Interfacing communication network to IBM System/360 and System/370 host processors—An end users viewpoint

by JOHN T. M. PRYKE and LAWRENCE J. MUNINI

Wang Computer Services
Tewksbury, Massachusetts

IBM COMMUNICATIONS EQUIPMENT

Before discussing interfaces per se, it seems wise to review briefly the existing IBM communications environment. As shown in Figure 1, there are four basic types of hardware components. These are remote terminal devices, which may consist of high-speed RJE stations; low-speed time sharing terminals, (CRT or hardcopy); remote CPU’s; and a variety of other specialized remote devices (credit card readers, etc.).

These remote devices communicate with an IBM System/360 or System/370 host processor by means of either a 2700-series Transmission Control Unit (2701, 2702, or 2703) or a 3700-series Communications Control Unit (3704 and 3705). The 2700’s are basic, hardwired controllers which contain a limited amount of logic peculiar to the characteristics of each line and/or terminal device. This logic includes the ability to recognize special control characters, the assembly and disassembly of characters, and certain line monitoring to time-out inactive devices. The 3700-series Communication Control Units are actually communication processors, but are programmed to operate as perfect emulators of existing 2700’s by means of an Emulator Program (EP).

Connecting the host processor with a Transmission Control Unit is the multiplexor channel over which multiplexed messages (either in one or four-byte blocks) are transmitted back and forth to the network. The multiplexor channel, its architecture and command sequences are difficult to understand, and for many years have blocked development of effective non-IBM communication controllers.

In the host CPU, communications are currently handled by three basic pieces of software. First, there is the operating system (OS, DOS, or VS) which is linked to the user’s application program via an access method (Basic Telecommunications Access Method-BTAM, Queued Telecommunications Access Method-QTAM, or Telecommunications Access Method-TCAM). The interface between the operating system and the access method contains linkages and protocols which, if violated, will “clobber” the operating system. The linkage between the access method and the application program is on a macro or assembly language level (with the exception of TCAM which incorporates a COBOL-level interface) and requires considerable skill by an application programmer to communicate with the network.

To transmit (or receive) a message, the application program must first frame the message with control characters appropriate to the destination device. Next, an access method macro subroutine is called to translate the message from System/360 or System/370 EBCDIC code to the code used by that device (i.e., a Device Dependent Module). The application program next begins to build the actual message in a buffer area; once the buffer is filled, a channel program starts transmission of the message to the network. In the meantime, the access method enters a “WAIT” state until transmission is complete before the next operation starts. The access method is also responsible for all basic teleprocessing functions such as routing, polling, error recovery, and device dependencies. Needless to say, all of these activities consume CPU cycles and core within the host processor.

TYPES OF CHANNELS

It is to this environment that the manufacturer of a non-IBM communications processor must address himself. The input/output architecture of the IBM System/360 and System/370 supports three types of channels, each with its own operating characteristics. The three channel types are the byte multiplexor, selector, and block multiplexor (S/370 only).

The byte-multiplexor channel can operate in either of two modes, byte-interleave mode or burst mode. Operation in byte-interleave mode gives the appearance of multiple, simultaneously active devices on the channel to the programs executing in the host processor. This is accomplished by splitting all I/O operations into short intervals of time, during which only a segment of the available information is actually transferred across the channel interface. In fact, at any one instant, only one device at a time is logically active on the channel. The appearance of a multiplicity of simultaneously active devices is achieved by servicing
requests from a number of different devices at a rapid rate and by constraining each request to be very short in length (and thereby, in time).

Traditional IBM teleprocessing control units request service for one to four byte segments at a time. Obviously, there is some signalling overhead on each service request to identify which device it is associated with. The time for this overhead is appreciable, when it is multiplied by the number of segments required to comprise a message. This overhead tends to make byte-interleaving attractive for short messages from slow devices (such as keyboard interactive terminals) but not as economical for fast terminals which typically communicate long messages (e.g., remote batch terminals). Operation in burst mode is defined as dedication of the channel control functions to one device for the duration of an entire information transfer.

The selector channel operates in burst mode only. The utility of burst mode operation is that signalling overhead is only needed once per complete message transfer, instead of once for each tiny message segment as in byte-interleaving. This is why “fast” devices (e.g., tapes and disks) are usually run in burst mode. Telecommunications devices, on the other hand, do not usually interface to an IBM host processor in burst mode because of their slow operation (relative to internal host CPU and channel speeds). For example, a buffered terminal which “sends” a 1200 byte message to a host CPU over a 9600 baud line (assume 8-bit characters so that 9600 bits per second and 1200 characters per second and no other delays, such as line turnaround, modem latency, etc., are taken into account) would require one second of channel service time. During that one second interval, only about one thousandth of the capacity of the channel would have been used. Furthermore, if burst mode had been used, the channel would have been “tied up” servicing that device for the full second. All other devices attached to the channel would have been “locked out” for the duration. However, if the system operation were reorganized so the message was buffered in a device attached to the channel (instead of at a terminal at the other end of a communication line), then the limitation of the line speed would not be a factor in conveying the message to the host processor via the channel. Thus, burst-mode operation would be more applicable to telecommunications traffic.

The block multiplexor channel operates only in burst mode, but it has the capability to interleave blocks of data from multiple devices. This is done by servicing a pending request from another device, when the device occupying the channel in burst mode transfer reaches an end-of-block condition. The result is that block multiplexor channel operation has all the advantages of the high transfer rate associated with burst mode operation, and the disadvantages of channel lockout are reduced.

**INTERFACING OPTION**

From a user’s standpoint, there are three basic methods of interfacing communication processors to IBM System/360 and System/370 channels and mainframes:

- Emulators
- Intelligent Emulators
- True Front-Ends

As its name implies, an emulator is a plug-for-plug replacement of an existing IBM control unit. To date, the three basic types of emulator on the market have met with varying degrees of success. These include the 2700/3700 emulator (270X/370X), the tape drive emulator, and the disk drive emulator. Of these, the 270X/370X emulator is probably the most popular and has met with the highest degree of success. By its very design, it interfaces perfectly with existing IBM telecommunications access methods and program products, and thus requires no change whatsoever in the existing IBM mode of operation.

For example, if BTAM is the access method used with an IBM 2703 or 3705, it remains present and unchanged when running with a perfect 270X emulator.

The 270X emulators can be divided into two categories; hardwired controllers (such as the Memorex 1270) and software-coded minicomputers (such as the Interdata 270X or the Digital Equipment PDP-11). Although the hardwired controller offers substantial cost savings, certain features are available (such as automatic speed detection) which give it limited advantages over the standard IBM 2701, 2702 or 2703. However, the advantages end there. The user is still forced to rewrite his host processor access method macros and change terminal adapters to support new devices and/or different applications.

Software emulators offer the user some cost advantages, but their greatest benefit lies in flexibility. Effective communications oriented real-time operating systems and modular device drivers make it possible to support non-
IBM-compatible devices by changing the communications processor program only. Thus, a programmable base exists for future expansion to permit the minicomputer to take over most of the communication control functions from the host IBM CPU. Thus, it serves as a stepping stone to a true Network Control Program form of front-end.

A second class of software emulator appears as a sequential device on the IBM selector channel. For the most part, these devices emulate IBM 2803 tape controllers, for example, Action Computer System's communication controller. Processors which emulate 2314 disk control units have been somewhat less popular (e.g., the large Collins “C” System). Although these devices represent an attractive interface to user programs at the GET/PUT level, (thus eliminating the complexities of BTAM), sequential devices are designed for long message bursts at high speed. If considerable communication traffic is present in a network, it is possible as noted above to lock out other devices from the selector channel such as standard tape or disk drives, thereby limiting the user's access to mass storage media. The sequential device emulator, while thus attractive in certain applications, cannot be classified as a really effective solution to network control.

One step above a standard 270X type emulator, but not really a true “front-end”, is the so-called intelligent emulator. This device usually consists of a programmable communications processor which appears to the host IBM CPU as a 270X, but which offers a number of enhanced features through programming. For example, non-IBM-compatible devices can be supported by such special software and thus made to appear to the host program as an IBM equivalent.

A second example is the “fail soft” protection against cessation of host processor operation. Should the IBM mainframe cease to operate, a special program module would allow the emulator to broadcast messages to all users telling them to discontinue traffic because the central site is down. Through the communication processor console, variable text could be inserted in such messages to inform the users when operation will resume. A third example might be automatic port selection. In this case, all users would dial a single number for the host computer. By a simple sign-on procedure, the user would identify his device; the intelligent emulator would connect him with that “port” of the IBM CPU through which he can access the application he wants to communicate with.

A fourth possibility might be a local message switching capability. Each incoming message would be examined for a destination address; those bound for the host processor would be passed directly to it, while those bound for other terminals would be routed accordingly.

In all of these example, the communications processor appears as a perfect 270X emulator, only the functions performed by the processor differ. Therefore, without change to existing IBM telecommunications environment, users may realize considerable network flexibility above and beyond the control units supplied from Poughkeepsie.

**NETWORK CONTROL PROGRAM**

Finally, there exists the “true front end” or what IBM is promoting as the Network Control Program or front end NCP. Fundamentally, a communications front end is intended to remove the functions of controlling the telecommunications network from the host processor. The reasons for doing this are threefold.

The first is to reduce the cost of supporting a communications network. Most IBM teleprocessing installations have a significant portion of host processor resources tied up in terminal-dependent processing. Each software package that controls terminals has its own access method. For example, HASP users must use RTAM. CICS uses BTAM and special purpose applications; ATS has its own EXCP level programming built right in for terminal control. Accordingly, any installation supporting two or three teleprocessing applications usually has two or three teleprocessing access methods occupying core and using CPU overhead to execute.

A second motivation to front end an IBM processor is the desire to overcome the somewhat arbitrary S/360 and S/370 architecture restrictions, (they are not arbitrary from IBM's point of view; they are a fundamental marketing approach). For example, each terminal (line) is dedicated to a specific device address on the channel and thereby limited to one access method when the operating system is generated. To date, a single terminal could only communicate with one access method and thereby usually only with one application. Furthermore, since the terminals specific programming is provided in the host computer, and IBM primarily distributes software to control IBM terminals, the user cannot take advantage of the many benefits offered by the new terminals from other vendors. The terminal industry is currently one of the most rapidly changing segments of the computer field. Considerable advances are being made. Greatly expanded capability is being offered on new terminals by incorporating mini and micro processors right into the terminals at ever decreasing prices. Many IBM users would like to use these devices sooner than IBM is willing to let them.

The third motivation to front end is to make the network more easily accessible to application programmers. This means removing terminal I/O from the “bit banging” level of device-dependent modules and assembly language coding. Access to and from terminals needs to be provided to programmers working in higher level languages. Additionally, this type of software support should isolate the user, as much as possible, from the unique features of each terminal device. This device independence will make it easier to substitute new, improved, and lower cost terminals into the network.

From the standpoint of interfacing a front-end to an IBM host, the user must examine his individual requirements and decide which of the above problems he wants to alleviate. Then he can start selecting one of the many alternatives available to him. Unfortunately, the current state-of-
the-art of software technology does not furnish a general purpose solution to all of these problems. Thus, the user is forced to decide which specific area or areas he wishes to attack, and implement his solution accordingly.

Some software technology already exists to ease the third problem, that of easing access to the network from high-level languages. These are the TP monitors that are on the market. IBM's CICS and PMI's INTERCOMM are examples of these. They can be effective if a user can implement all his applications under control of the monitor. Then he only has one access method in his machine, and terminals can select which application they wish to communicate with. Perhaps in the future IBM will bestow upon the industry an improved access method which offers some of these advantages. Don't hold your breath waiting for it would be our advice, based on IBM's past record with teleprocessing access methods (remember QTAM?).

IBM users supporting a network of polled terminals usually suffer the most from the polling overhead being incorporated in the host. These users would realize an improvement in their operations by an intelligent emulator which does all polling and error recovery. In most cases, the host processor application would avoid drastic revision if it could be regenerated to use what it "thinks" are either unpooled terminals or the autopoll feature. The necessary programming could be incorporated into the intelligent emulator to achieve compatibility with the existing access method in the host. Following this approach usually makes it easier to support non-IBM terminals, since vendors of such systems are usually willing to offer this level of non-standard support as part of their product line. Additionally, some vendors offer support to enable the user to undertake this kind of activity himself provided the user has the expertise to do minicomputer programming himself.

SOLUTIONS

Limited solutions to the problems of overcoming S/360 and S/370 architectural limitations and IBM marketing strategy exist at present. Intelligent emulators can contend for channel device addresses based on activity when there are more lines than addresses. This is an interim solution which works well until a user wants more than about 160 active lines. Then he needs an access method which will map many lines onto a few channel addresses. Rumors of such an access method from IBM exist today. However, these rumors exist for S/370 users only. IBM has decreed that all its future enhancements will be restricted to virtual operating systems which only run on the S/370. People getting useful work done on S/360s, which are very cost-effective performers these days, are currently limited to emulators, intelligent emulators, and TP monitors to improve their efficiency. The best hope now is that after OS stabilizes for a period of time, some vendor will be brave enough to bring out a replacement access method the way some people are currently supplying enhancements to DOS. This would still be a mixed blessing at best, since existing applications would probably require a significant conversion and rewrite to utilize such a front end access method.

SUMMARY

In summary, in selecting a non-IBM communications processor, the user must take into consideration a certain number of basic factors:

- Perturbations to his existing applications environment
- Conversion to new hardware and software
- The impact upon existing IBM program products
- The effect on his overall telecommunications network
- Emulation versus intelligent emulation versus front-ending as a means of meeting his needs
- Support and maintenance
- Cost and benefits

Only when all these questions have been answered, can the user select that communications processor whose interface will truly match his environment, and which will effectively meet his needs.

BIBLIOGRAPHY