A REAL-TIME, INTERACTIVE, MULTIPLE COMPUTER SYSTEM

R. B. McDowell
P. F. Bohn
N. K. Brown
P. M. Kirk
A. G. Witte

Introduction

THE COMPUTER, IN THE PAST DECADE, has grown from a device normally employed to manipulate numbers to a virtual automaton capable of mechanizing a very wide variety of processes. As the variety of applications grew, it became necessary for some applications to tie several computers together. The connecting together of computers and the use of increasingly versatile terminals with these computers seems now to be replacing the main trend of the 1950's and early 1960's, which was toward larger computers. Large computers are still essential, and indeed are often the key to the interconnection of smaller computers and terminals; but the ability to interconnect is now a dominant characteristic in the present general expansion of the versatility of computers.

Probably the earliest concept of interconnected computers consisted of an analog computer tied to a digital computer to form a hybrid[1] computer. The original rationale for such a system was to provide a real-time simulation* of a guided missile, including aerodynamics, a long high-precision trajectory, and the analog control system. The digital computer was to handle the complicated function generation of the aerodynamics along with the computation of trajectory, while the analog computer was to handle the rather high data-rate, inherently analog functions involved in the control system. The high precision and easy function generation of the digital computer could then, in combination with the high speed of the analog machine, provide a single real-time simulation of all major aspects of a missile system.

The key problem in combining the two computers was the design of linkage equipment to match the input-output characteristics of the two computers, including the control signals necessary for proper timing of each machine's operations.

The first linkages or interfaces were designed in 1955 by EPSCO, Inc. for Ramo-Wooldridge Corp. (later Space Technology Laboratories) and Convair Astronautics. Key problems soon became apparent:

1. Too little control was centralized in the easily programmable digital computer and too many unsupervised all-hardware analog operations had to stay in precise voltage and timing adjustment for long periods of time.

2. The digital computer was too slow to keep up with the analog computer in many problems, and in any case had to be dedicated to handling one program at a time.

[1] For a comprehensive description of hybrid computers and their applications see the book *Hybrid Computation,* by Bekey and Karplus, John Wiley & Sons, New York, 1968.

* Simulation is the analytical or empirical modeling (analogy) of a theoretical or actual physical system.

Using existing technology, generalizing and extending it, a large real-time, interactive, multiple computer system has been developed. It is believed that this is the first time that real-time, interactive, and multiprogramming features have been successfully combined in a general-purpose computer network. The work done has been oriented to a major extension of the use of computers for simulation. However, much of the interface and software design and implementation are sufficiently generalized to have many applications in systems other than simulation.

3. The use of vacuum tubes, along with too little error checking in the hardware and software, resulted in poor reliability and difficult maintenance. The linkage, for example, required nine racks and contained several thousand vacuum tubes.

4. The analog computers required a device controller and redesigned devices such as autoset potentiometers, programmable logic, function switch control, automatic digital voltmeter addressing, etc. in addition to the mode control and data transfer operations that were initially incorporated.

5. A way was required for the operator/programmer to communicate on-line with both computers to modify his program or obtain various related computer services.

6. Efficient use of the digital computer required a way for non-real-time jobs to run during the increments of central processing unit time not required by the real-time job.

Despite these formidable limitations, both the Ramo-Wooldridge and Convair systems were used (albeit laboriously) to successfully solve several aerospace problems. As a consequence, an appreciation of what might be accomplished by a hybrid computer system spread rapidly within the aerospace industry, culminating in the last several years with the key role of hybrid computers to simulate the Apollo guidance, control, aerodynamics, etc. These massive simulations were used primarily for analysis, design, and testing before and during hardware development.

A second, and initially somewhat less conspicuous aspect of computer interconnection also began to be developed during the 1950's. This was the use of terminal devices located remotely,[2] wherever the user might be located, but tied to a central digital computer. The impetus for such terminals grew from three needs:

1. To extend standard computer peripheral functions beyond the central computation center, so that printing and program submission would be more conveniently available.

2. To provide on-line conversational features to the user, minimizing turnaround time and the user's dependence on computing center personnel. Such terminals as typewriters and graphics devices characterize these terminals.

3. To allow direct data acquisition and/or process control at remote sites to eliminate the need for separate, dedicated computers or the awkward

[2] An excellent text on this technology in terms of the hardware is the book *Telecommunications and the Computer,* by James Martin, Prentice-Hall, Englewood Cliffs, N.J., 1969.

and time-consuming manual transfer of data by way of magnetic or paper tape or by punched cards.

The terminals used to meet these needs are of a wide variety, but fall into two general classifications: (a) noncomputer or "nonintelligent" devices and (b) computer or "intelligent" devices. The basic difference between the two types is, of course, whether or not they have a general (usually programmable) computation capability. Minicomputers,† which have recently come into prominence, are finding a major application as the key element in the so-called intelligent terminal. Typewriters are far and away the largest category within the nonintelligent terminal classification.

Finally, a last and rather recent class of computer interconnection is that of the digital computer network where several computers, often quite large ones, are connected together to provide immediate access to a comprehensive data base. Usually such systems also provide extensive, though as yet rather specialized, computational and other data handling services.

All these types of system, however, must find solutions to the same basic hardware and software problems associated with the development of the hybrid computer. At this point, it will be useful to reorganize and restate these technical problems as follows:

1. The development of comprehensive software supervisory services to allocate or manage, as well as error-check, all hardware and software operations. IBM terms this the operating system. Others variously call it the monitor, supervisor, etc.

2. The development of parallel hardware subsystems, particularly for input-output (I/O) channels to speed up key functions by making them less dependent on the computer's central processing unit (CPU).

3. The use of solid-state components in place of vacuum tubes and electromechanical devices to increase both speed and reliability.

4. The development of sufficiently sophisticated controllers within the terminal devices or associated data transmission channels to allow simple software-generated commands to exercise all hardware features of the terminal regardless of how complex or specialized. It is important that this burden be placed on the interface for each device to be interconnected in order to minimize and hopefully eliminate hardware or software design changes to previously connected devices.

5. The development of terminal support software sufficient to allow utilization of all hardware features of each interconnected device. On-line operator/programmer communication with executing programs is a key feature of much of this type of software.

6. The development, as a part of the basic supervisory software operating system, of such features as multiprogramming, real-time services, and multiprocessing‡ which automatically adjust the computer operations to the changing demands on the overall system. These features are aimed at maximizing the efficiency of the total system, while at the same time providing each user with what appears to be a computer dedicated to his particular job.

A seventh key technical problem area for interconnected systems in general is reliable, long-distance, high-speed data communications. This, however, was not a major factor for the systems discussed in this paper.

Development Context

During 1967 and 1968 the Laboratory identified four primary areas where simulation requirements exceeded existing capabilities:

1. Hybrid computing.
2. Interactive, all-digital simulation coupled with operating hardware.
3. Missile and radar receiver simulation.
4. Radar system evaluation using realistic computer-generated target scenarios.

Such needs usually had been handled by separate, stand-alone, often special-purpose computers. In order to gain projected economies of scale in terms of cost per unit work, and to provide a generalized system capability more adaptable to future needs, a decision was made to develop a series of computer-to-computer links using the Computing Center's IBM 360/91 digital computer as a general-purpose central processor. Relatively specialized functions were minimized and confined to smaller computers and/or related in-

† Minicomputers are a class of small digital computer generally priced under $25K, using 16-bit words, accompanied by minimal software.

‡ Multiprocessing is a system of two or more computers which share adaptively, according to work load, their resources in memory, peripherals, etc. This was considered to be beyond the immediately forseeable technology and was, therefore, not attempted in our design. It is mentioned here for completeness.

terfaces.§ The other computers and specialized consoles essentially became terminals to the central computer. This in turn focused the engineering effort on the implementation of links and the support of these links by software in the central computer.

The 360/91 already contained a highly developed multiprogramming monitor (OS/MVT) as well as time-sharing of typewriter and printer/card-reader terminals. A conversational programming system (CPS) was also in operation, although it did not prove useful for our particular needs. Two million eight-bit bytes of main core storage and a large amount of disk storage were also available. However, there were four items that had to be developed:

1. A real-time monitor that would allow the multiprogramming part of the system to continue to operate in background.
2. An I/O channel that would permit at least 200K 32-bit words per second to be transferred simultaneously into and out of the 360/91.
3. Terminal support programs to allow interactive use of the links that were planned.
4. Appropriate terminal-end interfaces.

Probably the most important work done was in working out in detail the sequence of operations necessary to solve these four technical requirements. Specifically, we had to decide:

1. How the hardware would communicate with hardware.
2. How the software would communicate with software.
3. How the software would communicate with hardware.

Here we capitalized on the successes and failures of others, as well as the experienced assistance of IBM and Electronic Associates, Inc. (EAI). IBM developed a real-time monitor and a high-speed I/O channel adapted to the overall system plan. EAI likewise provided certain basic hybrid support software and an interface for the hybrid computer. Further work was done in-house to design and implement terminal support programs, in particular, most of the interactive features, and a sophisticated terminal-end interface to tie three digital computers into the 360/91 and into each other.

§ We, however, rejected the use of a smaller computer strictly for use as a data and/or control interface between terminal and central computer for the following reasons: (a) restricted I/O data transfer rate and (b) increased complexity of software and user programming, especially for real-time operation.

The intent, then, of our planning was to meet the four Laboratory needs listed above by interconnecting into a single system hybrid computer, all three types of terminal discussed earlier, and some aspects of a digital computer network. These added systems were to have a variety of interactive, real-time, and multiprogramming (time-sharing) capabilities. In addition, none of the standard capabilities already available on the 360/91 were to be impaired by the addition of the new interconnections and supporting software.

System Configuration

Figure 1 shows the overall APL computing complex. Figure 2 is a hardware block diagram of the links that are the subject of this paper. The hybrid computer consists of an EAI Model 680 analog computer connected to the IBM 360/91. The simulation console (SIMCON) allows external hardware to be connected to the 360/91 to operate with all-digital simulations. Missile and radar receiver simulations are accomplished on the signal processing simulator (SPS), which may be tied to the 680 alone or to the complete hybrid system via the 680. Finally, the radar system evaluation capability is implemented by tieing a portion of the radar through a Univac Model

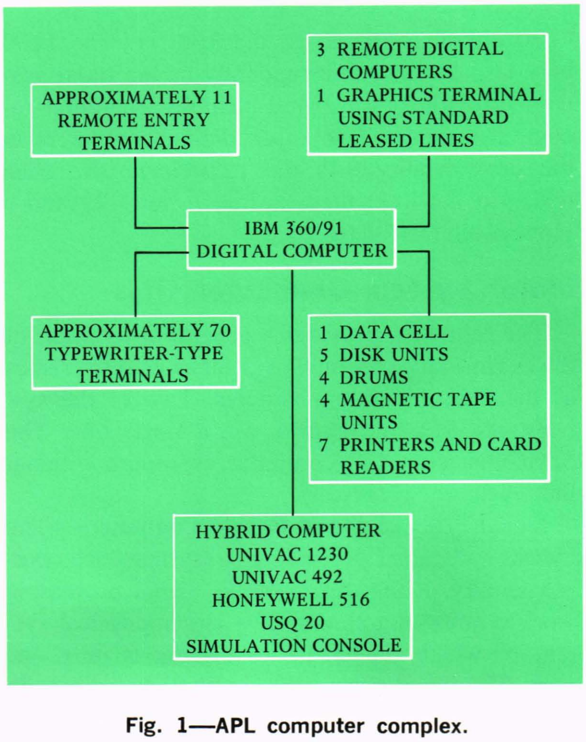
APPROXIMATELY 11 REMOTE ENTRY TERMINALS
3 REMOTE DIGITAL COMPUTERS
1 GRAPHICS TERMINAL USING STANDARD LEASED LINES
IBM 360/91 DIGITAL COMPUTER
APPROXIMATELY 70 TYPEWRITER-TYPE TERMINALS
1 DATA CELL
5 DISK UNITS
4 DRUMS
4 MAGNETIC TAPE UNITS
7 PRINTERS AND CARD READERS
HYBRID COMPUTER
UNIVAC 1230
UNIVAC 492
HONEYWELL 516
USQ 20
SIMULATION CONSOLE

Fig. 1—APL computer complex.

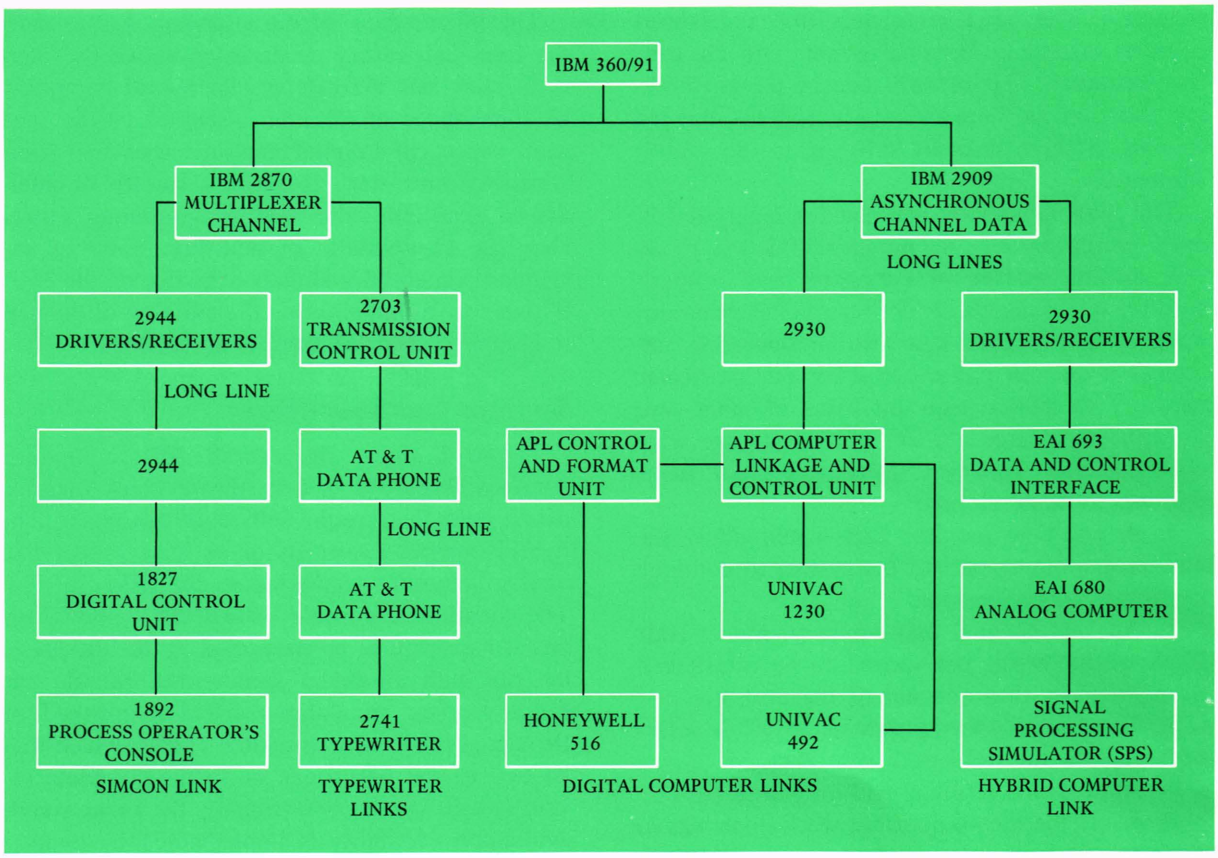
IBM 360/91
IBM 2870 MULTIPLEXER CHANNEL
IBM 2909 ASYNCHRONOUS CHANNEL DATA
LONG LINES
2944 DRIVERS/RECEIVERS
2703 TRANSMISSION CONTROL UNIT
2930
2930 DRIVERS/RECEIVERS
LONG LINE
2944
AT & T DATA PHONE
APL CONTROL AND FORMAT UNIT
APL COMPUTER LINKAGE AND CONTROL UNIT
EAI 693 DATA AND CONTROL INTERFACE
LONG LINE
1827 DIGITAL CONTROL UNIT
AT & T DATA PHONE
UNIVAC 1230
EAI 680 ANALOG COMPUTER
1892 PROCESS OPERATOR'S CONSOLE
2741 TYPEWRITER
HONEYWELL 516
UNIVAC 492
SIGNAL PROCESSING SIMULATOR (SPS)
SIMCON LINK
TYPEWRITER LINKS
DIGITAL COMPUTER LINKS
HYBRID COMPUTER LINK

Fig. 2—Interactive, real-time computer links.

1230 digital computer to the 360/91. The 1230 may also be used independently of the radar system, providing a general-purpose computer-to-computer mode. Two other digital computers in the same facility with the 1230 have also been tied into the link; these are a Univac 492 and a Honeywell DDP-516.

Major System Characteristics

The following provides a general description of the performance features and other characteristics of the four simulation systems. Two of the systems, the SPS and hybrid, use a single link. The SIMCON and 1230 computer use one additional link each.

Signal Processing and Hybrid Computers—The hybrid and signal processing subsystems are complementary in that much of the signal processing work requires simulation of a larger physical system of which the signal processing is only one part. The signal processing simulations, so far, do not require hybrid capability, except as a part of the larger simulation that already involves the 680 analog computer. Therefore, the 680 and the SPS use a single link to the 360/91.

The SPS is designed primarily for signal frequencies up to 1 MHz, and possibly as high as 5 MHz in the future. This frequency range does not allow use (with presently available technology) of such elements as potentiometers that could be automatically controlled. Therefore, there are no means provided for other than manual setup for the SPS. Its only connection with the 680 is through data lines to the 680 analog patch board.

The 680 computer, on the other hand, may be fully controlled for setup and run control from the 360/91. The 680, via the 693 interface, allows automatic mode control, potentiometer and function relay setup, component readout, data transfer, etc. This control may be exercised from the digital portion of a simulation program or from independent diagnostic programs. Both types of program may also be run as batch programs or may allow control, on-line and interactively, by an operator through a typewriter terminal. Of course, full manual control from the 680 console may also be used and can override the digital control.

Since the analog portion of the hybrid system is remote from the central computer and since the central digital computer is not dedicated, digital I/O equipment and a hybrid system operator's console must both be located remotely in conjunction with the analog computer. An IBM 2780 line printer and card-reader terminal are used for the I/O. An IBM 2741 typewriter terminal is used for the operator's console, along with special supporting software.

Simulation Console (SIMCON)—The SIMCON consists of an IBM 1892 process operator's console and an IBM 1827 interface. The 1892 control panel is shown in Fig. 3. It allows interactive control of all-digital programs with the following features:

1. On-line mode control such as initialize, operate, hold, stop, and terminate.
2. Data manipulation such as enter data (from a keyboard), display data, save data, and step a parameter value.
3. Special pushbutton functions such as print parameters, change integration method and increment size, and test all console functions.
4. Operational and diagnostic alphanumeric messages to the operator.

The SIMCON has been designed, along with its supporting software, to provide virtually any conceivable intervention (except a conversational language) and that a user might want for hands-on, on-line control of his job. Special attention has been given to providing all the capabilities traditionally available to the analog computer operator. Output consists of printouts, line printer plots, X-Y plots, digital and analog strip chart plots, and dual-trace memory-scope plots.

The 1827 interface provides analog-to-digital (A/D) and digital-to-analog (D/A) converters to drive the analog plotters. These converters also make it possible to tie hardware into digital programs.

The 1230 Link—The 1230 link provides two modes of operation: (a) computer-to-computer and (b) radar system to 360/91. The computer-to-computer mode allows the Univac 1230, Univac 492, or Honeywell 516 computers to transfer data back and forth to the 360/91. Only one of the three terminal computers may use the link at a time. The terminal computer is selected from the 360/91 by the user's job program. This program continues to operate as the master program, coordinating the operation of the two computers through the appropriate interrupts, control lines, and systems software. The program on the terminal computer is thus a slave operation that may do computation and other operations solely associated with the terminal computer on an independ-

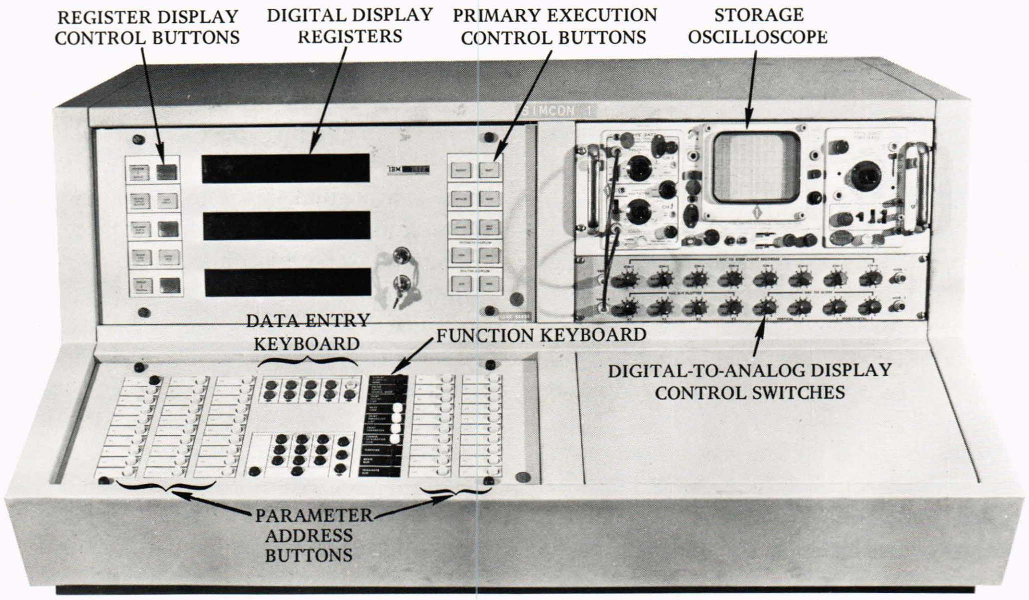
REGISTER DISPLAY CONTROL BUTTONS
DIGITAL DISPLAY REGISTERS
PRIMARY EXECUTION CONTROL BUTTONS
STORAGE OSCILLOSCOPE
SIMCON 1
DATA ENTRY KEYBOARD
FUNCTION KEYBOARD
DIGITAL-TO-ANALOG DISPLAY CONTROL SWITCHES
PARAMETER ADDRESS BUTTONS

Fig. 3—Simulation console.

ent or synchronized basis; however, it has no direct control over the link, the central computer, or master program. Carrying out job programming with this link requires a higher degree of awareness of the associated access methods and related computer hardware characteristics than for the other two links, where all required digital programming is executed by the central computer. The key factors that must be handled by the programmer are communication between job program and system software, division of job program functions between computers, and timing or synchronizing the execution of dependent parts of the two job programs.

The second mode of operation allows a partly simulated, partly hardware, radar system to operate in real time, with simulated targets being generated by the 360/91. The 1230 computer, in this mode, is a part of the radar system and contains the master program. The 360/91 target generation program is a slave to the 1230 program in certain respects but also continues control over the link in synchronism with similar operations managed from the 1230 end of the link.

As with the hybrid system and the SIMCON, the radar operation requires an operator's console to allow the interventions of starting, stopping, terminating, and changing parameters in the 360/91 program. Similar intervention on the 1230 may be carried out at the 1230 console, although such software must be coded by the job programmer since there are no standard software support packages or monitor. To implement typewriter operation, an acoustically coupled TST 707 portable typewriter terminal is located next to the 1230 console.

Link Organization

The generalized hardware form for any link is shown in Fig. 4. While the primary engineering effort is centered on the interfaces, consideration must also be given to transmission between interfaces if the distance is more than 100 feet. All three links at APL involve distances of 800 to 1000 feet.

The software required for the links consists, at a minimum, of access methods to allow applications programs executed by the central computer to use the hardware interfaces, and through them, to use the terminals. Other software, not necessarily concerned with driving the links themselves but oriented primarily to the applications for which the terminals are to be used, consists of special monitors, translators, emulators, test and diagnostic routines, and the terminal support programs.

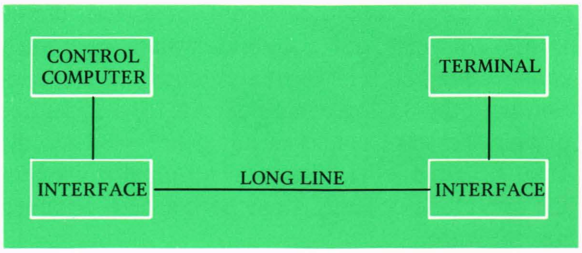

Fig. 4—General purpose computer link organization.

Basic Control Functions—There are four basic types of control function necessary to any linkage system: request, initiation, checking, and termination. Depending on the practical applications for which the system is designed, there is a variety of subfunctions associated with each of these four functions.

The *request* function can be carried out by addressing or selecting one device rather than another. This might be quite passive if the request is acknowledged only when the addressed device is idle. A further step can be taken by allowing such requests to be queued and possibly some status information sent back to the device initiating the request. Beyond this, deliberate multiplexing, either through software or hardware, can be employed.

A more imperative form of request referred to as an interrupt is also used. An interrupt system, while it might provide for queuing, requires that the device being addressed stop its present operation and begin a new one. Completion of the first operation is generally suspended temporarily, rather than being obliterated, and is resumed when the interrupting operation is completed.

Rather than accepting interrupts on a first-come-first served basis, it is desirable to establish priorities for the interrupts. It can easily be seen that if the last interrupt to arrive is always accepted, the storage and control for operations idled by successive interrupts could become unmanageable. One way out is to inhibit and queue any interrupts following the first one until that operation is complete. This is not necessarily satisfactory; certain following interrupts, such as for a real-time device, might well require more immediate servicing. In this case the inhibiting is carried out only for lower priority interrupts.

The function of *initiating* a given operation, once the request phase has been handled, has a number of parts: (a) clear interfaces and/or requested device, (b) initialize, and (c) start operation.

The clear phase is often necessary as a guarantee that any registers and other control circuitry that could possibly provide erroneous control of the new operation are not left active.

The initialization phase is carried out by the requesting device, selecting the mode of operation of the requested device and supplying the necessary parameters.

The start-operation phase completes the necessary "handshaking" through such events as READ READY, READ REQUEST, READ, or whatever the appropriate control sequence is between devices, along with the various acknowledgments so that data transfer begins.

Certain *checking* is normally carried out during data transfer by both hardware and some minimum of error-detecting software. As more checking is done by hardware, less time is required in overhead (nonjob program software execution time on the CPU). As more software is employed in detecting faults, the overhead unavoidably increases since software in general must operate sequentially, while hardware is generally employed in parallel. For real-time operations this is a major consideration.

Finally, data transfer must be properly *terminated*. This can be accomplished through initialization parameters, through step-by-step handshaking control, or through control exercised asynchronously by the requesting device.

Data Transfer—The data transfer function, which is the main object of developing computer-to-computer links, consists simply of two functions: read and write. The purpose of any link is to allow I/O between two devices. The substantive engineering content of handling data transfer is in terms of coding and formatting the data.

Coding refers to the meaning to be carried by the data signal pattern. The meaning is predetermined, of course, usually according to some rule, and the hardware and software are designed accordingly. In the APL links, the signal pattern is binary. Numbers are, in most cases, represented as binary without further coding. Alphabetic and special characters are represented by EBCDIC (Extended Binary Coded Decimal Interchange Code).

Formatting refers to the order in which the basic signal patterns are arranged. Such conventions as the following come under this heading:

1. Word length and character length.
2. The order in which particular words or characters are transmitted, particularly where control-type words are included.
3. The use of blocks or other groupings of words in batches.

The key consideration involved in the APL links was how to reliably convert the codes and formats of dissimilar devices at minimum cost. This required some tradeoffs between software and hardware, as well as providing several modes of operation for a link to accommodate a variety of devices.

Signal Transmission—Signal transmission is concerned with the transmission medium, modulation and demodulation, transmitters and receivers, noise, bandwidth, signal shaping and regeneration, repeaters, level conversion, and deskewing.

The APL system uses RG-62 coaxial cable for the 1230 and hybrid links, partly suspended above ground and partly buried in metal conduits bundled in groups of 24 cables with an outside sheathing of PVC. The data words are transmitted in parallel, 30 bits parallel for the 1230 link, and 16 bits parallel for the hybrid link. Cable lengths are 1000 feet.

IBM 2930 line driver/receiver and regeneration equipment is used at the remote ends of the 1230 and hybrid links. Voltage levels are 0 to +3 volts. Rise times after regeneration are maintained at 10 ns. Before regeneration, rise times are allowed to stretch up to 150 ns. The 1230 and hybrid links connect to the 360/91 memory bus via the 2909, which has the same type of drivers, receivers, and regeneration circuitry as the 2930's.

The SIMCON uses 22 AWG twisted pair cable, for 16-bit parallel data transfer. The cable length is 1000 feet, and two IBM 2944 driver-receiver-regenerator units, one at either end, are used.

Data rates are about 200K 30-bit words per second for the 1230 link, 200K 16-bit words per second maximum for the hybrid link, and 167K 16-bit words per second maximum for the SIMCON link. These are the maximum rates at which the interfaces can operate. The maximum rate of the 360 memory bus is 2M bytes per second between CPU and 2909 and CPU and the 2870 multiplexer. Cable delays and software overhead reduce these maximum rates to about 167K words for the 1230 link and 100K words for the SIMCON. The hybrid link is limited by the A/D converter and software to about 100K words per second.

Noise is minimized by outside shields around cable bundles and the use of 4/0 fine-stranded welding cable as a ground bus running between buildings, terminating at each end in the equipment complexes. The finely stranded 4/0 cable is felt to be a key to minimizing both AC and DC noise, particularly since substantial power ground levels as well as a myriad of AC sources from computers to radars can produce large voltage differences. Noise voltages in the transmission system are maintained under 10 mV from DC to 10 MHz.

Central Computer Interfaces—As described earlier, computer links generally require interfaces at both ends. Those interfaces at the central computer end are channels into and out of the computer. There are three basic kinds of channel that provide access to IBM 360 computers as shown in Fig. 5.

1. Multiplexer Channels (Model 2870).
2. Selector Channels (Model 2860).
3. Asynchronous Data Channel (Model 2909).

The first two types of channel are standard. The last one (2909) is provided subject to customer specification.

These channels are representative of two basic methods of providing access to computers. The multiplexer channel allows connection of many peripheral devices to the main computer memory

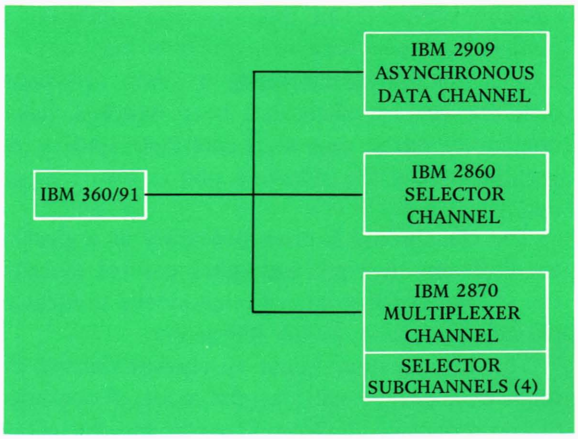
IBM 2909 ASYNCHRONOUS DATA CHANNEL
IBM 360/91
IBM 2860 SELECTOR CHANNEL
IBM 2870 MULTIPLEXER CHANNEL
SELECTOR SUBCHANNELS (4)

Fig. 5—APL's IBM 360/91 channel configuration.

by switching from device to device, giving each a portion of the time available.

The selector channel in contrast is, in principle, dedicated to a single peripheral device or semi-dedicated by being able to address a few devices, but only service one at a time until each successive one is finished. Also, rather than using the time left after the needs of the CPU are met, it takes priority in accessing the memory, maintaining control until its data transfer is complete. In many machines, this sort of operation, and variations on it, is called by its more or less generic name of direct memory access.

The asynchronous data channel is a combination of features from the multiplexer and direct memory access types of channel. It has all the features of the selector channel, but can also multiplex its own multiple subchannels according to its own priorities. It is, therefore, able to provide maximum speed transfers of data to and

TABLE 1 COMPARISON OF THREE TYPES OF IBM INPUT-OUTPUT CHANNEL

	2909	*2870*		*2860*
		Main Channel	*Selector Subchannel*	
Maximum Data Rate	2 Million Bytes/s	110 K Bytes/s	180 K Bytes/s	1.3 Million Bytes/s
Type of Data Transfer	Multiplex	Multiplex/Burst	Burst	Burst
Maximum Distance to I/O Device (ft)	1000	200	200	200
Data Path Size to I/O Device	1,2,4, or 8 Bytes	1 Byte	1 Byte	1 Byte
Hardware Hierarchy	Channel-Subchannel-Device	Channel-Control Unit-Device	Channel-Control Unit-Device	Channel-Control Unit-Device

from the CPU to several external systems up to an aggregate rate that equals the main memory input/output speed and forces all other computer operations to be suspended as necessary. This channel is, consequently, the most suitable for handling real-time I/O.

Table 1 gives a comparison of the characteristics of the three channels. The IBM 2870 multiplexer channel has a maximum data rate of 110K 8-bit bytes per second (b/s) and up to four selector subchannels may be added, with each subchannel having a data rate of 180K b/s. A maximum of eight control units and a theoretical maximum of 196 I/O devices may be connected to the IBM 2870 main multiplexer channel. In addition to this, up to eight control units and 16 I/O devices may be connected to each selector subchannel on the 2870.

As the name implies, the multiplexer channel uses time division multiplexing to interleave data to and from control units. However, a burst mode of data transfer between the channel and control unit is also possible using the selector subchannels which are similar to selector channels in that they are dedicated and relatively high speed. In this mode a control unit locks out all other control units on the multiplexer until the burst of data is complete.

The IBM 2860 selector channel operates only in the burst mode and has a maximum data rate of 1.3 million b/s. Because of its speed, drums, disks, and sometimes magnetic tape units are generally connected to a selector channel. A theoretical maximum of eight control units and 256 I/O devices may be attached to a selector channel. It should be pointed out, though, that while 256 is the maximum number of devices that can be uniquely addressed by the selector channel, the physical number of I/O devices is usually much less than 256, after being dedicated to one or two similar units such as drums or an extended core memory. Also, the 2860 does not truly multiplex or interleave data transfers between the various devices. It selects a device and then proceeds to completely service that device to the exclusion of the others.

The IBM 2909 asynchronous data channel (ADC), while not associated with standard system 360 I/O channels, is of particular interest in the simulation environment at APL. It is able, among other things, to handle both real-time devices under the real-time monitor and other devices under OS/MVT simultaneously. The ADC is composed of a main channel and up to 32 subchannels that may contain some of all the functions otherwise carried out by a terminal-end or controller interface. The subchannels can be designed by IBM for special applications. Three types of subchannel are currently installed in the 2909 at APL (see Fig. 6).

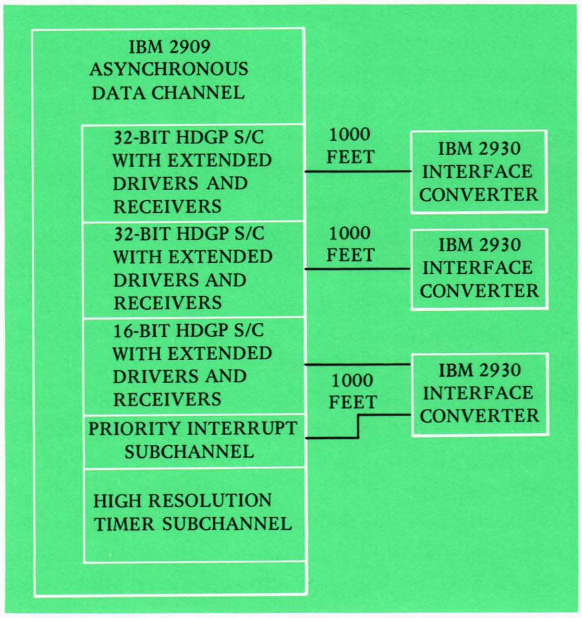

Fig. 6—APL's IBM 2909 subchannel configuration.

Half Duplex General-Purpose Subchannel (HDGP). The HDGP subchannels, which allow data transfer in both directions, but not at the same time, were specified by APL for data links requiring high-speed data transfers, generally from real-time drivers, when each data word consists of 16 or 32 bits. Two key features marked the HDGP subchannel design: (a) maximum data rate up to the 360/91 memory bus rate of 2.0 million 8-bit bytes and (b) drivers and receivers built-in which are capable of sending and receiving such signals over 2000 feet of coaxial cable with virtually zero probability that the usual ambient noise will cause errors.

In addition to these key features, a variety of control lines, some of which are optional, was provided. This allows more sophisticated error detection and error correction with software or special hardware. It also allows external devices to be multiplexed, or for the devices to interrupt on a priority basis, and it allows for off-line checkout of the 2909 with one subchannel driving another for a so-called loop test.

IBM 2909 Priority Interrupt Subchannel. The IBM 2909 asynchronous data channel at APL also contains a 16-level priority interrupt subchannel that allows 16, 32, 48, or 64 levels of priority interrupts. Each priority interrupt is uniquely identified by an address in the I/O old program status word during an I/O interrupt. Priority interrupt levels may be enabled, disabled, or reset by issuing commands to the subchannel priority mask register from the 360/91. Priority interrupt levels of lower priority are inhibited during the processing of a higher level priority interrupt. Software support packages are required before any use is made of the subchannel.

IBM 2909 High-Resolution Timer Subchannel. The high-resolution timer subchannel is a timing device that may be used as an interval timer or an elapsed time counter. The subchannel contains a 32-bit register that may be loaded with a count. The count is then decremented at the rate of the frequency generator in the subchannel. The decrementing rate may be from 1 to 127 μs/bit of the 32-bit register.

The high-resolution timer subchannel is controlled by the read direct and write direct instruction of System 360. An external interrupt is used to signal the 360/91 that a time interval has elapsed. Because of this, special software packages are needed on System 360 in order to utilize the high-resolution timer subchannel.

Terminal-End Interfaces—The immediately preceding material has provided the key characteristics of the central-computer-end interfaces. This section will describe the terminal-end interfaces. The computer linkage and control unit (CLCU) and the control and format unit (CFU) are entirely APL-designed. The CLCU will be described in some detail. The CFU is similar to the CLCU and will, therefore, be described more generally. The EAI 693 data and control interface was partly built to APL specification and will be described with minimum discussion of the standard characteristics. The IBM 1827 is a completely standard unit, although APL did provide a black-box modification to allow attachment to the 360/91, for which computer the 1827 is not a standard connection. Nevertheless, the 1827 will be assumed standard for the purpose of this paper and described only generally.

Computer Linkage and Control Unit (CLCU). The CLCU, as shown in Fig. 7, is the primary interface connecting the Univac 1230, Univac 492, and the Honeywell 516 to the 360/91. For the 516 it operates in conjunction with the CFU which makes the 516 conform to Univac characteristics on which the CLCU operation is based. The CFU also allows a mode of operation where the 516 is made to conform to IBM characteristics allowing it to replace the 360/91, where the 516 and 1230 are to be interconnected with each other.

The CLCU allows, then, both Univac computers (one at a time) or the 516 to connect to the 360/91. It also allows the 516 to connect to the 1230. And it further allows the radar synchronizer associated with the 1230 to be connected in either a 1230-to-91 mode or a 1230-to-516 mode.

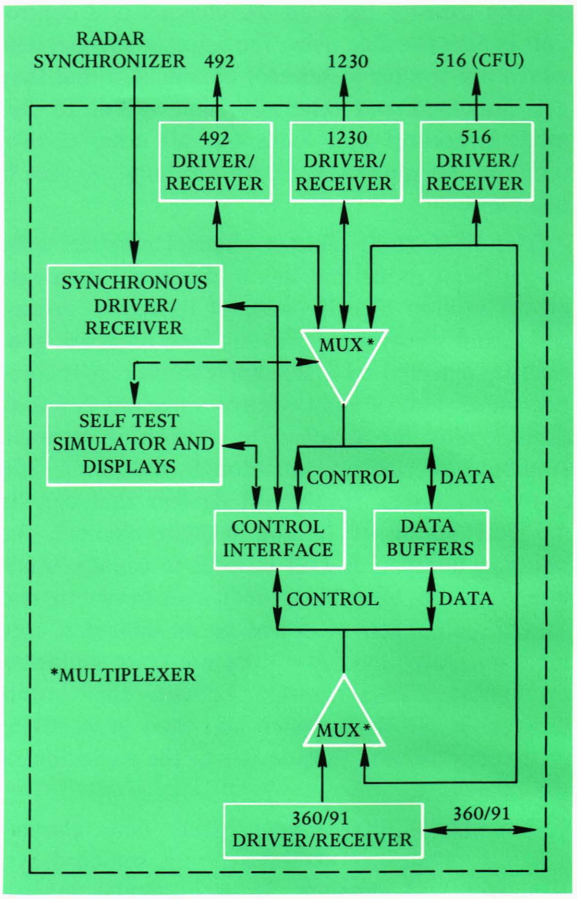

Fig. 7—Block diagram of computer linkage and control unit (CLCU).

In its present configuration, the CLCU is dedicated to one mode at a time, switching to a new mode only after the current user-computers indicate that they have terminated all communication. Provision has been made, however, to convert the CLCU to mode time-sharing under control of a suitable monitor on any of the terminal-end computers. The selection of modes at present may be made manually on the CLCU maintenance panel or by instructions sent by any one of the computers on a first-come-first-served basis.

The philosophy of the CLCU control is that any computer may act as a "master" at any time and request that another computer be "slaved" to it. Once the communication link is established, the master-slave relationship disappears for the CLCU, since it will accept further commands from either computer. Since the other computers in the overall system may not gain control of the CLCU once this communication link has been established, it is essential that a CLEAR CLCU instruction be given by one of the active computers when they are terminating their use of the CLCU.

The following specific functions are performed by the CLCU in order to carry out the role described above:

1. Provides full-duplex data transfer capability between Univac-type computers and the half-duplex 32-bit subchannels of the IBM 2909.

2. Satisfies the demand-response interface of all computers so that each computer feels it is in complete control of the data transfer.

3. Converts logic levels of the transmitting computer's I/O interface to levels acceptable to the receiving computer's I/O channel.

4. Provides switching capability under program control to allow a "master" computer to communicate with any one of a number of "slave" computers.

5. Transfers data at the maximum rate allowed between two computers (limited to the speed of the slower computer).

6. Provides a self-test mode to perform checkout of the computer link with either the 360/91 or a Univac-type computer.

This requires that the CLCU be divided into functional areas as shown in Fig. 7. Each computer has a level shifting driver/receiver package that transforms the logical voltage levels used by the individual computer I/O channel to the logic

TABLE 2
CHARACTERISTICS OF TERMINAL COMPUTERS USING CLCU

	Voltage (volts)		*I/O Specs.*	
Computer	*Logical 0*	*Logical 1*	*Word Length (bits)*	*Speed (Max) (words/s)*
Univac 1230	−3	0	30	167 K
Univac 492	−3	0	30	64 K
Honeywell DDP-516 (CFU)	0	+3	16	1000 K

levels used internally to the CLCU (see Table 2). Two multiplexers determine which computers will access the data buffers and control interface blocks which are the heart of the CLCU.

Control lines used by the three Univac or "1230 side" computers are identical as follows:

1. Input Data Request (IDR). Control line voltage raised by the peripheral device (CLCU) to indicate that data are ready to be input by the computer.

2. Output Data Request (ODR). Control line voltage raised by the peripheral device (CLCU) to indicate that it is prepared to accept data from the computer.

3. Input Acknowledge (IA). Control line voltage raised by the computer in response to an IDR to indicate to the peripheral device that the input data lines have been sampled.

4. Output Acknowledge (OA). Control line voltage raised by the computer in response to an ODR to indicate that data is on its output bus ready to be sampled.

5. External Function (EF). Control line voltage raised by computer to indicate to peripheral device that data is on the output bus. An EF differs from an OA in that no response or request by the device is necessary.

6. Interrupt (INT). Control line voltage generated by the peripheral device to cause a jump from the instruction being executed in one program to a given instruction in another program.

The control interface handles the control lines from the IBM and Univac "sides" in order to satisfy both computers. Figure 8 shows a timing diagram for the transfer of two data words from the Univac side to the IBM side. The CLCU responds to the IBM Read Ready by raising an Output Data Request (ODR). When the Univac-side computer returns a data word and an Output Acknowledge (OA), the CLCU sends a Demand to the IBM side. A WC = 0[||], EOR[#] sequence satisfies the IBM computer after all data are transferred. This sequence is typical for transfers in the opposite direction using the IDR, IA, and write ready control lines.

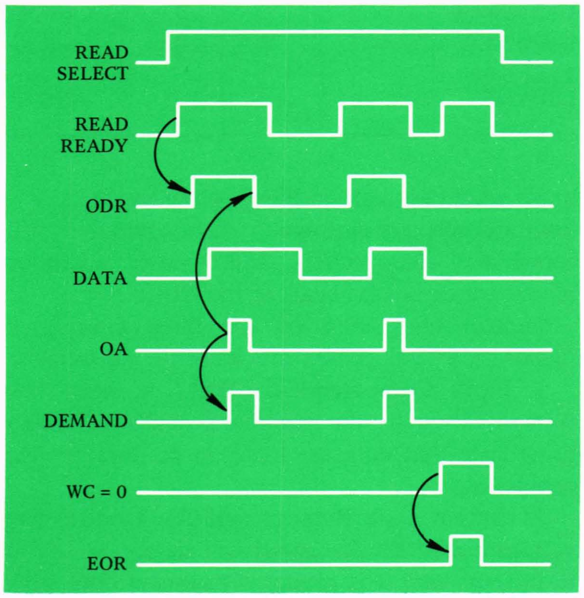

Fig. 8—Typical timing diagram of control lines for data transfer from the IBM side to the Univac side of the CLCU. Actual times depend on which computers are being used (see Table 2).

The CLCU maintains a status register (see Fig. 9) in the control interface that contains among other information:

1. Multiplexer setting on 1230-side (which computer is presently supplying data and control lines).

2. Multiplexer setting on 360/91-side.

3. Whether or not the radar synchronizer is controlling timing (SIMFAR or general purpose).

The contents of this register are manipulated by instructions from the computers. The Univac computers issue specific external functions and the 360/91 uses its device address bits to change the register or read its contents. Since the multiplexers

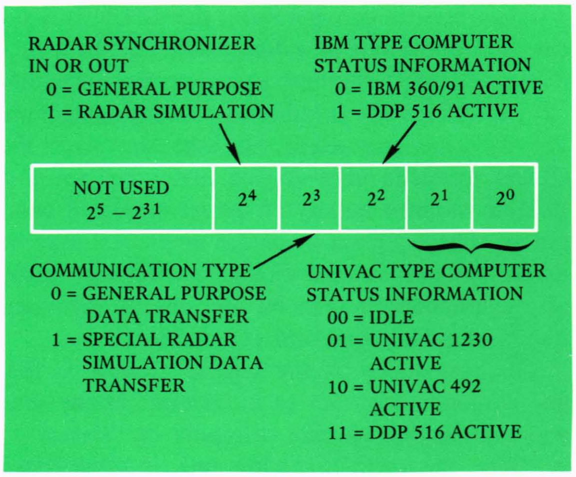

Fig. 9—CLCU status word format.

that select the two communicating computers are driven from the status register, changing the register contents has the effect of changing the CLCU mode. A computer may request the CLCU to send its status word (by the proper EF or OA) in order to determine if the CLCU is available. The status word is always returned through the requesting computer's driver/receiver package even if the CLCU is being used by two other machines.

An important diagnostic feature of the CLCU is its ability to simulate computers. In a simulate mode, the control lines on one side of the control interface may be manipulated from the CLCU panel (Fig. 7). Single cycling (stepping under manual control) or automatic operation is possible. For example, if a problem occurs in the 360/91-to-492 link, the panel may be used to simulate the 360/91 while the 492 is run normally and the operation (including data bits and control lines) is checked. Then the 492 (actually a general "Univac" computer) is simulated while the 360/91 runs. In this way the source of the problem may be isolated. Simulation of two computers (on both sides) is also permitted as a check on the circuitry within the CLCU. This allows quite complete off-line checking of the CLCU.

Control and Format Unit (CFU). The control and format unit (CFU) is similar to the CLCU. As with the CLCU, its basic function is to match up the different I/O control and data between two or more computers. Specifically, the CFU matches the I/O characteristics of the Honeywell

[||] Word Count Equals Zero (WC = 0)—This line is raised by the subchannel when the byte count for the current operation (read or write) has decremented to zero. (Note: Ready also comes high with WC = 0.)

[#] End of Record (EOR)—This line is raised by the I/O device to signal the channel that no more data will be input or output (depending on the current I/O operation).

DDP-516 to those of the 360/91 (2909) or the 1230. This allows the 516 to communicate with the 360/91 through the CLCU by appearing as a 1230 or to communicate with any Univac machine through the CLCU by appearing as a 360/91. This also allows the 516 to communicate (with slight modification) with any of the Univac peripherals directly.

The specific functions performed by the CFU are:

1. Transforms the 516 direct memory access (DMA) high-speed I/O channel voltage levels, pulse shapes, etc. to IBM and Univac requirements.
2. Performs packing and unpacking of 16-bit words and 32-bit words respectively.
3. Satisfies the channel-response (control lines) function between the DMA and the I/O channels of the other computers.
4. Provides data transfer between the 516 and the other computers.
5. Provides for self-test.
6. Provides switching under program control to allow communication to any one of the connected computers via the CLCU.

The CFU is connected to four DMA channels of the DDP-516, any of which may be used for communications with other computers. If a full duplex mode of data transfer is needed, two DMA channels are used, with one dedicated for input and one for output. Since the DDP-516 is a 16-bit word machine, the CFU must pack and unpack data words sent to the 360/91 (32-bit data word) and to the Univac computers (30-bit data word). In the case where the 516 is communicating with the 360/91, the reformatting is straightforward. Between the DDP-516 and the Univac computers, the most significant bit of the DDP-516 word is not used, and two DDP-516 data words form one 30-bit Univac data word.

Hybrid Computer Interface (693). The 693 hybrid computer interface is the controlling device in this computer link just as the CLCU was the controlling device in the all-digital link. The 693 was designed for a minimum number of separate control lines, however, requiring that many control functions be sent as commands on the data lines. One of the more obvious problems, then, is for the 693 to distinguish between command and data words. For example, if one wishes to issue a command to the 680 analog computer, certain control lines must be set; then the first two 16-bit words that follow are interpreted as commands rather than data. (If these control lines are not set, data are transferred to the A/D or from the D/A converters according to the 693 mode already set up by previous control signals. These control signals are generated by software on the 360/91.) The first 16-bit word represents an instruction to the 693 and the second "data" word elaborates the meaning of the instruction.

Finally, a status word and a fault word are associated with the 693 coupler; these words may be input to the 360/91 to enable the digital program to interrogate the condition of the 693-680 units. This is of key importance since the 693 and the 680 analog computer are complex devices, but lack the inherent ability to diagnose their own faults while in on-line operation. Consequently, programs on the 360/91 are used to interpret and act on these hardware conditions.

Simulation Console Interface (1827). The last interface to be mentioned is the SIMCON interface. The main area of concern is the use of the IBM 1827-1892 components as a somewhat special-purpose console. The configuration provides two interesting capabilities—interactive control of an all-digital simulation running on the 360/91 and both analog and digital I/O interfaces to provide interaction between the digital simulation and actual hardware. These hardware capabilities are part of the standard IBM configuration. However, to take advantage of them required a major development of software at APL.

Software

Key Software Design Considerations—Providing the many capabilities represented by the software is a matter of developing the appropriate program and the necessary communication by that program with hardware and other software. It is the intercommunications features and associated error handling and protection that distinguishes this work from applications programming. In this case, both the hardware system and the existing software system must be understood in detail, while for applications programming only a relatively few hard and fast rules imposed by the compiler and/or monitor need be followed. The designers of the System 360 decided that it would be more feasible to manage a multiprogramming computer system without certain aspects of system operation being solely the province of the control program or supervisor itself. Most prominent among these are program switching and access to I/O devices. Problem programs must not be allowed direct access to data on peripheral storage units because of the danger of inadvertent modification of data belonging to other programs or to the monitor. The problem runs, therefore, in "problem state" and any attempt to execute four privileged I/O instructions directly results in an abnormal termination of the problem by the supervisor (monitor).

It was decided to incorporate our new devices into the present 360 supervisor. The problem programs would then request access to these devices through the supervisor just as for the more standard devices such as line printers, disks, etc. The 360 monitor or operating system is made up of a number of separate though intercommunicating programs. One of these is the I/O supervisor (IOS) and is written in assembly language. The IOS is written in such a way as to allow, without modification of existing code, additional subprograms to describe those features of a new device required by the I/O supervisor to identify it and to issue the four privileged I/O commands to it. These subprograms in turn build up appropriate tables known as control blocks which contain the actual data used by the IOS during an I/O operation. The IOS, when appropriately called by a job program through the system supervisor, executes the new subprograms to build the control block tables and then commences I/O to the device.

The question is how the problem program should make the appropriate I/O request to the system supervisor. In our application this is done by an instruction at assembly language level called a supervisory call to the IOS, the argument of which contains information on the I/O operation to be carried out and the identification of the I/O device. Additional information was often also conveyed through control blocks. These control blocks or tables had to be generated or built by the problem programmer at the assembly language level or by a high-level language compiler or by calls to special I/O routines. Because the

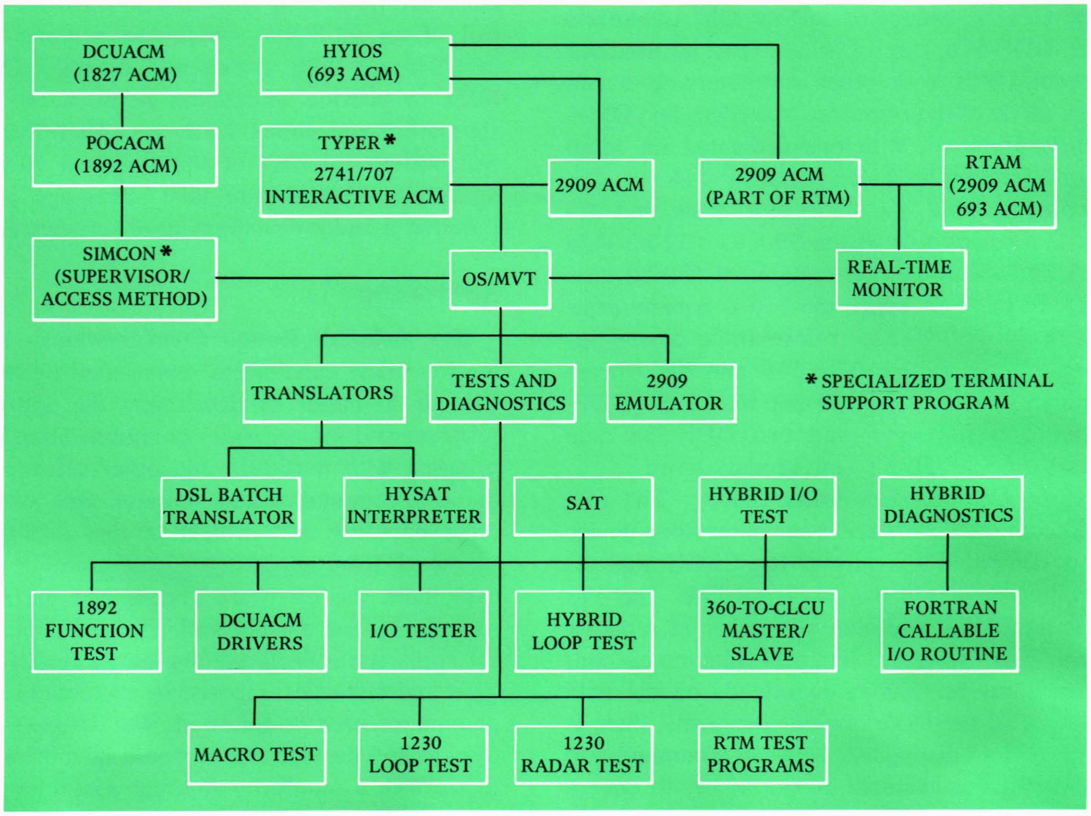

Fig. 10—Link system software additions to OS/MVT.

system control program has to be so extremely defensive about problem programming errors, providing interactive access methods for the peripherals is but half the battle. The interactive user of the simulation-oriented peripherals will invariably arrive at a state of program readiness where he must run the program, linked-to equipment, and try the various options on-line so as to test sections of programming that cannot be tested in any other way. Unless the software specialist has provided for CPU error recovery, the first program bug detected by the supervisor will likely end the test via abnormal termination of the program. Thus, the process of final debugging becomes an extremely slow and frustrating one.

The software to support the links and associated terminals is of six types:

1. Access methods.
2. Monitors.
3. Terminal support programs.
4. Test and diagnostic programs.
5. Translators.
6. Emulators.

The software was required to operate in a pre-existing environment with the following characteristics:

1. Time shared with a variable number of jobs.
2. Batch terminals.
3. Conversational terminals.
4. Graphics terminals.

The new software was to add the following basic capabilities:

1. Real time.
2. Control and data transfer over the new computer-to-computer links.
3. External priority interrupts.
4. Comprehensive link error routines and diagnostics.
5. Substantial operator interaction with the job program from the terminal end of the new links.

The IBM 360/91 normally operates in a time-sharing mode under its operating system with multiprogramming for a variable number of tasks (OS/MVT). This system allows a variety of jobs to be under execution at the same time from batch card readers, both local and remote, from typewriter terminals, and from graphics light pen terminals.

In adding the computer-to-computer links, no modification was allowed to be made to MVT or other OS software such as the remote job entry (RJE) that services all terminals, or the conversational programming system (CPS) that services the typewriter terminals, or the graphics processing system (GPS) that services the light pen terminals, the various translators, and service routines.

Figure 10 is a block diagram showing the software that has been developed to service the computer-to-computer links. In effect, most of this software becomes an expansion of the operating system. With this quantity of software and the fact that most job programs using the link facilities are on the order of 100K bytes, core space becomes an important consideration. Figure 11 shows the memory core allocations for all major systems.

TOTAL: 4096 K BYTES				
1530 K	1000 K	800 K	146 K	620 K
BATCH WORK	INTERACTIVE PROGRAMS	CONVERSATIONAL PROGRAMMING SYSTEM (CPS) AND TIME SHARING OPTION (TSO)	REMOTE JOB ENTRY (RJE)	SYSTEM SUPERVISOR

Fig. 11—Core allocations.

Access Methods—The key type of program necessary for any computer-to-computer link (or any peripheral hardware) is the access method (ACM). This function may be carried out in any number of configurations; it may be a separate package, part of the monitor, or part of an already-existing access method, or it may be left to the job programmer to include in his program. The access method is the software counterpart of the hardware controller or interface. It causes I/O to take place as required by applications programs, handles and generates hardware codes, and contains error-handling features to protect the integrity of the overall system from programmer errors, terminal operator errors, and terminal subsystem hardware errors. Access methods must, therefore, handle the following functions:

Initialization/termination
Input/output
  Read
  Write
  Control
Error handling
  Checks
  Recovery
  Messages

User exits
  User initialization
  User I/O format conversion
  User postprocessing

The initialization/termination functions are generally a single routine that interprets and drives the four basic control functions of the hardware interface, as already described.

The data transfer operation of the interface is driven and controlled in accord with the initialization routine parameters by an I/O routine. This routine handles most of the so-called handshaking and I/O data format operations and also operates in conjunction with an on-line error checking routine.

The error checking routine interprets any hardware flags, while continually checking for logical software problems and data format errors. Some capability is also programmed to provide limited error recovery. The error routine can also provide for a halt or wait of the job program, with appropriate messages and user intervention routines to allow operator-initiated recovery procedures to be used.

It is essential in real-time operations that the ACM be fast since it must operate in series with the real-time hardware and job program. Therefore, the basic program must be simple, providing only a minimum of error checking and recovery.

In general, there will be one ACM program for each interface and terminal making up a link. The functional organization of this software and hardware can be conveniently viewed as a nested arrangement. For instance, the simulation console system requires the console access method (POCACM), which communicates through the digital control unit access method (DCUACM), which communicates with the DCU itself, which then communicates with the console. Figure 12 shows this diagramatically. The other links follow the same pattern, and in fact this view can be extended to cover the various interfaces and related access methods making up the standard 360/91 system.

This same philosophy is also valid for the links having digital computers at either end. However, some modification of the concept, compared to nonstored program interfaces and terminals, is necessary.

Where two digital computers are concerned, there is a need for a set of access methods on both computers that can send data and control

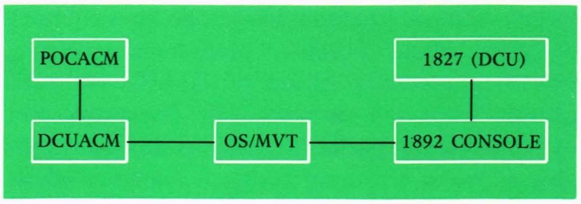

Fig. 12—Nesting of access methods and link hardware.

signals through all interface units. In order to minimize these access requirements, two things have been done: (a) the computer linkage and control unit (CLCU) interface has been made sufficiently intelligent that it can accept instructions for a variety of options that can then be carried out by hardware and (b) the two digital computers are not treated as hardware terminals with fixed capabilities, but are expected to have programs that intercept control and data signals, using these basic functions in whatever way the programmer wishes and the computer hardware allows.

With this approach the requirement for a separate set of access methods for communicating with the CLCU and the digital computers is eliminated, the necessary programming being contained in the master and slave programs or in actual applications (user) programs. This requires the 360/91 programs to communicate through 91 system access methods to the CLCU, but no further. Likewise, 1230 programs communicate only as far as the CLCU. For all practical purposes the CLCU looks to the 91 like a 360 terminal and to the 1230 like a 1230 peripheral. To make this programming relatively independent of any specific knowledge by the user of the interfaces, FORTRAN callable routines are available on the 91. On the Univac computers and the 516, however, the user must still, at this point, deal in assembly or absolute code with the interfaces, although similar interface support packages could be written.

Monitors—Two monitors are required for the overall system, the operating system with multiprogramming for a variable number of tasks (OS/MVT) and the real-time monitor (RTM).

The basic need for any monitor is two-fold: (a) to provide programs that make available to the job programmer the hardware capabilities of a computer without requiring him to understand the architecture, hardware codes, and possibly even the circuit design of the hardware and (b) to manage efficiently the communications or interactions of the programs making up the monitor so that the job programmer does not need to understand the operation of these programs.

In addition to these basic needs, a considerable amount of effort is usually put into hardware and software error-checking routines and diagnostics. This serves several purposes:

1. To protect the monitor and hardware from improper use by job programmers.
2. To protect job programs from each other, which is particularly important in a time-sharing system.
3. To assist job programmers in debugging.
4. To provide the computer operator with knowledge of malfunctions.

OS/MVT. The operating system with multiple, variable number of tasks provides the basic time-sharing or multiprogramming capability associated with the 360/91. It is the 360's monitor. It also manages all of the basic hardware facilities of the 360/91 such as storage, I/O and the associated access methods, accounting, messages to and from the operator, and oversees the operation of such associated software as translators or compilers and service programs such as link edit, sort/merge, data set management, etc. Such sophisticated monitors are major topics in themselves. Since APL has not modified the OS/MVT as a part of the computer program being described, no further discussion will be made here.

RTM. The real-time monitor (RTM) adds to OS the capability of guaranteeing that a real-time job is executed and the associated I/O carried out at given points in time. As necessary, RTM progressively dedicates the CPU and other facilities to maintain real-time operation. In addition, high-speed servicing of external interrupts and timer interrupts is accommodated. The RTM is a set of programs that can be integrated into OS as a system task (resident in core as part of OS) or it can be entered as a program in the batch job stream. Once in execution, RTM establishes control over OS through the following steps:

1. It initializes itself.
2. It issues a supervisory control (SVC) call to a program included in the MVT supervisor at system generation.
3. It then replaces the MVT program status words (PSW's) with its own, temporarily suspending program execution and OS interrupts, if any.
4. It either allows programs under MVT to continue execution or services a real-time job and associated interrupts.

RTM gives first priority to real-time job execution and handling the associated interrupts. Real-time jobs must identify themselves as such by issuing an SVC to RTM. The normal MVT job stream and interrupts continue to be processed in background by MVT. Real-time jobs, while operating under RTM initially, may also give SVC calls to MVT. In this way real-time jobs may utilize all OS programs, the nonreal-time requirements of these jobs being processed under MVT in background.

RTM as a monitor is composed of such usual supervisory programs as task scheduler, supervisory routines, service routines, and access methods. RTM also requires an initialization/termination routine to handle its own initialization and termination of control over OS.

The supervisory routines consist of such programs as the real-time interrupt handler, error detection routines, assignment of priority interrupt levels (PIL's), identification of real-time devices, real-time job initialization, termination procedures, etc. The service routines allow the job programmer access to certain options and capabilities of RTM such as program controlled "waits," and they specify the program to gain control upon the occurrence of an interrupt, reading and writing to a real-time device, use of a peripheral timer, masking and unmasking selected PIL's, etc.

The access methods to real-time devices may be included as a part of RTM. In our case, the 2909 access method has been included. No real-time devices, of course, can be accessed via the access methods that are part of OS/MVT.

Terminal Support Programs—Terminal support programs are closely related to access methods and may have some of the supervisory aspects of monitors. The computer links employ three of these terminal support programs:

1. SIMCON for the Simulation Console.
2. TYPER for the typewriter terminals.
3. FORTRAN callable I/O routines for 360 to CLCU.

These programs work between (or include) the terminal access method and the job program. They are link edited (called in from disk storage and combined) with the job program and are consequently not resident software.

Basically, the terminal support programs are an extension of the terminal access methods. For special terminals having such features as function buttons, displays, and other interactive capabilities, more servicing is required than just read, write, and the four basic control functions of request, initiation, checking, and termination provided by the usual access method. The interface with the job program may also be a relatively high level instead of at the assembly level.

In the case of SIMCON, the terminal program is called on by the job program to activate the simulation console and initialize its various special functions. It puts out a message on the SIMCON nixie lights: "BEGIN." In the course of job program execution and user commands, SIMCON also displays other English-language-like messages. SIMCON handles the interpretation and implementation of all function buttons for mode control, parameter changes, selection of integration method, etc. Finally, SIMCON handles the deactivation of the console and termination of the job program as directed by the user from the console.

TYPER handles similar functions for typewriter terminals and in conjunction with job programs that may or may not be associated with a computer link. TYPER allows a user at a typewriter terminal to communicate with his program through a series of simple high-level language instructions. The application program also may be coded to allow it to output on the typewriter in response to certain calculated conditions or instructions from the user. Originally the 2741's operated through a separate access method, requiring the operator to use several assembly language-type instructions to call various support programs. This was awkward and required some detailed programming. When TYPER was developed, it included the 2741 access method.

The FORTRAN callable routines are a set of routines that may be called by a user program to largely eliminate the need for understanding the operation of the 2909 and the CLCU interfaces. They also eliminate the need for low-level I/O programming. Unlike SIMCON and TYPER, these routines are not managed or supervised as a single package by a program in the package. The user programmer must, therefore, provide this function in his job program. On the other hand, the communication is not interactive and the control is relatively straightforward. This greatly simplifies the situation compared to handling the SIMCON or even the typewriter.

Test and Diagnostic Programs—The greatest number of new programs that were developed in support of the three links fall into the category of diagnostic software. These programs were keys to the successful completion of the overall system and the individual links. For the operation system, these programs are used for maintenance, and in the hybrid case some are also used for automatic setup and checkout of the analog computer.

Most of the diagnostic programs carry out three types of operation: (a) to exercise or drive the system, (b) to detect errors, and (c) to determine the cause of the errors.

In general, the driver will exercise the system in all normal modes of operation, with a wide variety of data and control signals; in some cases, where other software or hardware are supposed to cope with failures, errors are deliberately introduced for testing. This is particularly important in checking out the monitors and access methods.

In most cases a diagnostic program will be designed primarily to test either hardware or software since this generally avoids undue complexity in the programming; it can also minimize the number of subsystems that must be tied up at any one time; and it minimizes CPU time required for any particular kind of check.

The kinds of function that are often implemented by separate programs are as follows:

1. Operator-interactive single-step machine code tests.
2. Testing of all hardware features of the system individually and printing out specific data and errors.
3. Testing programs for software, with emulation of hardware functions.
4. Operational or demonstration programs.

Translators—Translators allow a high-level language to be translated to a lower level language. In general, a translator may be viewed as being (a) a batch operation, (b) an instruction-by-instruction translation with or without immediate execution, or (c) a conversational or user interactive operation.

Developing a translator was not a goal of the computer-to-computer link system since we expected to use those provided by the computer manufacturers. However, the digital simulation language (DSL), which seemed well suited to SIMCON applications, did require some adaptation to the 360/91 system. In addition, it was desirable to expand the DSL translator to accommodate various functions of the SIMCON. This expansion was accomplished by instructions that became "calls" to SIMCON which is then link edited as part of the job program. To meet these secondary objectives, the DSL was modified to provide easy, high-level programming for all jobs to be linked to the SIMCON.

A hybrid static analog test (HYSAT) is a second translator that was required to implement the various hybrid instructions for such functions as: (a) select analog mode, (b) read digital voltmeter, and (c) set potentiometer.

A great many more such operations are also incorporated into HYSAT. This translator is interpretive, allowing instructions to be handled one at a time under operator control, or a source program may be batch compiled.

Emulators—Only one emulator was developed for the APL systems, but it was essential for development of key software in parallel with the procurement and checkout of the IBM 2909 interface. The emulator simulated on the 360/91 the control functions and data transfer of the 2909. This allowed the hybrid I/O system (HYIOS) to be written and checked out in parallel with the 2909.

Wherever hardware has been defined, such emulators can be used to allow software and hardware development to proceed in parallel. In addition to the time advantage of such an approach, it also allows a more effective interchange to take place between software and hardware personnel before final design and implementation. This was most important relative to the 2909 since APL, EAI, and IBM were all involved in software and hardware interfacing at this point in the system, and the 2909 itself was developmental.

Basic Software/Hardware—The preceding discussion of software has outlined in some detail the relationship between the hardware and software. Figure 13 shows more directly how the pieces fit together. The hardware terminal systems are directly controlled by the IOS portion of the system supervisor (OS/MVT or RTM). At this point the software generates hardware codes (known as channel command words or CCW's in IBM terminology). The necessary access methods support special features and allow the applications programmer to communicate with the system supervisor and the IOS via SVC's, and control blocks. The applications programs communicate with the access methods via callable routines or directly compiled I/O instructions.

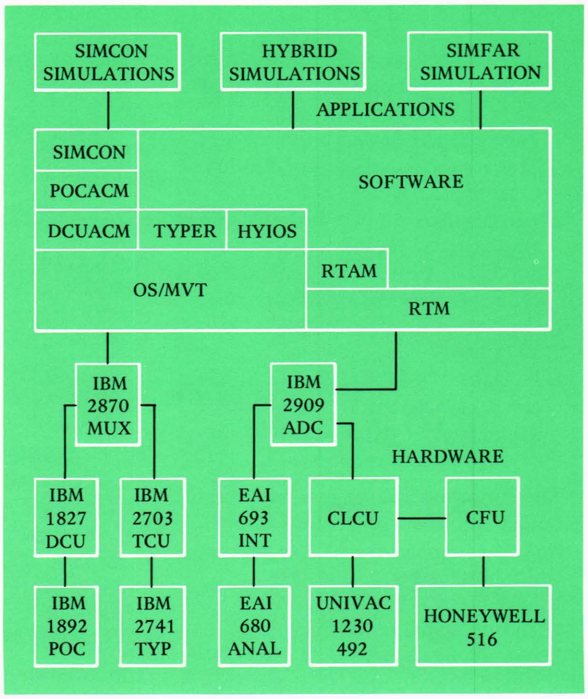

Fig. 13—How the pieces fit together.

Conclusion

All three of the basic links, SIMCON, hybrid computer, and Univac 1230 are in regular use. The hybrid computer has demonstrated the most rapid growth in workload and may soon require more than one 8-hour shift.

Work that remains to be done to improve the systems falls in a variety of categories. For the hybrid computer, fully automated, easy to run, very comprehensive maintenance diagnostics would be of major value in minimizing the maintenance effort. For the multiple computers, using the 1230 link, the ability to time-share the CLCU as well as time-sharing the several CPU's would greatly increase efficiency of the rather powerful computer complex. All three systems would benefit from a better simulation language with a compiler easily extendable with new instructions, and with an optional conversational capability. Also, the development of on-line error recovery and debug facilities for applications programs would greatly enhance the usability of the systems.