbitration rules*.

A commander either oversees the transfer of data between devices, or, more usually, exchanges information with one or more *slaves*. The commander selects slaves by placing *address* information on the bus; each slave compares this information with its own address. If they match, the master and slave establish a *connection* and the slave becomes a *responder*. Addresses that identify more than one slave are called *broadcast addresses*. A particular device may act as both master and slave, but not at the same time.

Once the slaves are connected, the commander exchanges data with them over the bus. The commander breaks the connection with its responders after all the data has been transferred. This sequence of actions—making a connection, transferring data, breaking the connection—is called a *transaction*. A commander may either relinquish bus mastership after a transaction or continue with other transactions.

Bus transactions are synchronized by *timing* information that indicates when address information and data are valid. Most buses also allow a commander to transmit *control* information that specifies the type of transaction it is about to initiate. Responders can indicate whether data was received or acted on by returning *slave status responses*.

The set of rules governing the use of the bus by masters and slaves is called the *bus protocol*. The volume of information that flows back and forth between devices following a bus protocol is large. To reduce the number of separate bus lines required, various multiplexing techniques can be used. Often the same lines are used for both address and data information. Lines used to transfer data are usually bidirectional—used for both read and write operations. Other common multiplexing schemes involve sharing either control and response lines or control and data lines. Designers must balance the risk of reduced performance when multiplexing against potential cost savings from using fewer bus lines.

Parallel buses either provide a dedicated line for each bit used in the protocol or use simple multiplexing schemes. *Serial buses* multiplex all information over one or more physical channels.

Bus *arbitration* determines which master receives permission to transmit over the bus. Any scheme for bus arbitration involves two considerations: Where does it occur—is the logic centralized or distributed? How is it done—what are the rules by which bus mastership is allocated?

Centralized arbitration schemes use a single arbitration circuit to determine which master is granted use of the bus. Each master may have separate grant and request lines, or a common request line and a single daisy-chained grant line may be used to reduce the number of bus lines. An intermediate scheme with several pairs of request and grant lines increases the flexibility of a bus in allocating priorities.

In one form of *distributed arbitration*, each competitor monitors all request lines and transmits over its own. Circuitry in each master allows the one with the highest priority to take control.

Many different priority schemes can be used. Some allow a high-priority master to deny bus access to lower-priority masters indefinitely. Others introduce fairness schemes, which do not allow "bus hogging." Designs with separate arbitration and transaction buses allow arbitration to occur in parallel with data transfer. This minimizes the delay between the time when a master releases the bus and the next master takes over, by selecting the next commander in advance.

Having gained control of the bus, a commander carries out a *bus cycle*—the exchange of information between commander and responder. The information may be an address, a word of data, or both. Timing signals, also transmitted over the bus to indicate when the information is valid, are generated either synchronously or asynchronously. Several bus cycles may be needed to make up a transaction.

In an *asynchronous bus* system, the commander issues a *strobe* timing signal to indicate that subsequent information on the bus lines is valid. The responder returns an acknowledgement timing signal. Receipt of this signal informs the commander that the responder received and acted on the information. This strobe-acknowledge mechanism is called a *handshake*. In multiplexed bus systems, separate strobe-

including Intel's Multibus I and II and Motorola's VME family. Each member serves a different function in system design, and all can be used together easily and flexibly. At first sight, this approach can appear attractive to systems integrators. But there are potential pitfalls that the designer must consider before committing to one bus family or another. Some family members may be highly proprietary; neither information nor protocol circuits may be available to support interfaces to these standards.

Meet the families

There are three major bus families in the marketplace today: a partly open, partly closed set from DEC, used in the company's minicomputer products, and one family each from Intel and Motorola, manufacturers of electronic components. Intel and Motorola developed their buses to support board-level microprocessor systems. However, members of each family were later accepted as international standards and are now published, open specifications.

The DEC family includes the well-known and much-used Unibus and Qbus, which have essentially open specifications. A wide range of devices using these buses is available. Less useful tools for the systems implementor are the Synchronous Backplane Interconnect (SBI) bus used on the VAX 11/780 minicomputer, and the Massbus for peripherals. They were designed for much more specific purposes, and remain proprietary.

The recently introduced VAXBI bus will be used through the 1990s, according to DEC, but unlike its support of Unibus and Qbus, the company has limited the availability of information about the VAXBI bus. The protocol chips are currently provided only to selected customers. Systems implementors may hesitate to make use of such a closed system, even if interfaces to open systems—in this case, Unibus and Qbus—are provided.

Intel's Multibus I (IEEE 796) was one of the early, widely used system buses. A family of subsystem buses, including a memory bus, high-performance peripheral bus, and an I/O expansion bus (IEEE P959), has grown up around the original system bus.

To cope better with modern multiprocessor systems, Intel introduced Multibus II, which has a number of innovative features when compared with its predecessor. These include message passing—an operating mode that uses write-only block transfers, which can be considered a form of local-area networking over a backplane bus. The Multibus II family includes several members of the original Multibus I family as well as a proposed serial bus. Adapters to link the Multibus I and II systems are available. Many devices are compatible with Multibus I, but as yet there are far fewer for Multibus II.

The VME bus (IEEE P1014) was developed by Motorola, along with its parent, Versabus (IEEE P970). The VME family was originally introduced to support the 68000 series of microprocessors, although other processors have been packaged into this standard.

acknowledge signal pairs are used for address and data. Hence the strobe itself can identify the type of information on the bus lines. Asynchronous systems can take full advantage of the speed of the fastest responding devices without needing to consider the slow ones.

In most *synchronous bus* systems, a central clock generates timing signals that are distributed to all devices on the bus. Changes in the state of bus lines occur at fixed intervals. The duration of a bus cycle is set by the clock speed, which in fully synchronous systems is constrained by the slowest device connected to the bus. Most synchronous buses use a *wait protocol* to avoid this constraint. For example, any responder that cannot respond to a request at the basic system rate simply indicates that the system should pause. When the responder is ready, the pause is canceled and the bus resumes normal operation.

A commander transmits address information over a bus to establish a connection with one or more slaves. *Geographical addressing* selects a slave at a particular physical position on a bus segment. *Logical addressing* specifies one of a global range of storage locations—for example, a memory location—without explicit regard to its physical position within the system. A device that sees this type of address on the bus will compare the address with the set of logical addresses allocated to it. If a match is obtained, it becomes the responder. *Broadcast addressing* is used to select multiple responders. Different broadcast addresses are used to select sets of slaves—for example, all idle slaves might become responders.

In general terms, *control information* is transmitted by a commander to specify what is to be done, and the timing lines specify when to do it. Many bus systems have additional control lines that affect all devices unconditionally. These lines allow unaddressed control functions like inhibit, clear, initialize, and power-failure warning.

Slave status responses may be given by a responder to a commander in addition to data. Response information varies greatly from bus to bus. It may include reports on the state of the address connection, the ability of the responder to respond, and an end-of-block-transfer indication.

In many situations, devices attached to a bus require the attention of other devices. They may have data to transfer or they may have completed some action. Various schemes have been devised for passing a *request for attention*, or *interrupt*, across the bus.

The simplest method uses a single service-request bus line. Devices requiring attention transmit an anonymous request over this line, which is monitored by an *interrupt-handler*. When a service request is detected by a handler, it polls all slaves in turn to identify which device requested attention. This procedure is speeded up if a device with an outstanding request identifies itself.

Another interrupt-handling scheme allows devices that require attention to become bus commanders and transmit interrupt messages to the required destination. There are clear advantages to this approach. Because the procedure is fully distributed, it is less susceptible to total failure. And requesters are allocated to handlers dynamically, making it particularly suitable for multiprocessor systems.

Two fundamental factors influence the reliability of bus systems: corruption of signals during transmission over the bus, and faults in devices attached to the bus. *Transmission errors* may be introduced by noise and crosstalk while information travels over the bus. The problem becomes more pronounced at higher transmission speeds. It can be reduced by attention to the electrical design of the bus and the transceiver circuits that transmit and receive signals over it. Modern buses—those with cycle times of less than 100 nanoseconds—usually have error detection schemes to generate and check one or more parity bits for each bus cycle.

Faults, as opposed to transmission errors, occur in devices attached to a bus. A key consideration is how vulnerable a system is to a single point of failure and how easily the system may be reconfigured. Systems with distributed arbitration are less vulnerable to faults than those in which the arbitration is centralized. Systems that support software reconfiguration can isolate faulty devices or replace them with standby units.

A basic bus protocol defines the rules for transferring bytes or words of data between devices. Further rules can determine the syntax and semantics of transactions performed over the bus. These rules can be enforced with software procedures invoked from high-level computer languages. Thus buses can have a hierarchy of protocols, similar to the layered architecture of networking designs. Many bus designers feel that the solution to problems of interoperability between devices on the bus lies in this concept.

—*W.K.D. and R.W.D.*

The family contains buses for accessing private memory and peripherals, as well as a serial bus. Electronic designers have widely accepted the extensive range of VME-based products.

Two mature IEEE standards for connecting data acquisition and control equipment to host processors also include families of bus specifications. The GPIB (IEEE 488), originally developed in the early 1970s by Hewlett-Packard, is particularly suited for small systems that span modest distances. The Camac standards, which are roughly contemporary with the GPIB, were developed by special-interest nuclear science groups. Camac supports multiple-crate segments that can be linked by a parallel or serial "intercrate highway." A wide choice of third-party products of all kinds, including numerous adapters to popular system buses, is available for GPIB and Camac. Some software standards to support these buses already exist; others are in preparation.

Bit-serial instrumentation buses often provide an effective way to interconnect a highly dispersed set of devices. Serial Camac provides a synchronous bit or byte serial loop at up to 5 megahertz, which interconnects up to 62 crates of equipment. Bitbus is a serial bus introduced by Intel in 1984 and now a candidate for standardization by the IEEE as P1118. It is aimed at low-end industrial control applications, and offers a cheap way of interconnecting intelligent devices over distances up to a few kilometers. Its data transfer rates, however, decrease as the distance spanned increases.

The line between a bit-serial bus and a local-area network (LAN) may often seem blurred. In general, serial buses of this type are targeted at an immediate response to transmitted information, while LANs usually are not.

IEEE 960 (Fastbus) is an advanced modular data-bus system for data acquisition, data processing, and control. It is intended to satisfy high-performance and large systems needs. A unique feature is its fundamental multisegment nature. Segments operate independently, but can link together to support intersegment transactions as needed. Fastbus is used primarily in nuclear and particle physics laboratories and in medical imaging.

Buses for microprocessor systems

A systems implementor has several choices of bus for small systems, all well supported by third-party products. The S-100 bus (IEEE 696) was one of the first microprocessor buses. This very popular standard is an evolution of the 1974 Altair 8800 8-bit microcomputer bus, which served an Intel 8080 processor. A de facto standard was quickly established as many manufacturers offered boards for the bus, but product incompatibilities caused difficulties for users. A standardization effort was launched by S-100 designers through the IEEE. S-100 was cleaned up, well specified, and expanded to 16 bits, with 24-bit addressing capability. Hundreds of products from many manufacturers, including many processor boards, are now offered for this bus.

A second choice, the STD bus (IEEE P961), is roughly contemporary with the S-100. It, too, commenced life as a manufacturer's product; Pro-Log used it to package a variety of 8-bit microprocessor systems. From these beginnings, STD has grown into a widely available standard particularly suited to low-end systems. Its capabilities were enhanced during the formal standardization procedure to incorporate 16 bits of data and 24 bits of addressing.

In Europe, the G-64 bus is aimed at a similar market and has become very popular. It uses Eurocard mechanics.

The introduction of the IBM Personal Computer (PC) has had an impact on bus standards. The PC's system bus, closely related to the Intel 8088 component level bus, has become a de facto standard, although it is not an IEEE standard. In addition to numerous plug-compatible memory and peripheral boards, manufacturers offer adapter boards to connect the PC bus with many other standard buses, including Camac, GPIB, and Multibus.

The PC bus has made serious inroads into many areas previously occupied by other buses, including data acquisition and control. The open system approach and widely available hardware and software of the IBM PC provide an attractive building block for third-party manufacturers and systems integrators.

Futurebus (IEEE P896), a next-generation standard, was designed by an IEEE committee as a state-of-the-art, high-performance, multiprocessor system bus. It has several innovative features, including built-in protocols for maintaining coherence among multiple shared memories. The data transfer rate is reputed to exceed 100 megabytes per second, stemming from careful design of its backplane and bus transceivers. The specification has not yet been released for general use, so products are limited, but their number is expected to grow rapidly.

Another next-generation bus, Nubus (IEEE 1096), was sponsored for standardization by Texas Instruments. It is another high-performance bus designed to be processor-independent, based in part on its simplicity. It has so many features in common with Multibus II that a merger of the two standards was proposed at one time. However, Nubus has a leaner protocol than Multibus II. In contrast to the subsystem bus philosophy of both Intel and Motorola, Nubus has no associated memory, I/O, or serial buses. It thus has a certain appeal to potential users who favor a stripped-down approach to multiprocessing.

In 1982 the IEEE microprocessor standards committee recognized a growing need for a single 8-bit data bus that would be both processor- and manufacturer-independent. The STE bus (IEEE P1000), based on Eurocard mechanics, has been developed to meet this need. Its protocol is easily implemented, and the bus is suitable for low-end stand-alone systems and as a front-end I/O channel for higher-performance buses.

The Small Computer System Interface (SCSI) is an American National Standards Institute standard for the connection of computer peripherals—disk and tape drives, for instance. It is rapidly achieving widespread acceptance in the industry, and it is used in over 100 products from a variety of manufacturers.

One question that has not been fully addressed is the need for bridges between different bus standards. Many designers in the industry feel that some form of standardized serial bus will provide the solution to this problem.

To probe further

IEEE Micro is a good source for news about bus standards. The August 1984 issue contains several articles of interest, including a bus tutorial, together with reviews of Futurebus (P896) and the STE bus (P1000). It is available through the IEEE Computer Society, 10662 Los Vaqueros Circle, Los Alamitos, Calif. 90720 (telephone 714-821-8380), or the IEEE Service Center, 445 Hoes Lane, Piscataway, N.J. 08854 (telephone 201-981-1393).

For further information on IEEE standards activities, contact the Secretary, IEEE Standards Board, 345 E. 47th St., New York, N.Y. 10017.

The characteristics, mechanical specifications, and pinout tables of several popular industry data bus standards are listed in a free manual available from the Hybricon Corp., 410 Great Rd., Littleton, Mass. 01460 (phone 617-486-0311).

About the authors

W. Kenneth Dawson (M) heads the technology and administration divisions at Triumf, a Canadian national laboratory for intermediate-energy nuclear physics. He is on leave from the University of Alberta, where he has been a member of the physics department since 1959. He is secretary of the IEEE Nuclear and Plasma Sciences Society and was the technical editor of the recently approved Fastbus system (IEEE 960). He received a doctorate in nuclear physics from Queens University in Ontario in 1955 and has been involved with the use of computers as a tool for nuclear physics since 1965.

Robert W. Dobinson (M) is a senior physicist at the European Organization for Nuclear Research in Geneva, Switzerland. He has worked extensively on instrumentation and data acquisition systems for particle physics experiments, and recently contributed to the design of the Fastbus (IEEE 960). He is presently on leave at the Triumf laboratory. He received a Ph.D. from the University of London in 1968. ◆