

Interfaces for a millwide information system

Robert J. Bolm

The millwide information system communicates with each of the mill's measurement systems through an interface. System efficiency is largely determined by the performance of the interface.

The earliest distributed control systems (DCS) were designed to collect data and control manufacturing processes. These early systems lacked the ability to store data, and operators had to manually track the effects of process modifications on log sheets. As DCS systems became more sophisticated, a limited amount of data-storage capacity was added. With the ability to retrieve historical data, operators could monitor the parameters for each process and observe the effects of process modifications. However, the data storage-and-retrieval capacity of the DCS was still incapable of providing the information needed to help operators cope with process upsets.

The ability to store data from DCS systems, defect detectors, and other process-information systems in a single location has obvious benefits. Process upsets can be diagnosed swiftly, and the effects of process modifications are easily tracked. However, even mills with the best DCS systems lack the ability to obtain all process information from a single source. Our millwide information system was developed to facilitate storage and retrieval of the enormous amounts of information generated by all measurement systems within the mill. Information is kept on-line for one year.

The following discussion of millwide information systems is supplemented by a glossary of specialized words at the end of this paper.

Millwide information systems

The information requirements of any production environment can be divided into two categories: (a) information about the manufacturing process and (b) information about the product being manufactured. Information about the production process is "real-time" information—a snapshot of a particular piece of process data, such as the amount of material being used or the flow rate of a given stream. Information about the product is "batch" information, which is generated as the product passes from one stage of the manufacturing process to the next. Examples of batch information include run number,

average weight, or grade. A millwide information system must be capable of storing and retrieving both real-time and batch information.

Mead's millwide information system

We chose an event-based, real-time database software package developed by Oil Systems, Inc., and a relational database developed by Ingres. Our system—QUPID (quality and production information for decision support)—has been in the development and implementation stages for two years. The entire millwide system includes a VAX 6320 computer, a VAX 8550 computer, four VAX 3600 microcomputers, and a host of interfaces to collect and store data from Foxboro, Accuray, Measurex, Ulma, Boyle, Lippke, and Avtron measurement systems.

The collected information is stored on-line for one year in a central location so that it can be retrieved via terminals located throughout the plant. Information is transmitted through a Decnet (Ethernet) network that was installed specifically for the millwide information system. Access to the system is obtained through Tektronix terminals or through PCs running Tektronix emulation software. Each user is connected to the network via an RS-232 communications port on a DecServer. Local area networks have access to the system through asynchronous communications servers connected to DecServers.

The interface between a measurement system and the millwide information system has the responsibility of sifting through the data to find exceptions. When an exception is encountered, it is transmitted to the host computer, where it is submitted to a compression algorithm and stored in a historical archive.

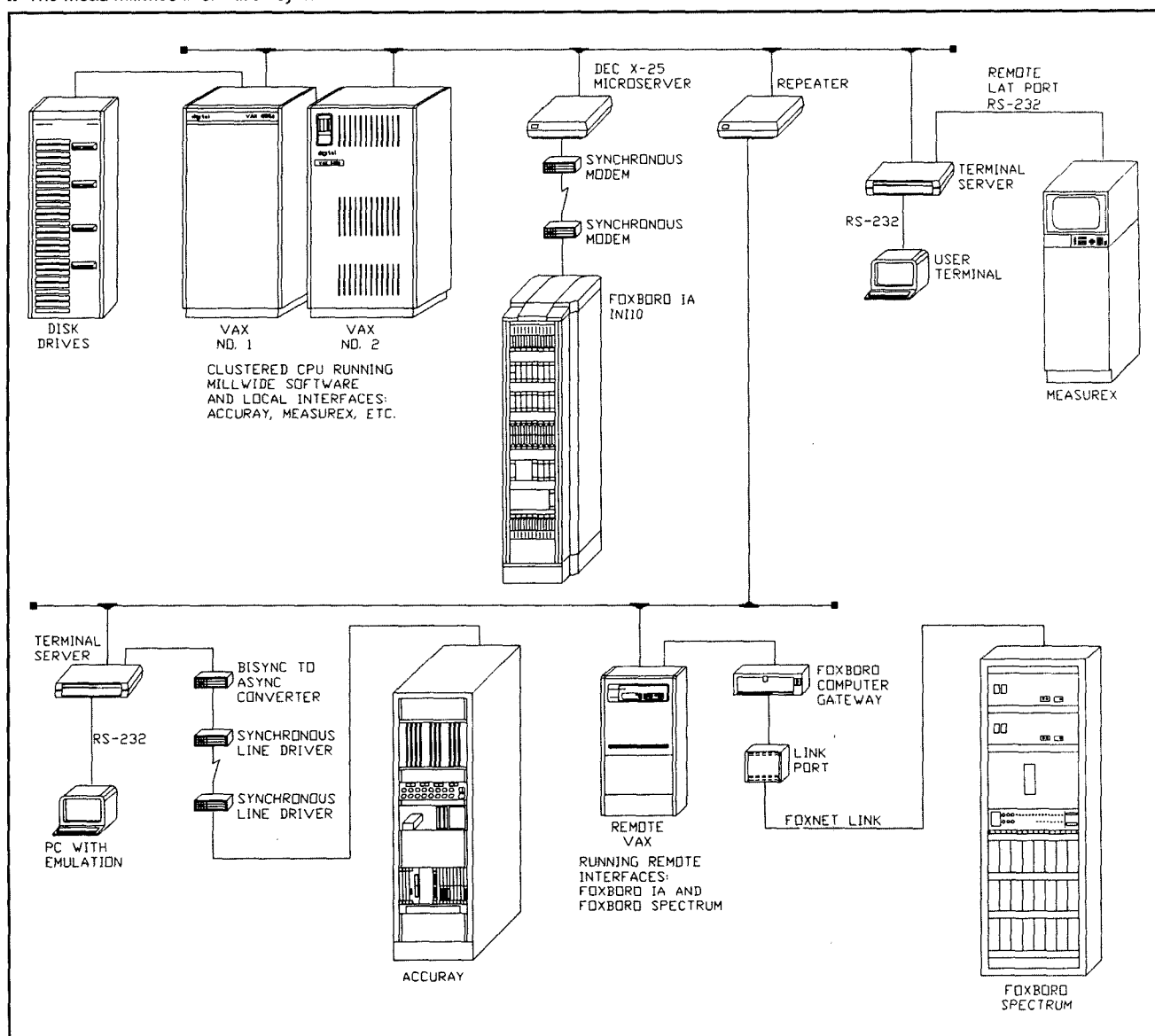
About the data

All data from a measurement device are transmitted through a port provided by the vendor of the device. The millwide computer must be ready to intercept the information being broadcast by the measurement device or the data will be lost. The interface is the means by which each measurement device communicates with the millwide database.

An interface can be either passive or active, depending on the type of data. Scanned data (passive interface) require the most resources, including disk and network

Bolm is senior process control systems engineer at Mead Paper Co., Publishing Papers Division, Escanaba, Mich. 49829.

I. The Mead millwide information system



communication activity. Polled data (active interface) require a more sophisticated program that allows transmission of data to and from the foreign device at a configurable rate.

About interfaces

The interface is the most important element in the successful implementation of a millwide information system. If the data are important enough to save, then they must be made available in an accurate, easy to use, and timely fashion. To accomplish this, it is imperative that effective interface specifications be written, reviewed, revised, and adhered to. There should be no surprises when the interface is ready for installation on a millwide system.

Data from most measurement devices are not broadcast in a form that is usable by any database-management system. It is the job of the interface to receive the data,

extract the needed information, and transmit it in a useful format.

Many scanning interfaces will patiently monitor a port's activity for long periods, collecting bits and bytes of data as they are monitored by the device. Other interfaces must capture "turn-up" data that occur as a blast of information for 30 seconds once each hour.

The interfaces utilized on our millwide systems vary widely in design. The Foxboro Spectrum DCS interface utilizes the Foxboro Computer Gateway (FCG). The gateway (protocol converter) connects the Foxnet network directly to an asynchronous port in a VAX. Another system—the Foxboro I/A DCS—uses an X.25 packet-switching protocol for communications. Synchronous modems were installed from the I/A (intelligent automation) device to a packet-switching interface (PSI) gateway connected to the millwide host computer. Foxboro I/A interface software running on a remote node (two Ethernet segments away) does all the communicating with the I/

A device. The millwide interface software simply calls the Foxboro software to get exception information on designated process variables.

The Accuray interface utilizes a logical assigned terminal (LAT) connection on a DecServer with the port dedicated as remote. The information is broadcast in EBCDIC through a bisynchronous port on the device to a gateway, which converts the data to asynchronous ASCII. The interfaces for the Foxboro Spectrum DCS and the Accuray system are run on the same mainframe that runs the database-management software.

The Ulma, Avtron, Boyle, Lippke, and Measurix interfaces are also set up to use a LAT port on a DecServer. The measurement devices all provide an RS-232 port and ASCII data packets that can be directly ported to a DecServer. For the most part, the interfaces are run on remote VAX 3600 microcomputers to help distribute the computational load over the network (Figure 1.).

We have found that interface development proceeds more smoothly when the programmer has access to production data from the device to be interfaced. Most developers use a PC to transmit packets of information that have been captured from the measurement device. Debugging of the interface can then proceed either on the production machine (not usually the best idea because of inherent software bugs) or on a third-party development system that may be thousands of miles away.

Interface specifications

Interface specifications must be thorough. Leaving the smallest bit of information out of the specifications will ultimately lead to a misunderstanding, or at least to different interpretations. The specifications should include all the necessary information about the bits and bytes being transmitted from the measurement device, and also whether the interface should be active (i.e., the interface program may poll the device) or passive (i.e., the interface simply waits to collect information transmitted by the device).

Corrupt data

Problems with noise on the communications line occur frequently, so the interface must be able to distinguish between a good data packet and one that is corrupt. Active interfaces are usually the preferred way to get data because of their ability to rebroadcast to the device if the transmitted response was not what was expected.

Passive interfaces usually require a program with more built-in "intelligence" than an active interface. If a data packet is corrupt, the interface must first try to discern what was being transmitted, then dissect the data from the packet. If the packet is bad, the interface program must then decide what to do with the data. Usually the data packet is simply thrown out or ignored. In some cases, the program may be able to extract some useful information for entry into the database.

Location of interface

The specifications also should consider the location of the installed interface. Will the interface be run on the same node as the database, or will it be run on a remote node on a distributed system? Most programmers will write the

code as if the program will reside on the same node as the database. Most millwide systems, however, consist of multiple computers. The specifications should address where the interface will be installed. The location of the interface is usually determined by the location of the measurement device as well as the need to balance the usage of the CPU (central processing unit) on the cluster. A remote interface will require more programming, but in the long run, the system manager will have many more ways to tune the system, and the response time will be enhanced.

Coping with hardware/software failures

The interface specifications must address how to resolve problems with remote computer failures, network failures, or simple disk failures. The programmer must know if there is to be a redundant interface program on another node. If the first program fails to communicate properly with the home node or the device, what will be the role of the second interface? The programmer also must decide how fault tolerant the interface is to be. If the interface is running on a remote node, can data be buffered on that node until communication is reestablished with the host? Can the measurement device buffer the data until communication with the device is restored? If a redundant interface starts because of lost communications between nodes, will two sets of data be transmitted to the host?

The interface must be able to handle a host of mechanical problems as well as data problems. The device may lose communications with the DecServer because of a detached connector. Or the DecServer may lose power and lose the data being transmitted to it (unless the server is equipped with an I/O buffering device). The server communicating with the interface can be located on an entirely different Ethernet segment, connected via a bridge or repeater. In this case, redundant repeaters are a good insurance policy.

Each network device is connected to the backbone of the network via a transceiver, which can fail both electrically and mechanically. The CPU also communicates over the network. The computer may lose connection to the network but still talk to the device through an asynchronous port. The interface must be able to determine what to do when the network is restored.

The interface must be able to retry communications to the device at a predetermined frequency; it should not just fail once and give up. It is a system manager's nightmare, especially on a distributed system, to keep track of all the programs that should be running but may not be because of some type of failure. The interface should be able to leave a message in some predetermined place or file whenever a problem occurs. This is a valuable tool when trying to find out why an interface failed at 2 a.m.

A set of tags should also be configured in the interface to report back on the data rates being transmitted from the device. An abrupt change in data rates is a sign of a potential problem. Over time, a system manager can establish statistics on the acceptable high and low data rates from a given interface. If the data rates fail to fall within these limits, the program can instruct the system to report the potential problem to the system manager for analysis.

The system manager should be able to change the settings for the rate of data collection. This prevents an

active interface from loading down the measurement device. The system manager also will want to control network activity to the home node if the interface is remote. Control over both these parameters will help the system manager tune the system loading.

The interface must be able to flag the database with messages such as "I/O timeout" if the device fails to respond within the designated response time. The interface also must be flexible enough to accommodate changes in the database, the points to be monitored, or the parameters of an existing point.

Exception reporting

It is imperative that the interface report to the database by exception instead of reporting all collected data. The load on both the network and the host CPU will be reduced if the exception reporting is accomplished at the remote node. This exception is usually set up as a configurable parameter in the target tag definitions, and it should be part of the information collected from the host when the interface polls for the tags it wishes to use for the data. If the exception task is contained within the interface program itself, response time can become sluggish. Such problems can be avoided by running the exception task as a separate program.

Response time

When the interface specifications are complete, they will contain goals of response time from the program to the database. The types of data that are expected, such as real, integer, or digital, should be well defined for each piece of information to be saved.

Communication protocols

There are no standards for establishing communications between devices and computers. Vendors typically insist that their systems can import data from other sources, but they are reluctant to help export data from their systems to foreign computers. In one case, data had to be intercepted while being transmitted to a local printer in order to get the data into the millwide database. Vendors are becoming more sensitive to this situation and have begun to provide a means of transmitting data. Unfortunately, the lack of industry standards has forced most of them to continue recreating the wheel.

Many protocols will be encountered as each device is investigated for addition to the millwide information system. Do not be content to use packaged systems that provide a "black box" gateway. Look beyond the obvious for ways to interface to different systems. Usually there is a solution that, with a little ingenuity, will provide a better, more reliable interface, without the aggravation of the finger pointing between the "black box" vendor and the designer of the database-management system. We have had many sessions with vendors regarding interface problems that have been experienced locally and that the vendors deny having at their location.

Developing the interface programs

There are several approaches to the task of developing an interface between a DCS and a millwide information system. Interface programs can be developed by the

vendor, by a systems integrator, a contract programmer, or an in-house programmer. The last two options are the preferred choices, although all development sources may encounter difficulties when database vendors and device manufacturers upgrade their products. No matter who does the development work, the system manager must be involved, since this is the person who ultimately bears the responsibility for making the system work.

Vendor

The vendor of the database-management system can be called upon to develop the interface. This is an expensive option, but for the most part will yield a working interface that is both reliable and totally compatible with the database software. The downside is that database vendors are in business to sell databases, and if faced with a choice of working on your interface or on new enhancements to their database systems, they will choose the latter. When the job is complete, the vendor can resell the interface to your competitor after you have paid the price for initial development. On the other hand, many vendors will make allowances for the development costs and retain the rights to the software code. It is up to you to decide if you want the vendor to be responsible for upgrading and enhancing the code, or if you want to purchase the code so you can take control and make any needed changes. It is also a good idea to have the vendor provide a way of recompiling the interface code should this become necessary following an upgrade of the CPU's operating system.

Systems integrator

A second source for development is the systems integrator. While this provides a timely resource for development, it leaves an open invitation for use of a developer's personal choice of "black box" software. Be very explicit about the developer's future responsibility regarding the interface.

Contract programmer

One disadvantage of hiring a contract programmer is that the individual will have to go through the normal learning curve before being prepared to develop a working interface. When the job is complete, however, you will own the source code and will have the ability to modify it should a new resource from the measurement device become available. A second disadvantage is that you may have to go through the same initial training pains with a new contract programmer when a new interface is required. At any rate, you ultimately will become responsible for the operation of the interface, which means that someday you will probably have to crash into the source code to debug a problem that didn't show up while the contract programmer was being retained. You also will be responsible for keeping the interface current with upgrades and enhancements to the system.

In-house programmer

The in-house programmer is by far the most palatable choice for development. The problem is that the system people, who are the most intimate with the database-management software, are also creating the custom applications and usually become too loaded with maintenance and training issues to devote much time to interface development. The in-house programmer also will take

Glossary

- Accuray®** On-line measurement device manufactured by Combustion Engineering
- Active** An interface that solicits information from a measurement device
- Archive** A storage area reserved for historical data
- ASCII** American Standard Code for Information Interchange, an alphanumeric information code for data processing
- Asynchronous** Transmitted characters have a start bit and a stop bit to indicate the beginning and end to the data bits
- Avtron®** Manufacturer of on-line Speed and Draw measurement device
- Backbone** A coaxial cable used as the main communication medium for Decnet
- Binary** On/off or yes/no digits
- Bisynchronous** Bit-oriented protocol, with control over who may transmit and for how long
- Boyle®** Off-line profile-measurement device manufactured by Data Measurement Corp.
- CPU** Central Processing Unit
- DCS** Distributed Control System
- DEC®** Digital Equipment Corp.
- Decnet** Digital Equipment Corp.'s implementation of the Ethernet/IEEE 802.3 network
- DecServer** DEC device that connects to Ethernet for terminal support
- EBCDIC** Extended Binary-Coded Decimal Interchange Code
- Ethernet** A local area network facility permitting high-speed data interchange
- Exception** A value that is not reported unless it falls outside defined values or time specifications
- Foxboro®** DCS system
- Gateway** Protocol converter
- Ingres** Developer of Relational Technologies®, a relational database
- Interface** A device that bridges different systems
- I/O** Input/output
- LAT** Logical Assigned Terminal
- Lippke®** On-line data measurement device
- Measurex®** On-line data measurement device
- Millwide** A central database for quality and process information gathered by various systems throughout the mill
- Passive** An interface that waits for information to be transmitted from a device
- PI®** Process Information, a real-time database developed by Oil Systems Inc.
- Poll** To search for data
- Protocol** Set of communication rules that allows for the orderly exchange of information
- QUPID** Quality and Production Information for Decision support, the millwide information system developed at Mead Paper, Escanaba, Mich.
- Real time** The transmission of data without delay
- Redundant** A backup system duplicating the function of a device
- Relational** Refers to a method of relating information in database files
- Remote** Program control from some distance
- Repeater** An Ethernet device that connects base-band Ethernet coaxial segments together
- Scan** To examine a channel for input/output activity
- Segment** An Ethernet coaxial cable with a maximum length of 500 m
- Synchronous** Sending and receiving parties operate on a common timing element; does not depend on start and stop bits for timing
- Tag** The name assigned to a set of data; used to gain access to historical database
- Tektronix®** Computer terminal vendor
- Ulma®** On-line defect detector supplied by ABB Industries Systems Inc.
- VAX** Virtual Address Extension (32-bit address) from DEC
- X.25** A packet-switching network protocol for data transfer

more time to develop the interface because of interruptions required by routine maintenance on existing software or by other development projects. A large investment in time and training will be required before an in-house programmer can even begin to develop code. However, the benefits are there. The programmer will immediately become very aware of the abilities of the database interactive software. The programmer also will become an important focal point for future enhancements to the system because of the knowledge gained. The efforts made during development stages will transfer to many other custom applications that will need to be generated. If a bug crops up, the in-house programmer will be able to track down the source of the problem.

Testing the interface

The three most important issues involved in inspecting an interface are test, test, and test. It is important to set a test plan well in advance for installation and implementation of the interface. In fact, it is a good idea to include the test plans in the initial specifications for the interface, so the developer will have a good understanding of what is expected. Changes to the specifications should be well thought out and documented to avoid confusion in the programmer's development strategies. Changes to the overall intent of the specifications will delay the release date and possibly raise the cost of development.

Testing should be comprehensive. Plan on testing for hardware failures (disk, cables, boards, and devices). Look at software weak points, including the application driving

the interface, and third-party software that may be feeding information to the interface program. Don't leave out network points of failure (cables, gateways, repeaters, power problems on servers, etc.). Testing also must include a fair amount of data verification to check for accuracy, speed, flexibility, and integrity. The data are only as good as these factors allow.

Some thought should be put toward synchronization of the data. Should the data reside in the real-time or relational database? Some data may reside in both places to accrue benefits derived from applications developed for both systems. In this case, there should be some way of ensuring that the data in both locations are correct. This is a difficult task to accomplish on some systems and may require some development resources in addition to the original interface.

The system manager will want to use the initial installation time to look at system loading and interface data rates. Adjustments should be made early to avoid problems with data loss and irate users who will need the data as soon as it becomes available.

Conclusion

A millwide information system collects data from distributed control systems, defect detectors, and other

process-information systems. The interfaces between the subsystems and the millwide system largely determine the utility of the information retrieved from the millwide database. An effective interface will enable the millwide system to collect accurate data in an efficient and timely manner, providing users with the process information needed for decision support.

This paper examines the development and installation of a single millwide information system at Mead Paper. The concepts presented here are hardware-independent and are intended to provide a general guideline for the consistent preparation of an interface. Specific issues such as choice of communications protocol, methods of checking data integrity, and selection of standards used to establish the interface specifications are matters that must be decided at each millwide site to benefit the system as a whole. □

Received for review Nov. 9, 1990.

Accepted Nov. 13, 1990.

To be presented at the TAPPI 1991 Process Control Conference.