Med TechNet

Online Continuing Education for the Clinical Laboratory Professional

Med TechNet Online Services✦ East Amherst, NY October 7 - November 4, 1996

Interfacing the Clinical Laboratory: A Primer for LIS Managers

John Selmyer and Bruce Cloutier

Dawning Technologies, Inc.

At a glance ...

Introduction

The contemporary clinical laboratory is a production environment that requires the coordination of many people, automated analytical instruments, patient specimens, supplies and procedures. The clinical laboratory is essentially in the information business, responsible for translating physician orders into test result reports. Each step in laboratory operations generates data affecting the information sought: tracking patients from whom specimens are to be obtained, identifying specimen containers at the bedside, assigning identification (ID) numbers to specimens received in the laboratory, preparing and identifying specimen aliquots for distribution to laboratory workstations, programming instrumentation to perform to ordered determinations, matching test results with specimen IDs, assembling collated reports for each patient specimen, and so on. The tremendous volume of data processed by the clinical laboratory has long been a source of concerns and problems to be overcome.

The business of operating a clinical laboratory has changed dramatically in recent years, in response to changes in market forces. The factors influencing current clinical laboratories include the charter of providing vital services (information) to the clinical staff, pressure to report this information ASAP, pressure to combine accuracy and precision with speed, economic constraints on staff resources, continuing pressure to define and reduce capital and operational costs, increasing regulatory requirements, the transition of the hospital laboratory from a profit center to a cost center under changing reimbursement guidelines, and the need to process billing information expeditiously. The current political environment promises to deliver additional pressures under the label of healthcare reform.

It has long been recognized that the clinical laboratory is a primary generator of data transactions within the hospital environment. Even early capacity studies indicated that the laboratory is responsible for 45% - 60% of the total number of transactions. Advances in laboratory technology, while improving analytical instrumentation and techniques, have not altered the fundamental transaction intensive environment of the clinical laboratory.

Traditional paper based laboratory data processing systems are slow, labor intensive, error prone and do not help meet changing regulatory reporting requirements. Paper based reporting systems generally are not adequate in the face of current clinical and business pressures.

One common remedy to these influencing forces is the implementation of a laboratory information system (LIS). In theory, an LIS can dramatically reduce the clerical labor required to handle information, increase accuracy of patient reports, and significantly speed their production and availability to the clinical staff. An LIS should also be of considerable help in analyzing the quality of patient test result information and in producing records compliant with regulatory requirements.

The extent to which an LIS achieves these general objectives is the sum of many elements of hardware and software, often from multiple sources. An interface to a hospital information system (HIS) can provide an efficient means of downloading patient demographics and order entry information and uploading completed patient reports. LIS software provides for assignment of specimen IDs and creation of worklists for the various laboratory workstations. Test results are produced by the array of clinical analyzers, which must be captured by the LIS. Once result information is in the LIS it can be manipulated according to range and delta check limits, reviewed, and verified prior to release. Completed patient reports are then printed and/or transmitted back to an HIS for distribution to the patient location.

The majority of information managed by the LIS is test result data generated by clinical analyzers. For this reason, the connection of automated instruments on-line to the LIS is perhaps the most important element of the system. Without available, functional, reliable instrument interface connections, the LIS would have little information to manage.

Fortunately, nearly all popular clinical instruments are capable of being interfaced to an LIS. The availability of a communication port on the analyzer is not, however, a guarantee of an easily established, affordable, or properly functioning interface to the LIS.

This paper addresses the clinical instrument interface process including methods, considerations and alternatives that laboratory personnel should be aware of when selecting, contracting for or developing an LIS.

Basic Considerations

An instrument interface is an automatic, electronic connection between analyzer and computer for the rapid, accurate exchange of information. A functional instrument interface may be viewed as a state of equilibrium across a number of influencing factors, as discussed below.

1. *Physical* (a) The instrument must be equipped with an active input/output port to which computer devices may be connected. On most contemporary clinical analyzers, this is a serial asynchronous RS-232 port. (b) The host system must have a corresponding I/O port available. (c) A connection cable is required, from the analyzer to an interface device or the host system connection point.

2. *Hardware* Many instruments include a personal computer (PC) or some other form of data management system (DMS) as an integral part of the analyzer. These DMS devices are typically used to provide data reduction, control of multiple instruments, reporting capability and/or QC data handling. In most cases in which some form of DMS is present, the interface connection is made to the DMS rather than to the analyzer itself.

Intermediate devices may be used in between the analyzer connection and the host system connection point, to provide distributed processing, data buffering, reformatting, networking or other control over interface communications. These devices include specialized interface workstations, PCs equipped with interface boards, user programmed PCs, black box interfaces, and simple buffer boxes.

Host systems are available on many hardware platforms, ranging from single user PC stations to large mainframes. Most contemporary LIS products are multi-user systems which run on minicomputers or PC networks.

3. Software The main categories of host system software are operating system software and application software. Operating system software controls basic machine functions such as interaction with I/O devices, memory management, disk access, and creating the application software environment. Operating system software is typically provided by a hardware manufacturer or a third party. Examples are MS-DOS and Windows (Microsoft Corp., Redmond, WA), PC-DOS (IBM Corporation, Armonk, NY), UNIX (AT&T Bell Laboratories, Murray Hill, NJ), VMS (Digital Equipment Company, Marlboro, MA) and OS/400 (IBM Corporation, Armonk, NY). Application software is the principal product of the LIS vendor, and provides all of the "user visible" LIS features. Application software may be written in a number of programming languages including C/C++, BASIC, MUMPS, COBOL, Fortran and Pascal.

The host system application software must include modules to control communication with the analyzer, according to the analyzer's format and protocol specifications. In applications in which an intermediate device is used, the intermediate device must contain software to similarly control analyzer communications. The host must then contain software to control communication with the intermediate device, which may be different than interacting directly with the analyzer.

The equilibrium of a running instrument interface can be disturbed by alterations to any of the above elements, from the obvious, of a broken connection cable, to the not so obvious, of a minor change to instrument output.

Clinical analyzer variables

There are currently many types and manufacturers of clinical analyzers available. The wide variety of popular clinical analyzers exhibit many operational, functional, and other differences. These instruments have evolved, driven by market forces that include new measurement technologies becoming available; improved productivity; changing regulatory and reimbursement environments; costs of acquisition; operation and maintenance; reliability; accuracy and precision; size; ease of use; reporting capabilities; and ability to connect to an information system.

Analyzers capable of being interfaced to an LIS are found in all areas of the clinical laboratory: chemistry, hematology, urinalysis, toxicology, immunoassay, coagulation, microbiology, blood gas, and special chemistry. Nearly all of the routine clinical instruments are equipped with a serial RS-232 I/O port. Many of the new point of care analyzers also provide some means of storing data for later upload to an LIS. Typically, instruments to be interfaced are prioritized by testing volume, placing chemistry and hematology instruments at the top of the list.

A primary problem in connecting clinical instrumentation to information systems is the lack of effective standardization of data formats and communication protocols produced by analyzers. Clinical instruments exhibit as many variances in their ability to communicate with an information system as they exhibit physically and operationally. This chaotic communication environment has resulted in substantial rewriting of host resident interface software with each new instrument, leading to increased cost to the consumer and longer development times, affecting availability.

Efforts to address the lack of instrument communication standards have taken several forms. A few analyzer manufacturers have attempted to provide a common interface format across several of their instrument models. Several independent companies have developed intermediate interface devices that reformat analyzer output into a more standard format across many instruments. More recently some industry organizations (ASTM, HL-7, IEEE, etc.) have led efforts to define communication standards for medical devices, including clinical analyzers. A more complete discussion of these standardization efforts is provided later in this paper.

In spite of sporadic efforts to standardize communication with clinical analyzers, most analyzers are quite different in their communication specifications from one to another. In fact, even the physical connection of a cable to the instrument often varies from one instrument to another. Over 30 different cable configurations are required to handle currently popular analyzers, due to physical differences in connector type, size, sex and PIN definitions.

A related problem to be overcome is the wide variety in quality of analyzer manufacturers' interface specification documents. An interface specification must contain three forms of information: 1) a definition of the physical connection point on the analyzer, to allow a connection cable to be built; 2) specifications of the communication protocol required to communicate with the analyzer; and 3) a clear description of the data format produced, including examples of data output in all modes of instrument operation. This must include specification of multiple formats if the instrument is capable of more than one. Various flag conditions and definitions must be included. Download format and protocol for bi-directional applications must be defined.

Highly desirable in the interface specification is a discussion of default switch settings, baud rates and other information necessary to establish a functional interface. It is helpful to know how to access interface related configuration screens or modes on the analyzer to check appropriate settings or make changes.

The expense of interfacing a number of analyzers has in some cases been higher than it should have been due to difficulty in obtaining the necessary information. Common problems found in analyzer interface specification documents include no specification of the connection port other than RS-232, reference to the data format in indirect terms (identical to that produced on the printer), incomplete specification of record output types, or simply inaccurate information (documentation does not match analyzer output). Some analyzer interface specifications have been massive documents, with many pages devoted to irrelevant information (defining the RS-232 standard at length), while declining to include important details (connector descriptions necessary to build a cable).

A last obstacle to mention is the lack of stability of analyzer interface specifications. The role of software in the operational control of a clinical instrument has been increasing with advances in microprocessors and related electronic technology. Instrument software controls internal operation of the analyzer, user interaction and communication with the LIS. Occasionally, instrument operating software is updated to fix bugs or add enhancements compared to earlier software versions. A regularly occurring circumstance in clinical laboratories is the installation of new instrument software that in some way modifies communication with the LIS. This sudden (usually unexpected) alteration in instrument output often results in interface failure, and may require modification of the LIS or other software to resolve.

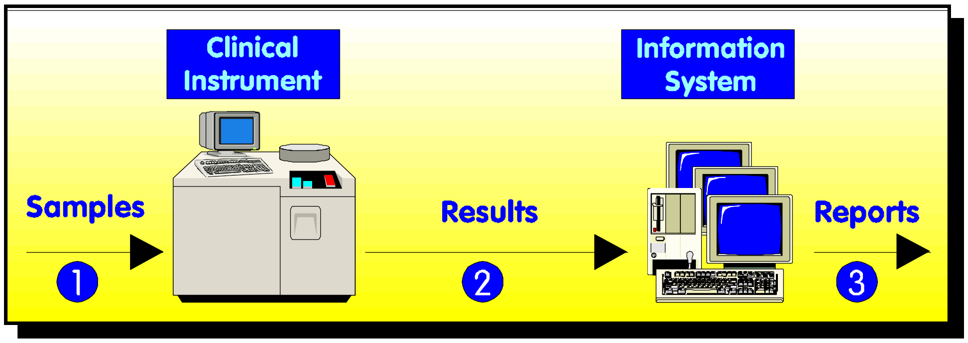

Figure 1: A Unidirectional Interface

These inconsistencies of analyzer output and other related characteristics have profound implications for the LIS, by forcing the LIS or other device connected to the analyzer to contain software specifically written to handle communications with that individual analyzer. With each analyzer requiring unique LIS or interface software, many programmers have been kept busy reinventing instrument interfaces. This situation contributes directly to the cost and availability of interface applications, ultimately borne by the end user.

Functional Types

Unidirectional applications

The traditional type of instrument interface is a unidirectional application: the instrument performs its test and transmits results to the interface device/host system in one direction only (upload). Many instruments have been limited by design to unidirectional interfaces. These analyzers are generally single test, batch or profile testing instruments in which the array of tests performed does not vary from specimen to specimen.

A unidirectional interface (see **Figure 1**) does not mean there is no communication at all coming down from the host or interface device to the analyzer. Most instruments require some form of handshaking (communication protocol) and/or error checking to obtain data records. Failure of the host or interface device to meet these protocol requirements will often cause the instrument to cease operation.

The data record transmitted (uploaded) from the analyzer generally includes the specimen ID, the array of test names, and results. Additional fields may include a variety of flag condition indicators, specimen type (blood, urine, STAT, control material), error check characters, and/or some demographic information among others.

A more unusual unidirectional application is one in which the communication is downloaded from the host only, with no test result upload. This situation occurs with certain autoloader devices, which accept a host download for control of loading specimens and reagents into microplates or other special containers. The specimen/reagent containers are then moved to a separate instrument for analysis.

Bi-directional applications

A bi-directional interface (see **Figure 2**) involves true two-way communication between the analyzer and interface/host. The host downloads specimen ID and test order information; the analyzer uploads specimen ID and test result information. Support of bi-directional interface capability is most generally found on random access testing instruments, which can perform a different array of tests on each successive specimen.

Historically, most random access testing instruments have been chemistry analyzers. More recently, bi-directional interface capability has been incorporated into many microbiology, immunoassay, coagulation and some hematology instruments. Not all of these instruments provide random access testing flexibility.

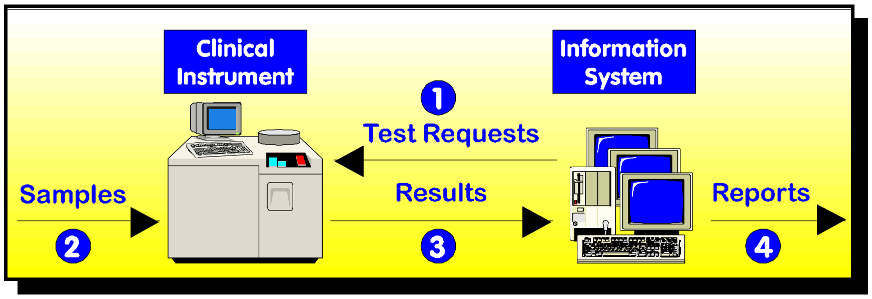

Figure 2: A Bi-Directional Interface

A bi-directional interface application saves the technologist the time to program test orders into the analyzer, and eliminates manual entry errors. This can result in a considerable improvement in analyzer productivity in a busy laboratory. Newer random access testing, bidirectionally interfaced analyzers will also incorporate bar code specimen label scanning which provides automatic positive specimen ID capability. This can eliminate manual entry of specimen IDs and/or coding of specimens by tray/cup position.

Some instruments support a bi-directional interface yet remain profile testing instruments. It is possible to download patient demographic information to these instruments, which may be useful if special reports are printed from the analyzer DMS. If reports are not being printed from these instruments, a bi-directional interface may be of limited use and considerable extra cost. Large-frame hematology analyzers fall into this category.

While most instruments support a bi-directional interface on one serial I/O port, there are several instrument models which use two ports for a bi-directional application (one port uploads, one downloads). This hardware requirement can double interface-related expenses (two interface devices, two host ports, two versions of host interface software, double cabling). Users should explore these analyzer requirements before determining a preferred method of interfacing.

Broadcast bi-directional applications

The trend in major chemistry analyzers in recent years has been to purchase multiple medium-volume instruments as opposed to one high volume instrument. These instruments then constitute an array across which the workload is distributed. Virtually all contemporary chemistry analyzers are random access testing devices that support bi-directional interfaces. A broadcast bi-directional provides a distributed download, where all of the download information is sent to all of the analyzers in the array.

The main advantages of the broadcast arrangement is that the technologist may load any specimen on any analyzer and be ensured that the proper download information is available (to process the specimen). This allows the workload to be more evenly distributed across the analyzer array, with adjustments made as specimen processing continues.

The main disadvantage of broadcast bi-directionals is that they create a download memory cleanup issue. If each analyzer received the total download information and only processes a fraction of the workload, some (perhaps the majority) of download information will represent specimens run on other analyzers in the array. This unused download information must be erased on each analyzer at the end of a shift or other logical stopping point. Otherwise, instrument memory becomes full, and the analyzer will likely halt LIS communications. The memory cleanup usually requires all the analyzers in the array to complete processing of all loaded specimens, transmit all result information back to the LIS, then stop for the cleanup procedure.

Download memory cleanup is a particular problem for busy laboratories, where the flow of specimens never really stops. This makes it difficult or impossible to identify a time when all analyzers may be interrupted long enough to perform the cleanup procedure. A separate problem sometimes encountered is that the analyzer may require a cumbersome procedure to do the memory cleanup, which can be frustrating.

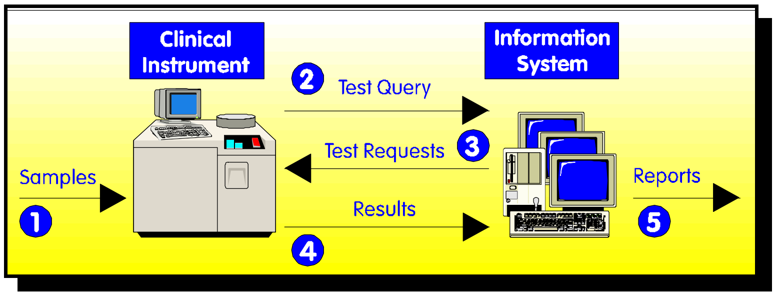

Figure 3: Query Bi-Directional Interface

Broadcast bi-directional applications can be simplified with the addition of intelligent intermediate interface devices (discussed in Part 2 of this article). An intermediate interface device can assist the host system by controlling the broadcast download, and networking the analyzer communications from multiple analyzers to a single port. In some cases, an intermediate device can automatically generate a blank download to other instruments in the array once an upload record has been received from one of the analyzers, solving the memory cleanup problem.

Query mode bi-directional applications

In a query mode bi-directional application (see **Figure 3**), the analyzer is loaded with bar code labeled specimens. As the analyzer processes a specimen, the bar code label is scanned and the specimen ID read. The analyzer then generates a query to the host, requesting download of the associated test order information. The host responds by looking up the specimen ID in its database, and generating the appropriate download. Query mode bi-directionals are generally an option on bar code scanner equipped analyzers.

Query mode bi-directional capability is generally perceived as the solution to the memory cleanup problems inherent to broadcast bi-directionals, although query mode interaction was implemented by some clinical analyzers before those problems were well known. A query mode interface ensures that each analyzer will only receive download information for the specimens actually processed at that station, even in cases where multiple similar instruments are involved. Memory cleanup of excess downloads is not an issue, and any specimen can still be loaded onto any analyzer.

While solving the memory cleanup concern, query mode bi-directional applications have created other problems, mostly related to communication timing. Query mode analyzers allow a specific time window following transmission of the query to the host system within which the host must respond with the appropriate download. In some cases, this time window is as short as 6 seconds. If the host does not respond within the specified time, the analyzer indicates a host communication failure and often stops.

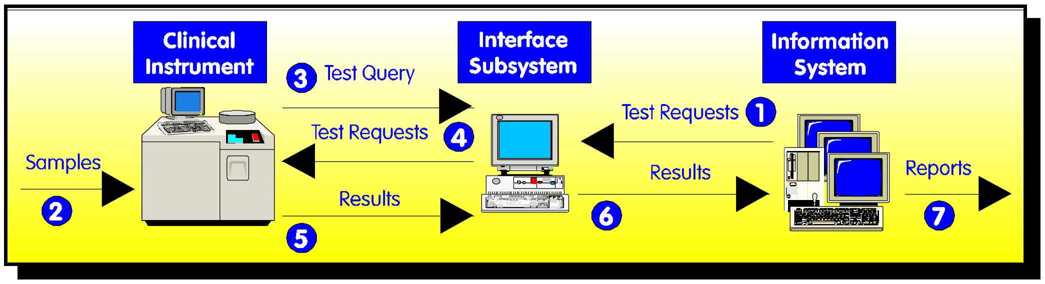

Figure 4: Distributed Processing / Query Conversion

Most host systems cannot always guarantee a response within 6 seconds. Some more recent analyzers have recognized the impact of communication timing restrictions on interface performance, and have allowed longer time windows for host responses, in some cases up to 2 min. However, lengthening the host response time window also has the effect of reducing instrument throughput.

The ability to support query mode bi-directional applications is significantly enhanced with the addition of some distributed processing at the interface level, by using a PC-based interface or other intelligent interface device (see **Figure 4**). The host system can download to the interface device in advance of instrument operation and store the download information in buffer memory. Then, when the instrument generates the query message, the interface device is able to respond quickly with the download, always meeting timing constraints.

In multiple instrument situations, it may be possible for the host to broadcast download to an array of interface devices, which in turn support query mode interaction with the analyzers. Memory cleanup of excess downloads at the interface level may sometimes occur automatically, via an intelligent intermediate interface device. This type of application provides the operational flexibility of query mode interfaces with the simpler host system demands of broadcast downloading.

Query mode bi-directional interfaces are supported by BMD/ Hitachi chemistry instruments (Boehringer Mannheim Diagnostics, Indianapolis, IN), the Beckman CX series and Array (Beckman Instruments, Inc., Fullerton, CA), the CIBA Corning ACS 180 (CIBA Corning Diagnostics, East Walpole, MA), the MLA 1000C (Medical Laboratory Automation, Pleasantville, NY) and the Becton Dickinson Bactec 860 Autoloader system (Becton Dickinson, Sparks, MD) among others.

Interface architecture

Instrument interfaces may be established via host software only or a variety of ways using some intermediate hardware and software between instrument and host. A discussion of each of these methods follows.

Straight-in, software-only interfaces

Straight-in, software-only interface connections are perhaps the most common type of clinical instrument connection found today. The physical connection is made via a cable straight from the analyzer to the host, with no intermediate devices. This approach requires an instrument specific communication (interface) program running on the host system.

The advantage of straight-is that no third-party hardware/software is required. Vendor programmers are already on staff and are considered part of the company overhead, therefore interface development may incur no external cost to the vendor. Some vendors trust only their own software. Straight-in interfaces allow the vendor to be in complete control, and not be dependent on any external devices.

The disadvantage of straight-in interfaces is their limited flexibility. Interface programs are very specific and inflexible. Any change to the analyzers' internal software that modifies its output requires a change to the host interface software. Upgrading to newer (replacement) instrumentation requires a completely new host interface program. Availability of new host interface software can be slow and costs to the user can be high.

Straight-in interfaces may affect overall system performance. Clinical instruments can place significant real-time demand on the host, particularly those that require a complex communication protocol or are high volume instruments. The demand on the host CPU (central processing unit) increases with the number of instruments connected; even small laboratories with only a few on-line instruments may notice the effect. Peak workload periods may produce a frustrating degradation in system performance.

An important limitation of straight-in interface connections is the lack of data buffering. When the host system is down for routine backup or any unscheduled reason, communication with the analyzer array is severed. Any patient data generated during this period must be entered into the system manually at a later time. Since most laboratories are 24-hr operations and specimen processing must proceed whether the LIS is up or down, the manual entry catch-up task can be substantial if the downtime occurs during a busy period.

There is a relatively constant flow of new interface requirements as new instruments become available, old instruments wear out, and the internal software of existing instruments is modified by the manufacturers. Even small LIS vendors using straight-in interfaces have a staff constantly busy with interface development. This can distract from development of core system features.

Examples of some vendors routinely using the straight-in interface approach are Meditech (Medical Information Technology, Inc., Westwood, MA), Cerner Corporation (Kansas City, MO) and Citation Computer Systems, Inc. (Maryland Heights, MO).

Black box interfaces

Many of the instrument interfaces in early laboratory systems were handled via black box devices between the analyzer and LIS. Black box interface connections were more common several years ago when the need for signal conversion (analog to digital, parallel to serial) from analyzers was a more frequent requirement. Black box interfaces were the first intermediate devices to provide data buffering, specimen ID entry and limited reformatting. They were also the first devices to provide a distributed processing approach to instrument interfaces. Black box interfaces specifically for clinical instruments were manufactured by Creative Computer Applications (CCA) (Calabassas, CA), Edmac (Fishers, NY) and Biovation, Inc. (Richmond, CA). Only CCA remains in the interface business.

Black box interfaces provided several advantages. Their signal conversion capabilities were required to interface some old but very popular analyzers, including the Technicon SMA series (Miles Inc., Diagnostics Div., Tarrytown, NY) and Coulter S and S/Sr. (Coulter Corp., Miami, FL) instruments. Several models of black box interfaces provided some data buffering between analyzer and host, allowing the instruments to be operated during periods of host downtime. Later black box variants provided a keyboard for entry of specimen IDs, providing some positive specimen ID for non-keyboard equipped analyzers.

Another feature introduced by black box interfaces was the automatic attachment of a short header to the analyzer data record format. This header was identical across analyzers, and included specimen ID, instrument identifier and other related information. The concept was that the LIS would recognize the header as instrument data and more easily process the record. Black box devices do standardize host system cable connections and communication protocol requirements, which help simplify instrument connections.

The disadvantages of black box interfaces are numerous. The major disadvantage is that while these products do provide the system programmers with some assistance by standardizing cabling and communication protocols and adding a standardized record header, most analyzers still require instrument specific interface software on the LIS. Communication software within the black box is hard coded in firmware, requiring interface boxes to be ordered by specific analyzer, preventing the application from being changed easily. Most analyzers now have keyboards or bar code scanners, eliminating the need for external specimen ID entry. Data buffering capacity is quite limited by contemporary standards, often limited to several hundred records or less.

Black box interfaces are relatively expensive (in the $ 1500 - $ 4000 range) and have limited bi-directional capability. These units incorporate only small LED displays, which prevents users from being sure of interface status during operation. Black box interfaces often become disposable when instrumentation is replaced, because they are hard coded for individual instrument models. All of these factors contribute to the uneasiness of some LIS vendors to incorporate such devices into their systems.

The major contribution of black box interface devices was to demonstrate the usefulness of data buffering and distributed processing. This approach to instrument interfaces has been used more recently by First Data Corporation, Health Systems Group (Saint systems) (Charlotte, NC), SmithKline Beecham Clinical Laboratories (King of Prussia, PA), Hospital Corporation of America (Nashville, TN) laboratories, and by Meditech for certain instrument applications.

PC-based software interfaces

The rapidly decreasing cost and increasing performance of microcomputers in recent years has made the PC based instrument interface more attractive both for LIS vendors and home grown systems. This approach uses a PC to interact directly with the analyzer, and will typically provide data buffering and possibly some reformatting before transmitting information to the host system.

PC based software interfaces have some advantages over black box devices and straight-in interfaces. The PC provides a distributed processing architecture, decreasing processing demands on the host system CPU and generally allowing the system to run faster. A PC with a hard drive can provide substantial data buffering for clinical analyzers, allowing operation during periods of host system downtime. It may be easier to develop the instrument specific software at the PC front end than directly on the host.

More elegant implementations of PC based interfaces may add such features as substantial reformatting of data records, range checking, terminal emulation, access to related data. Such devices are discussed in the next section.

The disadvantages of PC software interfaces are related to the software base of the approach. Most PC software interfaces will support only one analyzer, due to the limited processing power of the PC, and the complexity of multi-user software development. Flexibility of the PC software interface is quite limited, and controlled by the LIS vendor or system programmer. Adaptation to new laboratory instrumentation requires reprogramming, sometimes a considerable task. The ability to reformat analyzer data prior to transmission to the host system may be limited, leaving the need for instrument specific interface programs on the host as well.

Most PC software interfaces were developed primarily to provide data buffering in front of the LIS. Some PC interface implementations accept data input from several analyzers, but only buffer the information to the PC's main memory and not to hard drive. In these situations the buffer capacity may not be adequate to handle peak workload throughput without significant slowdown. In the worst case this may require shutdown of several analyzers to maintain production of others.

The cost of these units may be higher than straight-in interfaces, especially in cases where a unique host resident interface is also required. The cost and availability of new instrument applications are dependent on the background of the vendor/supplier and their profit motive with respect to interfaces.

The PC software based interface approach is used by Advanced Laboratory Systems (Eugene, OR), Antrim Corp. (Plano, TX), Community Health Computing (CHC) (Houston, TX), 3M Health Information Systems (MedLab) (Salt Lake City, UT), and New Laboratory Force Company, Inc. (Dallas, TX) among others.

Intelligent intermediate devices

A further approach to instrument interfacing uses a combination of hardware and software between the analyzer and host, producing an intelligent intermediate device. These devices may be PCs with sophisticated software applications, PCs equipped with specialized interface expansion cards, stand alone CRT (cathode ray tube) workstations that provide multiple interface related features, or other devices including minicomputer-based front end systems. Common among these devices is the addition of some processing power in front of the host system typically used to provide substantial reformatting of instrument data, buffering, and other features without placing a burden on the host system.

These devices require an instrument specific software program running at the interface level to control interaction with the analyzer. Data communication to the LIS is generally translated into some standardized format, allowing the host system to use a single interface program to control interaction with all peripheral instrument interfaces. Cost and availability of these devices vary with vendor/supplier.
An intelligent interface device offers a number of advantages to the user: Data buffering capacity is significant, allowing continuous operation of analyzers without regard to system status. Manual data entry following system downtime for any reason is eliminated. Transmission of data to the host LIS in a uniform format from all analyzers greatly simplifies host software development and maintenance relative to instruments. New analyzers can be assimilated into the system much more quickly and easily with minimal impact on the host system software. Likewise for modifications to analyzers already in place.
Conversion of all analyzer data into a uniform format makes networking a large number of dissimilar instruments to a reduced number of host ports much more feasible. Most of these intelligent intermediate device interface solutions provide both standardized formats to the LIS and networking to a single port.

The advantage of distributed processing in handling more complicated interface applications such as query mode bi-directionals is becoming well recognized. Many host systems cannot provide the necessary response to an analyzer query within required time constraints. An intelligent intermediate device may allow the host to batch download information ahead of analyzer queries, and always meet analyzer timing specifications.

This approach may provide a higher level of vendor/user control, allowing some flexibility as to how an instrument is operated and/or control over the data stream sent to the LIS. Some applications may provide more advanced features such as result review/editing, use as a host system terminal, printer control, networking of multiple analyzers to a single host port, data archive/retrieval, and user programmability. These devices may also have cost advantages over alternative methods of interfacing, due to simplification of host system software demands. If designed as open interface systems, they may be configurable by the user or vendor to more easily meet new or different applications.

There are some advantages to an intelligent interface solution provided by a single third party source common to many LIS products. One supplier can provide all the analyzer related programming in a uniform fashion, saving the LIS vendors duplication of this effort in many independent directions. The ability of the LIS to receive data from all analyzers in a standardized format can preserve vendor programming resources for development of other important system features.

The disadvantages of intelligent intermediate devices include an understandable concern that additional vendors and devices may unduly complicate the interface process. The cost of minicomputer-based solutions may be high due to the more expensive hardware and software involved, but may also provide a sophisticated array of features.

The intelligent intermediate interface device approach is provided by interface products from Dawning Technologies (Fairport, NY) (PC compatible expansion cards, workstations and networking software), and by LIS vendors such as Sunquest Information Systems, Inc. (Tucson, AZ) (minicomputer based interfaces as a component of the complete LIS).

LIS vendors incorporating Dawning interfaces include Ameritech - Knowledge Data (Larkspur, CA), Clinical Information Systems (Lake Oswego, OR), Collaborative Medical Systems (Waltham, MA), The Compucare Company (Reston, VA), Custom Software Systems (Nashville, TN), First Data Corporation (Charlotte, NC), HBO & Company (Atlanta, GA), Intermountain Healthcare (Salt Lake City, UT), Keane Health Sciences Division (Melville, NY), Northern Software (Ironwood, WI), Sentient Systems (Rockville, MD), System Analysis Corporation (Wellesley, MA), the Terrano Corporation (Lincoln, NE) and others.

Other intermediate devices

Several other types of intermediate devices may be found between analyzers and laboratory information systems in some circumstances. Protocol converters are used to connect serial devices (analyzers) to certain host hardware systems. These are generally required in IBM mini/mainframe base systems (IBM Corporation, Armonk, NY). Modems are often used with analyzer - host connections, most typically when the cable distances are long. A variety of networking hardware and software products are available, which allow multiple analyzers to be connected to a reduced number of host I/O ports. Networking is common, as it saves cabling and port related expenses.

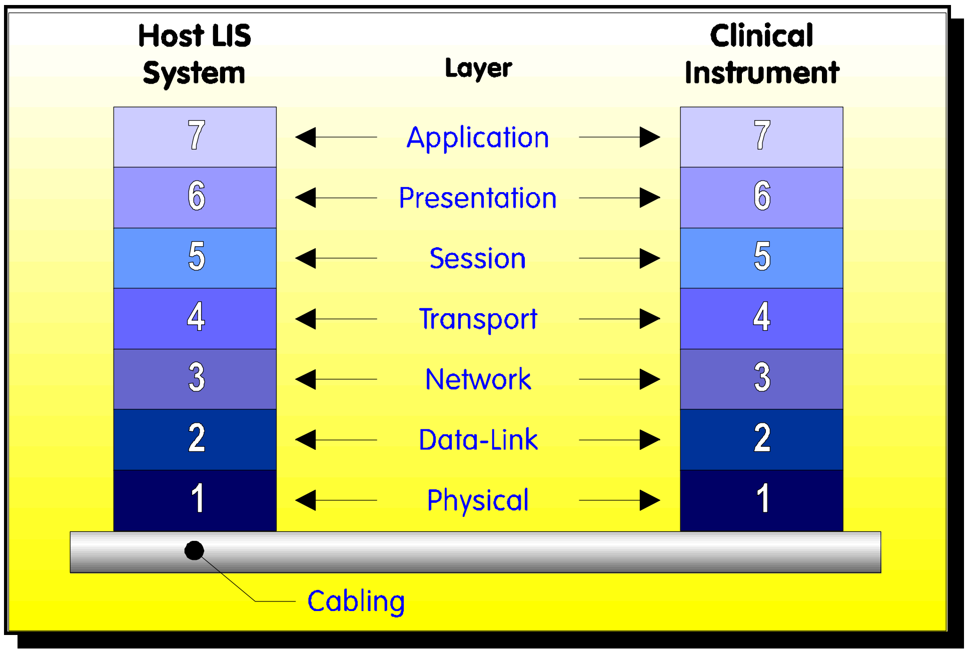

Figure 5: ISO Seven layer reference model

an LIS is actually present and initiates the communications.

Continuing the process, the Session Layer having determined that an LIS connection is present hands the formatted result data to the Transport Layer. Here the exchange between the devices is managed using a defined protocol that is responsible for reliably passing the information to the LIS. The data transfer occurs at this point. Each individual data character whether this is part of the protocol or actual data, is passed down next to the Network Layer.

Inside the Interface

The International Organization for Standardization (ISO) in the mid-1980s proposed the Standard Reference Model of Open Systems Interconnection ^{2}. This was to provide a common basis for the coordination of standards development but also it allows us to place existing standards and techniques in perspective by identifying the functional elements of the interconnection. According to this model, the interface or connection between any two devices, whether these devices are instruments, computer systems or intermediate interface components, can be envisioned as having a layered architecture.

The ISO model (see **Figure 5**) defines seven functional layers ranging from the most basic, involving the actual physical interconnecting wiring, to the top level application: information management in our case. Each layer has its own responsibilities with regards to the interface. Layers 1-6 provide for a step-by-step communication process. For an example let us consider a clinical instrument that completes a set of tests and decides that this information is to be sent to an LIS host system. This decision is made at the Application layer, essentially a block of specific program code, within the instrument. Within the instrument the raw result information is made available to a separate section of programming (the Presentation Layer) which formats the information in a fashion that can later be understood by the LIS. The Presentation Layer then hands this formatted information down to the Session Layer which determines if

In the current environment, for the most part, the Network Layer is trivial. Clinical instruments are typically connected directly to another communications device and not through any form of network. In the future networking might become more prevalent. If a network did exist, it would be the responsibility of the Network Layer to insure that data is passed to the proper destination over the net. This is physically forced with a point-to-point RS-232 connection.

In the absence of a true network, the data is merely passed through the Network Layer to the Data-link layer which in the clinical laboratory is the RS-232 world. Finally the Physical Layer represents the hardware and cabling needed to establish the connection.

Once the information flows through the Physical Layer to the LIS, the process proceeds in reverse until the raw test data is successfully presented to the LIS Application Layer. In the LIS system, the Data-link layer receives data from the instrument, and presents it to the Network Layer. The Network Layer here might have a somewhat better defined purpose in that the LIS does receive information from more than one instrument. The Network Layer in this case might simply identify for the LIS the

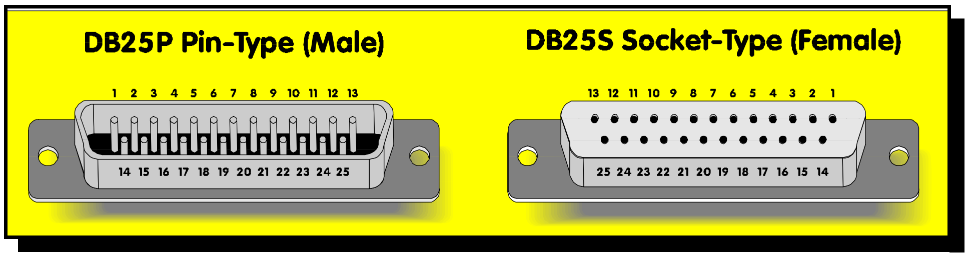

Figure 6: RS-232 D-Subminiature Connectors

source of the current data. The Transport layer collects the entire transmission and hands the block of formatted results to the Presentation Layer. Here the formatting is parsed to extract the raw test results and other information needed by the overall application, the laboratory information management.

This layered model assists the software designer in structuring the device programming. The actual implementation might not contain separate modules or routines that relate directly to each independent layer. If the programming can closely approximate the ISO Reference Model, it might be more easily upgraded in the future when, for instance, a Local Area Network (LAN) is substituted for the RS-232 world.

We will proceed here to take a closer look at each of these layers. Our goal will be to provide enough description to help the LIS manager to develop an understanding for the functionality and also the potential trouble areas. This will better define the kind of support an interface may require both at the initial installation and later as operational situations change.

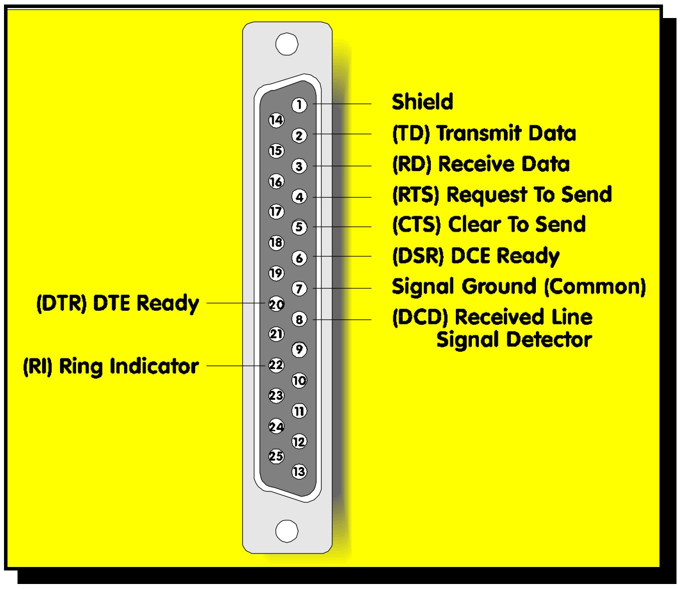

Figure 7: DTE RS-232 Pin Assignments (IBM PC)

The Physical Layer: Making the Connection

The EIA RS-232 standard [3] for asynchronous serial data communications has been in use for over 30 years. It defines a connection between Data Terminal Equipment (DTE) and Data Circuit-terminating Equipment (DCE) or Data Communications Equipment (DCE) as the latter is sometimes called. The two meanings for DCE are used interchangeably by the standard's authors in many application notes [4]. This standard allows manufacturers to develop and market the compatible equipment necessary for the remote connection of computers and their peripheral devices. A modem is an example of such DCE communications equipment while the computers and many peripherals fall under the DTE category.

One key and troublesome difference between DCE and DTE devices under the RS-232 standard is the wiring of the cable connector. The original DCE and DTE definitions permitted these two types of equipment to be interconnected with a simple 25-wire cable wired straight through, that is pin-1 to pin-1, pin-2 to pin-2 and so on. A 25-pin D-subminiature mating connector set (see **Figures 6 and 7**) was specified where a pin-type (male) connector is defined for DTE devices with data transmission occurring using pin-2. For a straight through connection, a corresponding socket-type (female) connector was to be used for DCE devices with the data being received on pin-2.

The simplicity of this design soon got lost amongst attempts to simplify the interconnection of DTE devices directly to other DTE equipment. In this case some manufacturers chose to alternately configure their products as DCE or to provide a method for switching a device from DTE to DCE internally. Some manufactures required the use of a crossover cable where, internal to the cable, pin 2 and 3 were exchanged. This being necessary since both DTE devices would be transmitting on pin-2 and before the signal got to the other end of the cable it would need to be on pin-3. The term "crossover" comes from the appearance of the cable connections when drawn in schematic form.

Matters were further complicated as DTE devices which then could be switched from DTE to DCE could only alter the wiring definition (pin assignments) but not the connector hardware. Devices with the incorrect style (sex) of connector would then be common. Some manufacturers incorrectly chose to use the socket-type connectors regardless of device category as these connector are less likely to receive an electrostatic discharge during handling, an event which could damage or degrade the equipment. With the advent of the personal computer (PC) a 9-pin connector arrangement for an RS-232 connection was introduced most likely to overcome space constraints (see **Figure 8**).

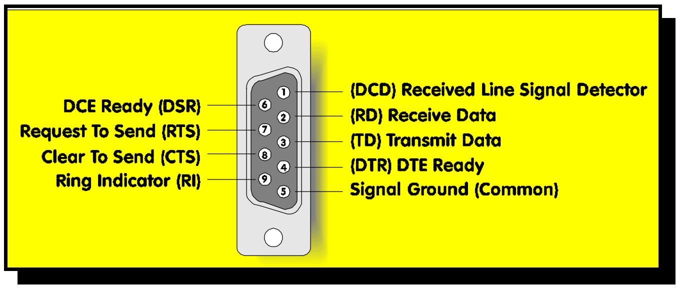

Figure 8: IBM/AT Style RS-232 Pin Assignments

With the resulting confusion over pin wiring and connector configuration, there is no substitute for the availability of a good cabling diagram for each and every connection. Constructing the proper cabling for an interface can be a challenge. It is no surprise that the majority of initial interfacing difficulties are cabling related. Ohm Meters and Break-Out Boxes can be an indispensable tools for debugging and constructing cables.

TABLE I: RS-232 Signal Interpretation

Notation	Interchange Voltage	
	Negative	Positive
Binary State	1	0
Signal Condition	Marking	Spacing
Function	OFF	ON

It is important to identify the physical location of the RS-232 connector on any instrument to be interfaced, the connector type (9 or 25 pin, pin-type or socket-type), and actual pin function definitions prior to acquiring an interface. The RS-232 connection might be an option on the instrument which may require additional time to obtain. The presence of a connector, even if appropriately labeled, does not guarantee that the internal circuitry is present or functional. It is good practice to have the instrument manufacturers service technician verify the operation of your instrument's RS-232 port on a routine service visit.

The Data-Link Layer: Moving the Information

All of the clinical instrumentation commonly located in the laboratory present their data in ASCII character form. This is to help simplify the interface development process as the formatted results can be easily printed or viewed on a terminal or data scope. Interface formats that represent numeric values in binary form, while generally being more efficient or carrying additional precision, require a different and less available level of programming expertise. Such formats are more common in research environments.

The American Standard Code for Information Interchange (ASCII) defines a relationship between the characters (and punctuation) of the printed page and the binary codes suitable for computer transmission and storage. The letters and numbers on your computer screen each have a unique numerical value assigned by the ASCII standard. While there have been extensions to the standard ASCII character set, the basic lower and upper case alphabet, the numbers and the normal set of punctuation all have consistent values which can be represented with a 7-bit word. The upper case letter 'A', for instance, has the assigned value of 65 (decimal). This value might be presented in hexadecimal as 0x41 or 41h, or in binary as 1000001. However it is represented, the numeric value remains consistent. An ASCII table defining the characters and their values is often found in the appendices of computer documentation.

Extensions to the ASCII code permit the use of control codes, that can define the arrangement of text on your screen for instance, accented characters for foreign languages and graphical elements such as boxes and lines. An extended set of ASCII codes assigns characters, symbols and codes to the values from 0 to 255. These 256 codes represent all possible combinations of bit values (0 or 1) that make-up a computer byte, a total of 8 bits.

In an RS-232 transmission the bits of these character data are marched along the wires serially, one right after another. To do this, the transmit data line (TD) assumes one of two states representing either a binary zero (0) or a one (1). An in-depth and technical understanding of RS-232 is not generally necessary to be able to develop a functional interface. This can be helpful, however, in understanding the need for quality workmanship in cable construction or in diagnosing cabling difficulties.

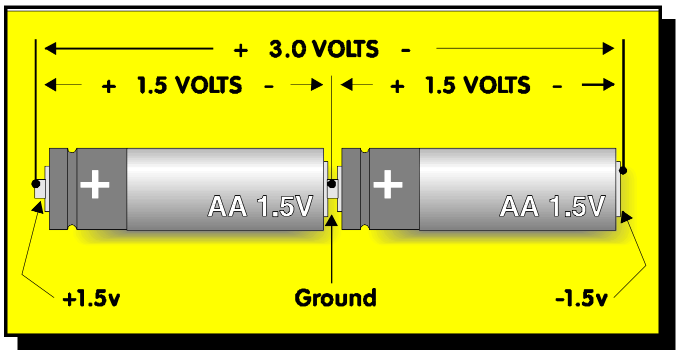

Figure 9: Positive and Negative Voltage Example

The voltage of the Transmit Data (TD) line relative to the signal ground at any time defines its state. A voltage is always defined relative to another point or connection. You read on the label, for instance, that a AA battery provides 1.5 volts (at least until it dies). This means that the '+' positive end of the AA cell is at 1.5 volts relative the '-' negative end. Stack two AA cells end-to-end and the total voltage is 3.0 volts (see **Figure 9**). If we define the center point where the two batteries meet as a reference or *ground* point, then we say that the '+' positive end of the arrangement is at a +1.5 volts relative to ground and the '-' negative end is at -1.5 volts relative to the ground.

Unlike a AA battery, the voltages of the RS-232 signals

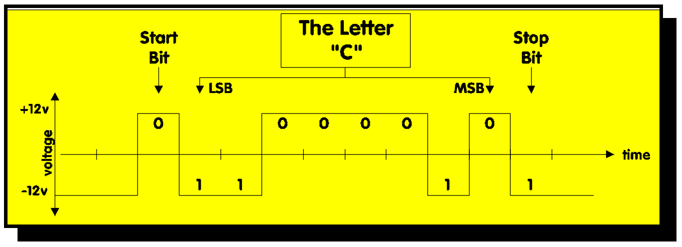

Figure 10: Typical Form of Character Transmission

change rapidly from positive to negative and back. The RS-232 signal voltages are defined relative to the Signal Ground (pin 7 on 25-pin connectors). For RS-232, any voltage exceeding a positive 3 volts is considered to represent the binary state 0. This is sometimes referred to as the *spacing* signal condition. A negative voltage below -3.0 volts relative to signal ground represents a binary 1 or *marking* condition.

Most manufacturers will use +12 volts and -12 volts for the two states respectively (see **TABLE I**). This insures that the 3.0 volt levels are easily exceeded so that reliable communications can be achieved even when some voltage is lost through long cabling, bad connections or otherwise poor electrical environments. Clean and secure connections in cabling are important. *Shielded* cables are necessary to prevent the influences of external electrical interference such as that caused by laboratory refrigeration units, sample tube rockers, fluorescent lighting, or other heavy duty electrical devices.

Any external event that can influence the signal voltages enough to have one of them fall into the -3.0 to +3.0 volt range for even the briefest moment, can (and will very likely) cause a communications failure. This could mean permanent electrical damage but is much more likely to subtly alter an ASCII code resulting in the reception of character different from what has been sent. The resulting transmission error could completely alter the meaning of the transmission. Later, we will mention some of the ways in which these errors are detected and possibly corrected or otherwise handled appropriately.

The RS-232 Transmit Data line (TD) takes on the value of a bit for a duration specified by the Baud Rate (see **Figure 10**). It then is switched to the next bit for an equal length of time and so on. A character's bits are sent starting from the least significant to the most. Each character is preceded by a start bit (always a marking 1) and is followed by at least one stop bit (always a spacing 0). In this way the time between characters can vary greatly and data can be transmitted in an asynchronous fashion with no constraints on the exact timing between characters.

Since the possibility exists that external electronic disturbances can corrupt the bit stream, an additional bit called a parity bit can accompany each character. The parity bit is calculated based upon the value of all of the other bits in the character. Typically the parity bit is a marking 1 or spacing 0 as would be necessary to force the number of marking 1 bits in the current character to be odd or even (defined by the setup). If a transmission error occurs, it is likely that the parity bit would be incorrect (an odd number of bits results when there always should be an even number for instance). This then indicates that an error has occurred and that the data character may not be valid. Parity is just one of the many methods used to handle transmission errors. By itself it is not sufficient to guarantee the detection of all errors. Other more sophisticated techniques are needed and are implemented by the Transport Layer protocols.

Fortunately, the electrical part of the RS-232 exchange is handled in a standard and consistent manner by the individual hardware components in the design. Generally, some parameters controlling the operation of the RS-232 connection can be set by the user and must be configured properly. The user is responsible for defining the bit rate (baud), the number of data bits (7 or 8), the number of stop bits (typically 1) and the use of parity (odd, even, 0, 1 or none). Communications are only successful when both the transmitter and receiver have been configured with the same baud, bits, stop and parity settings.

In the laboratory, it is important that the LIS manager know how to set these parameters on all of interfaced instruments and on the corresponding system or intermediate interface device. A log of the proper settings should be maintained to facilitate the resetting these parameters should they be altered.

The Network Layer: Identifying the Source

As discussed previously, most instruments are connected point-to-point. In the absence of a network connection, when data is received we know precisely where it came from just from the physical attachment. Within the instrument, the Network Layer is trivial. In the LIS, however, the Network Layer can insure that some identification as to the data source is provided. Many labs have two or more instruments that can perform the same determinations, which may or may not use the same test methodology. A glucose value, for example, might be obtained from more than one analyzer. A provision for assigning values from differing instruments special test names or otherwise identifying their source, can be important should it later be determined that a particular instrument's results are faulty, or for routine tracking of quality control/assurance (QC/QA) information.

The Transport Layer: Understanding the Flow

The RS-232 connection permits the flow of character data in both directions. Characters can pass from the instrument to the host computer and, similarly, from the host computer to the instrument. The maximum rate at which these data characters can be moved from one place to another is controlled by the *baud rate*. The baud rate establishes the fixed rate at which bits can be transmitted.

Ten bits (each either a marking 1 or spacing 0) are required to pass a single character using the typical parameter settings for 8 data bits, 1 stop bit and no parity. The start bit accounts for the tenth bit: 1 start bit, 8 data bits, and 1 stop bit. For an example, consider a baud rate setting of 9600 baud. This provides a maximum transmission rate of 9600 bits per second or 960 characters per second using the typical 10 bits per transmission. This amounts to only about one half of a screen full of ASCII information (12 lines by 80 characters) assuming the transmission of all positions including blanks or spaces (ASCII 32).

The 960 characters per second represents a maximum rate assuming that data is transmitted without pause. The mere asynchronous aspect of the RS-232 connection assumes the existence of some delay between characters. This could be as little as a single bit time or much longer. These delays are often required by the transmitting device which typically builds the transmitted message retrieving information from storage devices such as disk drives. This retrieval time often results in transmission pauses. As a result, at 9600 baud, the throughput rarely reaches the 960 character rate.

Since typical baud rates range from 1200 to 19,200 baud, 9600 is quite common. Yet at this rate it would take two (2) seconds to transmit an entire screen full of information. This would be unacceptable if it weren't for some of the available techniques for enhancing communication speeds. In the case of the screen display, blanks are normally not transmitted. Some protocols incorporate data compression. This is all geared towards reducing the transmission length (overhead) and thus increasing the throughput for any fixed baud rate.

An RS-232 connection can then present a bottleneck for the transmission of large amounts of data. It is clear that the throughput limitation depends on the overall size of the transmitted messages. This will be discussed in more detail in a later section. It is important to note that the throughput is not only limited by the transmitter. The system receiving an RS-232 message must be able to do so at a rate equal to or faster than the transmitter or data will be lost.

Buffering

The receiving system must process the incoming data, extracting numeric and textual data, ultimately filing this newly obtained information into growing data bases. This rarely can be accomplished fast enough to keep up with the transmitting device. The receiver then cannot get back to collect incoming data characters in time to insure no loss.

To prevent data loss, the receiver typically collects the entire message before beginning its processing. The incoming data is then analyzed and filed away during times when the transmitter is quiet. The data is collected into a buffer which usually has a fixed size and defined large enough to hold the biggest message expected. If the buffer is too small, a large transmission will overflow the buffer and data will be lost. Data buffers are critical to the reliable performance of the serial link. Buffers are used for both receiving and transmitting functions.

Handshaking

A data buffer alone cannot completely guarantee zero data loss. It may not be practical to allocate a buffer larger than the maximum message length or this maximum length might not be known. The possibility for buffer overflow therefore always exists. To prevent overflow the receiver is provided with a means by which it can stop the transmitter. Both a hardware and software technique exists for this.

The original hardware handshake scheme for RS-232 was devised for use with early modems which could process information in only one direction at a time. A DTE device having data to send would assert the Request to Send (RTS) signal to the connected modem. Once the modem was ready to take data from the DTE device it would respond with the Clear to Send (CTS) signal. Upon seeing CTS the data characters are transmitted. Finally the RTS signal is removed, the modem cancels the CTS signal and the DTE device can then receive a possible response.

In situations where two DTE devices are interconnected, equipment manufactures have taken some liberty in the application of the hardware handshake. In many cases the RTS of one device is connected to the CTS of the other and vice versa. The result is the ability of either to stop the flow of data from the other. The RTS lines in this situation are used to control the flow of incoming data rather than as an indication that there is data to be sent. As a result, the techniques for hardware handshaking may differ from device to device rendering their interconnection incompatible. With the confusion over the implementation of the hardware handshake, a software technique has become more popular.

The XON/XOFF software handshake defines two special control characters. To stop incoming data the receiver sends an XOFF (ASCII 19) to the transmitter. Transmission is restarted after the receiver is able to free up its buffer space by sending the XON (ASCII 17) character. Some implementations will restart transmission with any character following the XOFF. This mode of handshaking requires that the XON and XOFF characters not appear in the normal message transmissions. This might become a problem when binary data formats or error checking characters are included. This will be discussed in a subsequent section.

The Transport Layer: Getting the Message Across

The RS-232 standard provides us with the ability to move characters and other forms of data from one device to another. At this level there are capabilities to control data flow and to perform error checking. Neither of these techniques are robust enough by themselves to insure complete flexibility for flow control or completely reliable error detection.

A more enhanced form of handshaking can be implemented as part of a complete protocol. In these cases a set of control characters are defined which permit the receiver to request a transmission and then acknowledge the successful reception. The protocol might include forms of error checking permitting the detection of faulty transmissions. When transmission errors are encountered another control character can cause the retransmission of the message.

These protocols can range for the very simple to the very complex depending on the features provided. Each is designed to successfully move a block of character data containing the formatted result data or other information from one point to another. This is done without regard to the specific format of the block or *message*.

So you've connected an instrument to your LIS with the proper cable, you've set all of the proper communications parameters and there is data in the instrument ready to go, but nothing is happening. You've double checked the settings and even tested the hardware lines using a break-out box. Still, there is no communications activity. This is a typical situation and the problem is likely in the protocol.

According to an old Webster dictionary (circa 1957), protocol is defined, in this sense, as "the ceremonial forms and courtesies that are established as proper and correct in official intercourse between heads of states and their ministers". This description must have been updated certainly in the past 40 years, but even then it encompassed the basic purpose that still has an appropriate meaning. There must be an established, proper and correct procedure for communication. At the software level there still remains an element of ceremony. The old definition implies an exchange between entities of differing levels of authority. Call it what you will, but these terms come to mind: speaker/listener, master/slave, or client/server. Indeed we all might know how protocol can easily fall by the wayside when there is any disagreement regarding the roles of the participants.

The reference to government in the original definition of protocol is not far off-base as politics certainly has come into play. Sit a number of clinical analyzer manufacturers and information system vendors down at a table. Give then the task of defining a standardized protocol. A master/slave protocol of some kind might be the simplest and a good place to start, however, it is doomed from the start. There is certainly no chance of gaining any consensus as to whom would be the "master" and whom would be relinquished to "slave". This was the scene some years ago during the initial meetings of the E31.14 ASTM laboratory Interface Standards group.

In many instruments, sample testing is synchronized to mechanical and fluid processes. There is a specific time allotted for each test and this can lead to timing constraints on the information system interface. This would not be of concern with the instrument as master and the information system being responsible for always paying attention and meeting the timing. On the other hand, information systems serve large numbers of users, multiple input and output devices, process many varying tasks, and have little or no time to focus on any one job for very long. Not a problem since as master the system can proceed along as demands dictate and later gather information from the instruments when time allows. The political solution is to consider all devices as being created equal.

Another type of protocol then has a better chance of acceptance. This one serves the "peer-to-peer" interface. With no single device being designated as the master and no third party moderator, either device can begin a data exchange at will. Now the situation occurs when both begin to talk and neither listens. The protocol must necessarily become complex here in that both talkers must back-off a random period of time and retry the transaction. This procedure continues until one or the other device successfully gains control. It is temporarily the master. A typical master/slave handshake can then follow but the negotiation process has added some difficulty to the implementation. This comes at some additional cost in transaction setup time and protocol overhead. The exchanges are less efficient.

Now what if one device achieves master status in this way and then decides to keep it for as long as it wants? "That's not fair!" comes the cry from one side of the table. So into the protocol goes functions to interrupt transfers so that you could argue again over the master status. Hopefully winning for your side this time. And why not include a means to refuse a bid for master status right from the beginning. These are factors behind the many aspects of such protocols.

The ASTM protocol falls into this category. Control codes for line bidding and interruption are defined. This effectively tables the master/slave argument so that it might be fought day-in and day-out along your cables. The outcome (on average) might strike a balance, but much depends and the accuracy of the implementation. Unfortunately, the software coding may not fully implement all codes and timings as accurately as was originally intended. In some cases, one protocol implementation might be more successful at the negotiating than another, to the advantage of its owner. It is also feasible that, with such variations, an implementation might yield a particular match between system and analyzer that is inoperable. This is likely, even if by accident, as the specifications are not straight forward and not for the average programmer.

Now, back to the instrument that refused to cooperate. There's no transmission. In this case, the analyzer is likely to operating as a slave. It is waiting for the LIS to ask for its data. A simple request or poll sent by the LIS in the form of possibly only a single control character might unleash a flood of data. Another scenario, might have the instrument serving as a master. Here it is sending a very brief request periodically to the LIS. You may not have seen it if it only involved a few control characters sent every few minutes. Now the instrument is awaiting a response from the LIS indicating that it is ready to receive the result data. This response could also be just a few characters.

Some instruments take the master role and make the assumption that the LIS is always paying attention. In this case the entire result data message is transmitted no matter if the LIS is ready or not. Now the instrument sits waiting for some form of acknowledgment from the host that the message came through. The combinations or possibilities are great. It is the responsibility of the Transport Layer protocol to proper initiate the exchange and then carry it through to its successful completion.

The typical elements of a protocol include:

1. *Poll.* A message transmitted to request data from the interconnected device.

2. *Acknowledgment.* A message transmitted as a response indicating the successful reception of a previous message.

3. *Retransmission Request.* The previous message could not be understood. An error of some kind had been detected. The retransmission request asks the interconnect device to repeat the previous transmission. It is important that the number of retransmissions be limited so as to prevent an infinite retry situation. This would lock-up the connection.

4. *Checksum.* A calculation is made inclusive of all of the data characters within the message exclusive of the checksum itself. This is typically a summation modulo-256 of the 8-bit ASCII codes. The checksum is transmitted following the message. The receiver makes its own calculation and compares the result with the provided sum. If the totals match then the message is accepted assuming that it is error free. The use of 8, 16 and even 32 bit checksums can greatly improve the probability of error detection over that simple afforded by the parity bit. The checksum is appended to the message in binary or represented in some ASCII form. A binary checksum can be the source of problems should its value be confused with other control codes pertinent to the protocol.

5. *Longitudinal Redundancy Check (LRC).* A different form of checksum calculation involving the bitwise exclusive logical OR (XOR) of the ASCII codes in the message. An 8-bit LRC results from 8-bit ASCII data. The LRC is also appended to the message in similar fashion and with the same concerns as is the checksum.

6. *Circular Redundancy Check (CRC).* A calculation involving a polynomial equation providing a 16, 32 and even 64 bit result that can be used in place of a checksum or LRC with significantly better results. The choice of polynomial is made to facilitate easy calculation by the computer using combinations of exclusive OR, shifts and summation. The CRC is appended to the message in a similar fashion to the checksum with the same concerns.

7. *Line Bid.* A request used by a device to gain control of a connection. The device wants to become the Master over all other slave devices.

The Presentation Layer: Understanding the Message

The information passed from one device to another must be represented in a format that is universally understood by both devices. Unlike the low-level RS-232 bit arrangements and the sophisticated protocol used to the move the data, the formatting is something much more visible. Since most clinical instruments format their data in ASCII, the content of the messages can be easily viewed using a text editor. If a word processor is used, it is best to display the messages using a font with fixed spacing.

There are many variations used to format information. While it is quite easy to "see" the data, extracting the information automatically is another matter. Every aspect of the format must be accurately applied at the source and the destination must use a precisely identical set of rules to extract the information. It is often difficult to achieve such a match between source and destination. This is of concern as even the slightest misinterpretation of the procedure can result in interface failure.

TABLE III: Variable Field Formatting (reduced message size)

```
0,004,04a,BSC,951127,151534,90513049,C0,,009(cr)(lf)
1,WBC,00000,6.7,SAMPLE RERUN(cr)(lf)
1,RBC,00000,5.42,(cr)(lf)
1,HGB,00000,14.2,(cr)(lf)
1,HCT,00000,49.9,(cr)(lf)
1,MCV,00000,87.0,(cr)(lf)
1,MCH,00000,28.5,(cr)(lf)
1,MCHC,00000,36.9,(cr)(lf)
1,RDW,00000,14.2,(cr)(lf)
1,PLT,00000,326,(cr)(lf)
9,90513049,009(cr)(lf)<end of message>
```

We know that a typical *message* contains a block of bytes containing numeric byte values whose relationship to the letters and numbers that are familiar to us is specified by the ASCII standard. There is little confusion here as ASCII character tables are readily available. Yet, there are differing applications for the extended portion of the character set, those values from 128 to 255. In communications outside of the United States, many extended ASCII codes are used to represent the accents applied to various letters in other languages. Sometimes the extended codes are used to display graphical characters by providingvertical and horizontal lines, box corners and intersections, as well as, a couple of different line styles. A disagreement between data source and destination as to the character set can be a source of interface difficulty.

Each message may contain one or more *records* of information. The term "record" is usually used to describe a block of data that achieves a particular purpose or contains information about a particular item. In the clinical laboratory a record might provide the test result data for a particular patient. We would expect other transmissions then to contain records with test result data for other patients. Each record follows a defined structure.

The ASCII characters within a record are often formatted into *lines.* Although it is not required, many formats use lines to facilitate the display or printing of the information. Many formats define "lines" that are under 80 characters, the typical screen display or printer page width. Other formats permit lines up to 255 characters in length, a maximum that can be easily be handled by some programming languages like BASIC for instance. A typical line terminates with the carriage return character (ASCII 13). Often this character is followed by the line feed (ASCII 10) as might be necessary to move "down" to the next screen position or to advance the printer paper. In some cases only the line feed terminates a line. Confusion over line termination can provided yet another cause for interface failure. Lines, as well, need to have some structure.

Each line of data is broken into a number of *fields.* A field is simply a group of characters that usually define the value of a particular item. In the clinical laboratory this might be a single test result or the patient identification number. Each field is separated from the preceding and following fields by a *field separator*. This is usually some form of punctuation such as a comma or semicolon. It may simply be a space(ASCII 32) or tab character (ASCII 9).

If all of the fields in a particular data format are of a predefined character length then a field separator may not be present. In this case the receiving process extracts the individual fields from specific positions in the line. The test value, for instance, might begin at the 21st character position in the line and contain exactly 12 characters. This is known as a *fixed field* format (see **TABLE II**). It can be easily separated into its component parts or *parsed* but proves to be inefficient for transmission. Each field must be defined large enough to contain the longest non-blank character representation or *string* that can occur there. On average, many of the non-blank field values are shorter than the field length and therefore must be padded with additional spaces. Fixed field formats usually contain an high percentage of blanks which by themselves convey no information but must be transmitted.

If field separators are used, the field values can be of any length. This is called a *variable field* format (see **TABLE III**). No blanks are required to pad fields to any particular length and the format is much more efficient for transmission (and storage). The receiving process must work a little more to extract the field contents as each line must be scanned for separators in order to obtain the field values. Here the test value, for instance, might be defined as the third field in the line. Absolute character position being meaningless but the order of the fields must be clearly defined. Care must be taken not to include the field separation character within the value of any field. This would confuse the receiving process and result in interface failure. For example, if a comma (ASCII 44) separates fields, the comma must be strictly avoided in field values and there is a tendency to use the comma in the last name - first name representation of the patient name. To overcome this, some variable field formats enclose field values with quotation marks (ASCII 34) allowing the separator then to appear within the value but not other quotation marks.

TABLE II: Fixed Field Formatting (includes blanks)

```
0,004,04a ,BSC ,951127,151534,90513049 ,C0,   ,009(cr)(lf)
1,WBC ,00000,         6.7,SAMPLE RERUN                 (cr)(lf)
1,RBC ,00000,        5.42,                             (cr)(lf)
1,HGB ,00000,        14.2,                             (cr)(lf)
1,HCT ,00000,        49.9,                             (cr)(lf)
1,MCV ,00000,        87.0,                             (cr)(lf)
1,MCH ,00000,        28.5,                             (cr)(lf)
1,MCHC,00000,        36.9,                             (cr)(lf)
1,RDW ,00000,        14.2,                             (cr)(lf)
1,PLT ,00000,         326,                             (cr)(lf)
9,90513049  ,009(cr)(lf)<end of message>
```

Variations in field formatting can cause interface problems. Many fields are optional but must still be included as completely blank (all spaces) in fixed field formats or an empty field (no characters) with back-to-back field separators in the variable field formatting. As formats are expanded to accommodate new communications requirements, fields must be appended to lines keeping the previous field structure in the line constant and backwards compatible with existing interfaces. Sometimes optional *subfields* can exist within an individual field with these subfields having their own special field separator. Also, completely new lines are often added to records in order to provide new or additional information within the format. Such changes if not designed carefully can render previously functional interfaces helpless and in need of revision.

In the clinical laboratory each instrument performs differing test methodologies. A standard format wherein a field is assigned for each and every possible lab test is not feasible. First of all, the mix of tests is constantly changing. Secondly, we would not want to have to modify the transmitted format from each instrument when new tests are added. Generally the lab does not have the ability to do this. A standard format would be impossible and the LIS would have to use a different (custom) set of rules to interpret data from each instrument. Fortunately, there is a solution.

When an instrument sends a test result, it often identifies the test method with a test name such as “GLU” or “BUN” in a field preceding the actual value of the test in the format. This technique is sometimes referred to as *name-value pair* formatting. It can be accomplished in a number of ways. Sometimes and equals sign ‘=‘ (ASCII 61) can separate the name and the value as CHECK=“32.4” defining a CHECK value of 32.4. Quotation marks are not always used and name-value pairs might be restricted to one per line but not necessarily. This type of format can be found in many places. The CONFIG.SYS file on your PC, some .CFG files and the Windows .INI files all use variations of the name-value pair format.

The Application Layer: Getting the Job Done

Ultimately, data is successfully moved from the laboratory instruments through the interface to the LIS. The overall application often requires that some information flow in both directions. Only with properly operational interfaces can you then begin to actually do the work at hand: manage the information. There are a wide variety of LIS capabilities providing many benefits. A discussion of such things has been the topic of many articles and papers in the past and, no doubt, will continue to stimulate authors on into the future.

Interface Communication Standards

Background - the need for standards

Most of the problems associated with interfacing clinical analyzers to information systems are related to widely varying data format and communication protocol specifications across the population of analyzers. While there have been numerous discussions of establishing communication standards for instruments in the past, the analyzer communication environment remains chaotic. This results in considerable effort still being dedicated to establishing functional instrument interface connections.

There is also wide variation in the quality and accuracy of instrument manufacturers' interface specification documents, based on which most instrument interfaces are written. This creates the need to modify interface software on site in many cases.

This section will address the various elements of communication standards and describe some of the efforts to achieve standardization in this area.

Standards structure

There are a number of elements to be considered in the specification of standards for communication with peripheral devices such as clinical analyzers. These elements include:

- electrical specifications (RS-232 has been the standard for serial communications since the 1970's)
- connectors and cabling (a variety of connector sizes and cable definitions are still in use)
- communication parameters (baud rate, parity, data bits, stop bits, ASCII standard character set)
- communication protocol (the "handshaking" required between host and a peripheral device for data exchange to occur: ACK/NAK, Polling, Kermit and XModem are examples)
- message blocking (how does each side of the communication define where a message begins and ends?)
- message structure (what will be the order of data fields, or where is the specimen ID relative to the rest of the data record?)
- message content (what is the specimen ID and what are the test names and results associated with this ID?)

The ASTM standard

Within the past two years, standards for communication between laboratory analyzers and information systems have been defined by ASTM (American Society for Testing and Materials, Philadelphia, PA) Subcommittee E31.14. There are two standards that currently apply: E 1381-91 which defines protocol and E 1394-91 which defines format/content.

The conceptual goal of the ASTM standard is to simplify the connection of clinical analyzers to information systems, reducing the development time and expenses involved. Ideally, clinical analyzers would become plug-and-play, in which instrument connections could be established quickly and easily. Analyzer upgrades and replacements would have minimal effect on host system software.

In reality, only several analyzer manufacturers have thus far attempted to adopt the ASTM standards. Several of these manufacturers have only partially implemented the standards. At least one manufacturer who claims the ASTM documents as its interface specification does not conform to the standard, leaving the user and LIS vendor to sort out the differences on site.

There are several problems with the ASTM standards:

- They are far more complicated than is necessary for clinical analyzer - information system communications. This makes the development of interface software more complex (expensive) at the host end.
- The current specifications are extremely broad, allowing too much room for variances when implementing the standard. More complete implementation documents are required to provide analyzer manufacturers better guidelines in implementing the standard. This has resulted in a number of analyzers claiming compatibility with the ASTM specification actually requiring a specific interface program to handle.
- The communication protocol is not compatible with the way some analyzers operate. This has prevented some of the current analyzers from implementing the standard.
- The ASTM standard will have little or no effect on instruments already in place in clinical laboratories.

Much of the concern with complexity of the ASTM standard stems from the high number of data fields that can be exchanged with a clinical analyzer. Many of these fields are unnecessary for the average clinical analyzer, yet they must be accounted for in any implementation. Like many standards, the (ASTM) result is representative of the individuals defining the standard. In this case, it is heavily influenced by microbiology instrument manufacturers. Microbiology instruments have interface requirements often distinct from routine clinical instrumentation, and generally more complicated. The ASTM standards have extended this level of complexity to all analyzers. It is still possible to establish interface connections to all appropriate clinical analyzers, but it will take more effort to do so.

The current published ASTM standards documents are quite limited in scope. While they define the standard, no specific examples are included. This leaves too much room for interpretation by developers at the analyzer manufacturer for an effective standard to result. ASTM recognizes this shortcoming, and is making some efforts to improve the implementation guidelines available.

The ASTM standards were developed in 1991. They will not affect any of the analyzers designed prior to that time, including instruments in current manufacture. The standards are primarily being adopted by some new instruments coming to market, but not all. This leaves many clinical instruments in use which will not directly support ASTM.

Common Data Format

In 1984, CRT workstation interfaces were introduced that converted analyzer data into a Common Data Format and Communication Protocol (Dawning Technologies). The Common Data Format is flexible at the interface level and is able to be modified by the user or vendor into a best fit for a specific information system. A line of interface products including workstations, PC-compatible card interfaces, network control software and special entry consoles (Dawning Technologies) are also being manufactured.

While these efforts predate ASTM interface standards, the goal of the Common Data Format is identical to that of the ASTM standards: to simplify the connection of clinical analyzers to information systems and to reduce the associated costs by presenting a uniform data format to the host system from all analyzers. A number of these products provide both ASTM and Common Data Format output selections.

The HL-7 Standard

Several years ago, a group of manufacturers and information system vendors formed Health Level 7, or HL-7. This group attempted to define a standard for communication between various parts of healthcare information systems (HIS), principally system-to-system communication. In the course of their initial efforts, HL-7 became aware of parallel efforts at ASTM. While the HL-7 group is organized separately, the communication standard they have adopted is largely the work of another ASTM subcommittee within the E31 committee group.

The goal of the HL-7 group is to define (and refine) a standard which would allow information systems of varying manufacture within the hospital to easily connect to each other. The connection of different systems to an HIS has involved solving many problems similar to interfacing analyzers (widely varying formats, resulting in much custom software development). The HL-7 standard was originally defined for communications between laboratory (and other) systems and an HIS. Recently the HL-7 standard has gained acceptance for use between instruments and the LIS by the Veterans Administration (VA) and Department of Defense (DoD) facilities.

The IEEE MIB effort

Several years ago, an effort to standardize communication across devices found in the intensive care unit (ICU) setting was organized by the IEEE (The Institute of Electrical and Electronics Engineers, Inc., New York, NY). This group is working to define an intelligent network around a system called the Medical Information Bus (MIB) standard. The MIB is aimed at defining a new means of connecting the various patient monitors, ventilators, analyzers, etc. commonly found at the ICU beside. The goal is to allow clinicians to simply connect a new device to a wall socket, and have the information system immediately aware that the new device is on line, and where.

The MIB effort is longer range and more encompassing than ASTM standards efforts. The MIB will include a new design for each level of communication, beginning with a new proprietary connector for these devices. Thus far, the MIB project has resulted in demonstration of some early stage devices exclusively for the ICU. The MIB project will likely be active for several more years before finalized. The main participants are currently equipment and information system suppliers to the ICU system market. While the MIB project is not specifically aimed at the clinical laboratory, it is mentioned here because the results of their efforts may well have some applications in the laboratory in the future.

Planning for Instrument Interfaces

Clinical laboratories planning to modify or augment instrument interface connections to their laboratory information systems (LIS) should consider a number of factors before committing to any particular means of handling interfaces. As earlier sections of this article have indicated, there are a number of different types of instrument interfaces, and they are not all equivalent in features, function and performance. Careful examination of individual needs and expectations early in the process can help prevent surprises later on. This planning process is likely to be most beneficial when making decisions regarding a complete new LIS, but can also benefit those making small changes to an existing system. Some elements to be considered include the following:

Interfacing instruments currently in the laboratory.

The combination of available, useful and popular clinical instruments is in a constant state of change. This creates the need for frequent development of interface software, either directly on the LIS or within an intermediate interface device. The approach to interfaces offered by a vendor should accommodate with no delay all the instruments currently in your laboratory, provided each instrument has a stable interface available. Some very new instruments may not yet have final software in place, which may prevent interface development until complete.

Data buffering.

Most busy clinical laboratories require that analyzers be available at all times, without regard to whether the LIS is up or down. Data buffering provides the ability to operate an instrument without interruption during periods of host system downtime or unavailability during backup procedures. The data buffer is temporary memory storage between the analyzer and LIS. It is an important aspect of the instrument interface connection. Without data buffering capability, interface connections will be broken during host system downtime. If the analyzer is operated during that period, the information generated will likely have to be entered into the LIS manually when it comes back on line. The manual catch-up procedure can be very burdensome and may occur while additional specimens continue to arrive at the workstation.

Users should ask vendors specifically about how instrument buffering is provided, and check with other client sites for their experiences. The busier the laboratory, the more important is the attention given to data buffering and procedures to continue operation through routine system downtime.

Bi-directional Interfaces.

The overwhelming trend in clinical instrument communication is toward support of bi-directional interfaces, particularly on instruments capable of random access testing. A bi-directional interface can provide a significant savings in technologist time and improvement in accuracy over the life of the instrument. Users should have access to bi-directional applications as needed, including upgrades to existing unidirectional interfaces. Bi-directional applications for new instruments that may be acquired in the future should be available within a reasonable period of time. Broadcast and query mode interfaces should be available as needed. Users should require the vendor to specify which bi-directional applications are offered and how they work. Reference sites should be asked specifically about bi-directional interface applications for feedback. Bi-directional applications affect the highest volume, most critical instruments in the laboratory.

Operation of interfaces.

Since there are a variety of interface types and architecture, there are also a variety of routine operational requirements placed on the lab's clinical staff. When considering a new LIS or a single updated interface, the laboratory staff should make the LIS vendor clarify exactly how each interface will operate, what is expected of the technologist in routine operation of the instrument, and what procedures are required for host downloading in bi-directional applications. Particular attention should be given to multiple instrument situations, regarding flexibility in workload distribution across the instruments, downloading and cleanup procedures.

Integration with other LIS functions.

Users should know what processing of instrument data is occurring at what point in the system. Key system operator personnel in the lab should have a clear understanding of how the instrument interfaces relate to other important system operations such as result verification, range and delta checking, and QC functions. This is also important in establishing validation testing Standard Operating Procedures (SOPs).

Unusual or custom interfaces.

The requirements of each interface should be examined for any unusual aspects. These might include a requirement for additional calculations included in the instrument output, rescaling of data to better match a similar instrument, modification of test names and special handling of additional specimen identifiers or instrument flags. Many of these requests, while seeming perfectly reasonable to an individual laboratory, may not be applicable to every lab and therefore may be considered custom modifications to a standard interface at additional cost to the client. It is better to identify such requirements during the planning phase than when interface software development is underway or complete and installed.

Future instrument selections and upgrades.

The method of interfacing chosen by the laboratory should accommodate new instruments in the future with a minimum of disruption to laboratory operations and/or equilibrium of the system. It is only a question of time before the laboratory will require a new interface application, either as an addition to existing instruments or as a replacement for an existing unit. The interface method used by the LIS vendor should be sufficiently flexible to provide new applications in a reasonable period of time. Availability and costs associated with new interface applications are important questions for reference sites contacted. It is all too common in the LIS industry for users to wait close to a year for new applications.

A related concern is how to best handle software upgrades to existing instruments as they are made available by the analyzer manufacturer. The laboratory should be notified by the manufacturer in advance of any instrument upgrades that will affect the performance of an interface connection. The user should then notify the LIS vendor of the planned upgrade for advice on system implications of the change. In some cases, the vendor will have to provide modified host interface software to accommodate the instrument upgrade. Under no circumstances should the laboratory allow an instrument service person to leave the lab before function of the interface has been verified. If the new software installed in the instrument renders the interface inoperable, it may be necessary to reinstall the old software until the interface can be configured to accept the change.

Costs.

The costs associated with all elements of interface connections include host resident software, hardware, cabling and any costs associated with peripheral items. The costs for new interface applications should be clearly defined. Users should also understand what costs will be if modifications are required later, either to accommodate instrument software change or to add any desired custom features. The projected costs should be balanced against the functionality (value) provided by the resulting interfaces. Surprisingly, the prices charged by some vendors for straight-in software only interfaces are higher than those from other vendors for interfaces using intermediate devices that may provide added functionality.

Charges to users for new or modified interface applications vary widely between vendor and instrument. In many cases, the costs associated with interfacing are significant and must be planned for during the acquisition of new instrumentation. Often, an LIS vendor will provide a discount for multiples of the same interface. This creates financial advantages for using several copies of one analyzer where appropriate, or for using a front end interface system that can convert all instrument output into a standardized format.

Sometimes overlooked are the costs associated with providing sufficient host I/O ports and cabling. Instrument connections are most often made via serial RS232 ports, typically provided on multiport cards at the host end. At some point, providing one extra port will require an additional MUX (multiplexor) card or similar device at the host, which may be a significant expense. Similarly, the cost of connection cabling when combined with the cost of installing the cable through walls, ceilings, etc. can be more than originally planned. These costs may be reduced by networking analyzers to a smaller number of host ports (also using less cabling). Users should contact reference accounts and visit vendor user group meetings to acquire additional information on vendor costs and procedures.

Reliability.

LIS users quickly become dependent on functional instrument interface connections. Nothing less than 100% uptime is tolerated. Naturally, this is an impossible standard for any system to uphold while keeping system costs reasonable. Users cannot assume that the system will never fail. There are, however, many ways in which users can build in some redundancy of critical systems at a reasonable cost. Weak links should be explored and reference sites queried for experiences.

Physical layout of the laboratory.

Bench space in most clinical laboratories is minimal, and installation of an information system often adds many new items to the laboratory, such as terminals, printers, PC's, and cabling. These items must be anticipated and planned for. Instrument interface connections will introduce their share of new devices to the lab hardware confusion. There will be more cables to secure and some devices used to make the LIS connection. Here, again, there may be benefit to networking instruments found in one laboratory area to reduce cable confusion and the number of displays and other related devices. Perhaps several clustered instruments can share a single LIS terminal as well, further reducing hardware requirements.

Electronic data communication with analyzers is vulnerable to physical disruption (tripping on the cable) and to electronic noise. Noise can be produced by short duration fluctuations in power line voltage (sags and surges) coming from outside power sources, or generated by devices within the laboratory. Refrigerators, fluorescent lights and other devices with motors or high voltages involved are potential problems. Other analyzers in the laboratory can be sources of noise. A communication noise problem in one hematology laboratory was traced to the small tube rocker used to mix specimens. It was located next to the analyzer, powered from the same circuit, and close to the interface connection cable. The small motor in the rocker was particularly noisy (electrically), affecting the instrument interface. Powering the rocker from another circuit solved the problem which had taken several weeks to identify.

In the LIS environment, with many electromechanical devices interconnected, it is nearly impossible to accurately diagnose noise problems. Contaminating noise can be introduced via power lines, communication cables, printer cables, and monitor cables. Interface connection cables should be placed as far as possible from potential sources of noise. Communication cables should be built carefully, using shielded cable with proper grounding. Many instrument and computer manufacturers recommend connecting Pin 1 to the cable shield at one end only to reduce noise contamination potential. Large instruments should be connected to dedicated power circuits. Communication devices should not be powered from circuits shared by other devices with large motors or other noise producing components. Surge suppressers should be used where feasible. Data communication cables will not be equipped with noise suppressers, and remain vulnerable.

Vendor support.

Questions posed by the user should include: How does the vendor typically support instrument interfaces? What are the procedures for handling problems and the conditions under which additional charges to the client can be generated? Are on-site services available should they be necessary, or is everything handled via modem? The vendor should describe how interface support is handled, then the user should check references closely.

LIS Upgrading.

What options does the user have with interfaces when the LIS is upgraded? Are any major LIS upgrades currently scheduled? Will interfaces need to be redone or can the existing interfaces be carried to a new LIS application? Costs can be considerable when all the wheels must be reinvented.

Interface Validation Procedures.

Validation testing requirements for LIS are increasing as the FDA, AABB and other regulatory and industry organizations increase scrutiny of these systems. Instrument interfaces have generally been considered components of the overall information system, and validation tested in that context. The interface connection by nature has flexible elements, regardless of the specific interface type implemented. A Standard Operating Procedure should be developed by the laboratory for validating the performance of each instrument interface before releasing the interface to routine operation. This must include operation of the analyzer in all possible modes and conditions, and following the data stream to its various end points. A careful audit of analyzer and interface processed data should indicate no alterations from those produced at the analyzer, other than any data modifications built into the interface system with the approval or direction of the laboratories. This interface validation procedure should be performed any time a modification to the instrument or interface has taken place. Interface implementations that permit direct observation of the data stream at various points in the communication process may simplify validation testing. Documentation of the interface validation procedures, results and personnel involved should be retained.

Interface System Validation

Regulatory background

There is no question that the scrutiny of clinical information systems by government regulatory agencies, the FDA in particular, has increased significantly in recent years. The current FDA environment of increased regulation and enforcement indicates that this trend will continue for some time. To date, most of the attention has been focused on blood bank computer systems. This is the result of public and political concern over the safety of the nation's blood supply following the growing HIV epidemic. However, additional FDA draft guideline publications have clearly indicated an intent to eventually regulate all clinical information systems in similar fashion. Since clinical laboratories are increasingly dependent on information systems, formulating a plan to conform to such widening regulation is becoming more important.

The basis for FDA regulation of clinical information systems is the premise that the computer system is a medical device. In the regulation of blood bank establishments, the FDA has held that blood centers are manufacturing pharmaceutical grade products and that the associated information system is a process control device which directly affects the quality and safety of the resulting product. A blood bank information system may not only control production of the units available for transfusion, but also to which patients units are released. This allows FDA to regulate blood banks in the same way as pharmaceutical manufacturers, requiring them to conform to the agency's defined Good Manufacturing Practices. The result is a dramatic increase in documentation requirements to assure that all agency requirements are being met. These requirements include validation of manufacturing processes, procedures and systems.

The validation plan

The FDA has defined computerized systems to include hardware, software, peripheral devices, personnel and documentation. While detailed validation of the various elements of the system can be a major undertaking, it is becoming a normal part of the implementation of clinical information systems based on the experiences with blood bank systems. According to the FDA, validation is intended to establish documented evidence that the information system will consistently perform according to pre-determined specifications and quality attributes.

A first step is a close analysis of the system architecture (hardware and software), specifications and risks associated with the system. Following this assessment, a detailed validation plan can be developed. Any time hardware or software is modified or updated, the system should be revalidated prior to release for general use. A good plan should allow the validation process to proceed as efficiently as possible with minimal disruption to ongoing laboratory operations.

The plan should indicate what types of testing will be conducted, who will develop the test plan, how testing will be performed, who will run the tests, what forms of documentation will be available, who will review the testing and who will approve the testing. When the validation testing is completed, the plan with testing results and the various approval signatures and dates represents a complete documentation package for the validation process.

Incorporating instrument interfaces into the validation plan

Instrument interfaces are an integral part of the LIS, handling all of the data produced by analytical instruments in the lab. Since the architecture of instrument interfaces can vary from straight in software modules to intelligent intermediate subsystems, the validation plan should reflect the type of interface in use. The validation plan should define the various types of information which can be exchanged between instrument and system in all operating conditions. This includes handling of results, units, test names, flags, errors, control material records, calibration information, specimen type and any other information handled by the instrument.

Clinical instruments vary in the variety and type of information they process. There may be a number of user definable aspects to operation of the instrument which affect how it communicates with the LIS. This can include channel assignments, result units, format of results, test names, certain flag conditions, communication configurations and other parameters. The validation plan should reflect these differences and expectations for each instrument interface to be tested.

Finally, it has been mentioned earlier in this document that the performance of a clinical instrument interface is a state of equilibrium involving the instrument, interface and host LIS. There are applications in which an intelligent intermediate interface system performs some planned interpretation to the instrument data stream. This may include the addition of some calculated parameters based on available measured parameters, reformatting of results to an arrangement more compatible with other lab reporting, interpretation of flag conditions is a more desirable way, etc. These interpretive situations must be recognized and incorporated into the validation plan for appropriate testing.

Validation testing

FDA describes three aspects of a validation testing plan for blood bank systems: installation, process performance and product performance.

Installation Qualification

Installation qualification is intended to ensure that the system is suitable for its intended purpose, has been installed properly and is functioning as intended. In the instrument interface application, this would include validation of the instrument's performance on a stand alone basis and validation of proper installation and connection of any intermediate interface system. The host LIS should have the proper interface control software installed and available for testing.

This phase of validation testing should verify that all of the hardware and software components relative to instrument interface communication are installed and properly configured. Physical (cable) connections should be verified at each step, and the various port configuration parameters checked and recorded for the proper match at each end of communication. The instrument should be in the proper mode for the desired type of interface operation (many instruments have a number of available settings). Configuration of the interface subsystem should be documented as having all the necessary matching parameters set. Similarly, connections and communication parameters between the interface system and the host LIS must be documented as matching appropriately.

In applications which use an intermediate interface subsystem, it may be possible to further validate interface communication through the use of such tools as transmission logs or communication line monitors. This capability may be used to validate communication to/from the instrument and to/from the host LIS. All aspects of interface communication should be documented including listings of interface program code. Error checking algorithms should be used wherever they are available in the instrument communication chain.

Process Performance Qualification

Process performance qualification is defined as establishing confidence that the process is effective and reproducible. In the instrument interface context this can be interpreted to mean validation of communication between analyzer and host LIS in all possible modes. Performance testing is usually accomplished in a number of ways including normal, boundary, invalid case, stress and special case tests.

Normal testing includes cases that test the overall function and integrity of the system. Input data all fall within normal ranges. Sufficient repetition assures that the system is performing as intended under normal conditions.

Boundary testing uses input values which force the system to determine if the input is valid or invalid or to make a decision to execute another element of the software. In the case of instrument interfaces, this would involve the instrument processing specimens which would generate various range and error flag conditions which must be interpreted by the interface system and host LIS.

Invalid case testing checks for informational errors which should either not be communicated or handled in some specific manner. Examples include the instrument inhibiting result output for certain channels under specified conditions; the host attempting to download erroneous information to the interface; typographical errors at the instrument, etc.

Stress testing is generally testing of the system under volume load conditions. This challenges the system to perform as expected at its physical and processing limits. Such testing would involve operating the instruments at capacity while including error flag conditions and other record types. If an intermediate interface system is employed which serves multiple instruments, all instruments connected should be operated at capacity simultaneously. Buffer full and power failure conditions should be checked. This is typically worst case scenario testing.

Special case testing documents the system's reaction to specific types of data or lack of data and is intended to ensure that the system does not accept unsuitable data. Examples would include what happens with multiple instruments communicating simultaneously, missing data fields, or with result information outside of acceptable ranges. Instruments should be exercised to produce all possible record types.

Completion of these various types of system testing should make the operators more familiar with the system and should have generated every possible flag and error condition.

Product Performance Qualification

Product performance qualification involves validating the integrated system as a whole. This involves testing that the information produced by the clinical instrument arrives at the designated place in the LIS database intact, with all appropriate related information. All relevant storing and reporting aspects of the LIS must be examined and documented to ensure that result information being generated by the instruments are properly reported on any paper reports or electronic communication to another (HIS) system. In cases where some planned modification of the data takes place at instrument output, within an intermediate interface system or on the host LIS itself, the modifications should be tested for accurate performance under all conditions. Performance testing may be handled via **parallel testing** against another system and **on-line monitoring** once the new system is placed in regular operation. Parallel testing of a new LIS is very common prior to "going live". While parallel testing does not generally validate the system in all possible modes of operation and should not therefore be the only type of testing done on the system, it can serve as a valuable way to learn the new system and spot any irregularities with normal system operation prior to stress and other testing.

Analysis and review

Once the validation testing is complete, the resulting information must be compiled, analyzed and reviewed. This process will include organization of the testing documents and written explanations of why any tests produced results other than expected. In some circumstances, work-arounds will be identified to deal with any processing shortcomings.

Acceptance

With the validation testing completed and the documentation assembled, the last step is for the appropriate managers to sign in acceptance of the new/updated/modified system. The document then serves as a necessary record for any regulatory agencies and also as an important reference when any future modifications are made to the system.

Revalidation

Instrument interface applications in clinical laboratories are frequently modified, updated or added to the system as new instruments are acquired and implemented. Each time a modification is made to the configuration or operation of interface communication, this aspect of the system should be revalidated. This task is made significantly easier when there is an available validation plan and procedure from the original and/or last validation testing. Here again, there may be an advantage to use of an intermediate interface system as the communication links may be tested and validated in discrete sections with more complete documentation of relevant program code and configuration parameters available. This may permit faster validation testing by compartmentalizing information processing.

Onsite Personnel Requirements

While the initial installation, testing and validation of an instrument interface can be accomplished on a contract basis, an on-going need exists to monitor and maintain an operating interface. Whether it is a change in the way in which an instrument is utilized, the appearance of a rare sample, or an upgrade in some other aspect of operation, adjustments to interface operation are common. This creates an additional demand for on-site personnel.

The systems and interface suppliers can provide a significant level of support There simply is no substitute for an individual on-site who is familiar and comfortable with the technology involved. This can be key to identifying the existence of a problem, isolating the symptoms, and assisting with the debugging. Many of the troubles which tend to be most frustrating and costly are those which occur infrequently, intermittently or whose symptoms have been misinterpreted.

An on-site individual who is globally familiar with laboratory activities and the relation these have to the performance of the information system, can often provide the clues needed to identify, isolate and resolve an interface problem. The remote system or interface vendor relies on this assistance to lead them to the source of the issue. Only when the interface problem is simulated by the vendor on its own local equipment, can corrective action be taken and a verified solution be offered. Otherwise a trial and error process is required. An on-site contact can greatly speed the implementation and verification of corrective changes. The end result being minimal down-time and utmost system performance.

Unless a facility operates a significant number of interfaces, an on-site interface specialist may not be required. Responsibility for these activities might be combined with the coordination of the overall computerization in the laboratory.

Common problems with interfacing.

Instrument interfaces are prone to certain types of problems. These can be recognized and anticipated, and solutions planned in advance to minimize potential disruption of LIS function or overall laboratory operation. Typical problems follow:

Inadequate instrument specifications.

Instrument interfaces are developed primarily against the written interface specifications provided by the manufacturer. In some cases, insufficient information is included in the interface specification, or there are problems in obtaining the specification document. Users can do much of the background work in obtaining this information, and save the vendor considerable time. Laboratories planning to implement new interface applications or complete new LIS systems should prepare a notebook for the vendor including a section for each analyzer. These sections should contain the complete name, address and contact information for the manufacturer; specific Model number and serial number of the instrument; and a copy of the relevant interface specification. The technical support department of each manufacturer can provide the specification document, if it is not included in analyzer documentation already on hand. If an analyzer has a specific version of operating software, the specification should match the software revision. Some work is entailed in preparing this document, but interfaces will be up and running sooner with fewer problems and surprises.

Familiarity with instrument operation and configuration.

Interfacing an analyzer that has not previously been connected to an LIS, or altering an application from unidirectional to bi-directional usually requires that some minor changes be made to instrument configuration. Primary instrument operators must be familiar with any special menu screens that must be accessed on the analyzer to activate and configure the interface port. Communication parameters set at the instrument must match those set at the host or interface end for proper communication to occur. A simple mismatch of baud rate settings will result in garbage being received at the host. In addition, many instruments are capable of producing several different data formats. The instrument must be configured to communicate in the format expected by the interface or host.

The laboratory should anticipate making an instrument and operator available during installation of the interface. This is essential for generating data and testing function of the interface. The personnel involved with interface installation may not be familiar with the operation of the instrument. Many interface installations have taken far too long, simply for lack of coordinating the personnel resources required.

Sufficient time for implementation.

The laboratory should order any necessary interface components far enough ahead of time to allow for a comfortable installation process. Implementation of interfaces should not be treated as a just-in-time procedure. Sufficient time should be allowed for some debugging to occur. Installation needs to be done carefully, and validation testing performed before routine on line operation can begin.

New instrument interfaces.

Accounts with very new instrument models should be aware that the interface related software in the instrument may not be complete or stable for some time after introduction of the analyzer. Systems connected to the analyzer may require software changes as the instrument communication specifications evolve. If the laboratory is being charged for modifications to interface software, it may well wait for an instrument to stabilize before developing the interface application. When planning a new interface application, the vendor should be asked its experiences with new instrument models, and which of the instruments the laboratory plans to interface may present such problems. The analyzer manufacturer should be questioned regarding the status of instrument based software.

Future Considerations

As laboratory information systems become more sophisticated and strive to provide many more benefits, additional demands will be placed on the instrumentation connections. This trend is apparent. Over the past fifteen years the on-line analyzer has grown from a novel experiment to an absolute necessity. The functionality of these connections has increased from the simple unidirectional collection of results to the sophisticated bi-directional specification of testing requirements. Reporting requirements have continually increased the variety of information exchanged. Earlier instruments merely passed simple numeric values while those interfaces today exchange increasing amounts of statistics and demographics.

The resulting increase in the information flow between instrument and LIS has begun to stress the serial RS-232 communications technology. The need for more accurate and reliable connections has forced the development of increasingly complex protocols to control the flow. Today’s protocols implement additional levels of error control. This is becoming of more concern in the face of increasing government regulation.

The increase in protocol complexity has brought the lack of successful standardization to the forefront of concern. The sheer volume of the information exchanged has reached the limits of RS-232 capacity and the bottleneck effect is apparent. The use of data compression technologies has brought temporary relief. Clearly we will soon reach the end of the usefulness of the RS-232 connection. The on-going efforts to standardize communications in this environment will be for naught. We will have to leave RS-232 behind long before the benefits of RS-232 based standards in the laboratory are realized.

The reliance on the bi-directional connection of instrumentation will grow. This function, which in its original form simply specified the tests to be run on a sample-by-sample basis, now grows to incorporate varying levels of instrument control. Those familiar with control system design know that delays in the availability of information, sometimes even on the order of microseconds, can make the difference between function and failure. RS-232 serial communications cannot provide the level of response that will be required. Many other industries have long realized this and have adopted other technologies.

For the clinical laboratory, the future interfaces will incorporate the local area network technologies available today. Even the simplest of local area networks can move data at over 100 times the rate possible with RS-232 with the more common networks handling information tens of thousands of times faster. The hardware and technology is off-the-shelf. It is readily available and inexpensive. Many facilities already make use of it to link their systems with one another. It is only a matter of time before clinical instruments are equipped with a direct network connection.

The volume of information processed will continue to grow. Many devices now can report their results in graphical form. Graphic images, which were once drawn with ASCII characters, now are commonly done in high-resolution color. Instruments which can create and display such graphics today, cannot forward their images to the LIS. The RS-232 limitation is a barrier to this. Even in the network world, these graphical images are routinely exchanged using data compression as their size can be limiting at 10 million bits per second. Yet, graphics are becoming commonplace in the information industry and will likely become so in the clinical arena as well.

Of interest is the effect that this move to high performance local area networks (LAN) and graphics will have on the standardization efforts. In fact, the need for low-level standards development is eliminated. Widely adopted standards already exist. The interconnection hardware, the electronic elements, and the protocols are all defined. The first five (5) levels of the ISO reference model for interconnection discussed earlier are givens. The Physical, Data-Link, Network, Transport, and Session functions required to establish connection are already well-defined.

Even with graphics, many formats are widely in use and most systems can handle the common forms. The clinical industry will likely adopt a single format (.GIF for instance). What remains is the specification of the proper formatting for the actual test results, test requests, and other related messages. And it would seem that continued use of the ASCII formats that we commonly encounter today would be appropriate. This will certainly continue to require the attention of the standards groups.

Summary

This article has attempted to highlight some of the variables and considerations of which a clinical laboratory should be aware in dealing with instrument interface connections to information systems. In the past, LIS clients simply accepted what the vendor offered, with no questions asked. Many users are currently moving into third- and fourth-generation laboratory systems, and are asking more difficult questions of vendors prior to commitment. The vendors must provide more information and in many cases modify the product slightly to accommodate the client. Two final considerations for LIS users are user control and options.

User control.

Laboratory personnel should explore their level of control over instrument interface applications with a new LIS. Some vendors allow access to interface related code, which may enable users to implement their own new interface applications. In general, interface applications involving a peripheral device will have greater flexibility than straight-in software-only interfaces which need to be replaced completely with instrument change. Users should compare and contrast the various options for instrument interfaces to balance functionality, control and cost. Anticipating additions and modifications to the mix of instruments in the laboratory over the life of the system may alter the preferred method of interfacing to one that provides more user control. A new LIS will usually outlive a number of lab instruments.

Options for Users

The core staff of laboratory management should carefully consider all aspects of a planned LIS implementation, including methods of interfacing equipment. It may be that the best method of interfacing for a laboratory is different from the standard method supplied by a vendor. Most vendors are able to accommodate some changes to interface methods without great difficulty. Laboratory management should discuss interfaces in detail with the LIS vendor prior to execution of a new contract. Questions should be asked early in the process. The ability to make changes or consider other options for interfacing (and other elements of the system) may decrease exponentially following signing the contract.

References

1. Mischelevich DJ, Tuefel BG. Capacity planning guide for mainframes running the IBM Patient Care System (PCS). Computers in Hospitals 1981, Sept/Oct: 19-25.

2. ISO 7498-1984(E) International Standard for Open Systems Interconnection - Basic Reference Model, UDC 681.3.01

3. EIA-232-D, Interface Between Date Terminal Equipment and Data Circuit-Terminating Equipment Employing Serial Binary Data Interchange, Electronic Industries Association, ANSI/EIA-232-D-1986, January 1987.

4 Industrial Electronics Bulletin No. 9, Application Notes for EIA Standard RS-232-C, Electronic Industries Association, May 1971.

Med TechNet Presentations ...

The online discussion for this presentation can be found in Conference Area #61, on Med TechNet; and will be active from October 7 through November 4, 1996. Successful completion of the online post-quiz earns the participant 4.0 P.A.C.E.® CEU's or California State accredited credits.

This is a presentation of Med TechNet Online Services for the Clinical Laboratory. "Talks", such as this one, last for three weeks, during which time, the author and other participants discuss the topic. Users can participate by telnet'ing to Med TechNet on the Internet at, bbs.medtechnet.com, or dialing in directly, via modem, at, 1-716-688-1552, or, 1-716-636-8235. Discussions can also be distributed via Internet E-Mail.

For more information, visit the Med TechNet Web site, http://www.medtechnet.com, or call toll-free, 1-800-836-0720 (M-F, 9a-4p Eastern).

Med TechNet is a service of Western New York Microcomputer, Inc., PO Box 84, East Amherst, NY 14051, USA

The Dream...

Clinical instrument communications are completely standardized.

Every instrument employs a standard format, protocol and connection.

Your application is off-the-shelf.

Connections are established within hours.

Custom requirements are met in minutes.

Unlimited Customer Support provided without charge.

Is Reality!

dawning

Dawning Technologies, Inc.

Clinical instrument interface specialists since 1984.

IMPLEMENT IT!

sales@dawning.com
http://www.dawning.com
(800) 332-0499 (716) 223-6006