Described is an implementation of a three-level computer hierarchy that provides a high degree of performance, availability, and ease of use for laboratory automation applications.

System/7 in a hierarchical laboratory automation system

by H. Cole

Laboratory automation, a subset of sensor-based computer use, is an area of computer application that is characterized by the computer elements servicing user-attached devices. These are usually devices such as instruments, test stands, and terminals that are part of the real-time instrumentation used in carrying out experimentation. The main difference between laboratory automation and sensor-based production systems in general is that each user of laboratory automation is likely to modify his own application from time to time, rather than being a part of a group that uses an application that has been installed by a separate group of systems people. The laboratory automation system described in this paper is currently being used at the IBM Thomas J. Watson Research Center in Yorktown Heights, New York.

Laboratory automation has had a rather strong growth history over the past few years. Professor Perone, of Perdue, in a review article on publications in the field, gives some 194 references to papers describing applications in this field.[1] A selection of other publications is given in the references.[2] One could safely say today that there is no instrument that has not been automated in some way. Its importance is reflected in the fact that the ASTM (American Society for Testing and Materials) has set up a section on laboratory automation.[3] As we briefly review details of a particular laboratory automation system, bear in mind that its importance lies in the fact that, it represents a new level of improvement in the instrumentation available to engineers and scientists for carrying out their exploratory, development, and standard testing work.

Currently, most laboratory automation applications are carried forward (not necessarily carried to completion, as defined later in this paper) by the use of small dedicated processors, or even by stand-alone minisystems. Our own experience with laboratory automation—and especially with its growth—indicates that this approach could lead to a high total investment with a diversity of minisystem capabilities. For the two to three hundred experiments forseen in the Research Center at Yorktown, such a configuration might not be as cost effective as a more systematic approach.

On the basis of our experiences with the IBM 1130 and 1800,[2a,2b,2h] and with laboratory automation in an IBM System/360 Model 67 system,[4] we believe that the system to be described in this paper—which is now in actual use—is a more effective solution. We needed a system on which applications could be readily developed, not a system of collected applications. Before describing the current system, it might be advantageous to elaborate the uses and concepts of laboratory automation in the next two sections.

Uses of laboratory automation

In such fields as research, development, quality control, process control, and product testing—all of which make use of computerized instrumentation, and often make use of the same instruments—the justification given for computer attachment of the instrument is sometimes different for the different applications. Justification usually falls under one or more of the following headings:

- Greater productivity
- Higher precision
- Better calibration and drift detection
- More information extracted from raw data
- More data taken
- Broader range of activities
- New types of analyses
- Results available sooner

Greater productivity is the most straightforward reason for automation because more samples can be examined per unit time at a reduced cost per sample. Professor Brombly, of Yale, has said privately that he believes that in the field of nuclear scattering experiments, the computerized handling of data should permit an experimenter to carry out forty times as many experiments in his professional lifetime as without it. A survey of automated experiments in the IBM Research Laboratory of San Jose,[5] indicates that a three- to five-fold increase in productivity is achiev-

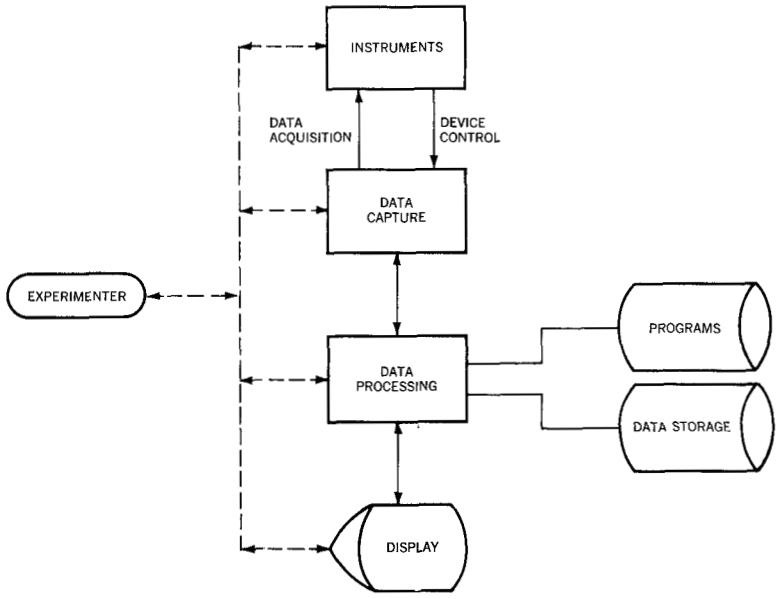
able, without considering the impact of the changes in the work that may take place. Productivity, of course, contains many factors. Our experience in crystallographic data taking[6] has shown a six-fold improvement in the number of structure determinations completed in a year through automation. The experimenters could handle the increased workload since they were freed from the lengthy data taking and could concentrate on the analyses. Part of the productivity increase is based on the use of the computer resource for graphics and report preparation.

Beyond increasing their productivity, engineers and scientists are interested in making more deeply significant measurements or measurements with increased precision. Computerization of the data capture tends to make measurements more reproducible, in the sense that the computer program does the same thing each time and collects the data with fewer bookkeeping errors. When crystal growing procedures were first automated with the intention of learning more about the physics and chemistry of the process, increased growth rates soon resulted. These improvements were attributable to the improved reproducibility or steadiness. Further, with automation, it was easier to calibrate more often by running a standard procedure periodically, thereby detecting and making adjustments for long-term drift in an instrument.

Extracting more information from data being taken and the taking of more data are important factors in the justification of computer-assisted individual experimentation. With data handling by computer, more mathematical analyses are used, better statistical tests applied, more correction terms included, and more of the data contributed to the information extracted. Precision is thus increased through mathematics rather than through hardware. Often no more data are taken, but they are in a more useable form so that more information is extracted. The five parameters that describe a peak—position, height, half width, and background on each side—thus become a fit to a Gaussian probability distribution. Uncertainty is removed from an analyses in which too few data are taken manually by the ease with which more data are taken and analyzed by computer. Such a procedure can confirm or stop investigation of a material being studied.

There are today measuring devices that are so complex or have such high data-producing capability that it makes no sense to use them without computer control and data handling. When a spectrometer with a multichannel analyser can produce 4000 data elements in a few minutes, it seems grossly inappropriate to spend hours dealing with these data manually.

Figure 1 Laboratory automation functional blocks



From a management point of view, the stimulation provided by laboratory automation cannot be overlooked. Although the transition period of computer attachment and use may be frustrating, as is often the case with a new tool, the rethinking of experiments—in the light of why laboratory automation is being used—often leads to new analyses.

Rethinking in terms of results being available sooner leads to the need to clarify what is involved in obtaining answers sooner. In the next section, where we discuss concepts of laboratory automation, we go into this question in some detail. The resultant of the benefits of laboratory automation—increased number of studies, increased precision, and new analyses—is to give the investigator the potential to enhance the quality of his experimental work. What investigator when he is offered the capability of doing five times as many tests in the next year will chose to do them at his present rate?

Concepts of laboratory automation

We conceptionalize laboratory automation as that part of an experimental activity that returns an answer—beyond mere data collecting—from an exploratory run, a development cycle, or a series of standard tests under computer control and data processing. In other words laboratory automation is computer assisted experimental research.

Figure 1 illustrates the major functional task of laboratory automation. Whether for control or data recording or both, data from the experiment or instrument must be captured. This requires both receiving and transmitting channels for input/output (I/O) between the instrument and the data capture device, the first computing function. Often one begins with a minimum of instrument control, and emphasizes, rather, data acquisition. However, the more of the major steps of an experiment that the computer can control the easier it is for the instrument–computer system to remain synchronized. At the computer-instrument interface we confront the whole engineering problem of attaching two pieces of electronic equpment together, with the usual concerns for such things as having the right signal levels and avoidance of noise (which might give spurious signals). The interface is sometimes called *instrument automation.* Instrumentation on the market today is often designed for computer attachment. It may also have built-in logic that performs a well-defined procedure for data capture, and includes initial steps in data handling, such as averaging and display.

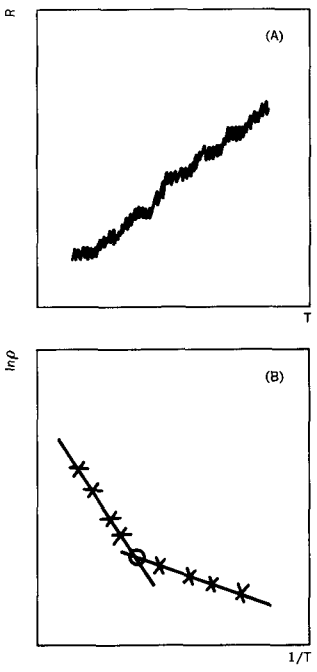
A major problem in attaching instruments in general is the I/O programming for correct data acquisition and instrument control, not because the correct procedures are not understood, but because the instrument often does not behave in the expected way. Thus programming for possible error conditions, taking into account the real-time synchronization requirements, can lead to several iterations of programming.

To ease the startup of laboratory automation, it is sometimes necessary to dissuade a new user from his misplaced faith in excessive data rates and unrealistic precision. Larger gains are to be made in the data processing step than in the data capture step.

In the data processing step, analysis is done and information is extracted. This is also where new analytical techniques are developed and where the user is most inventive. It is here also that questions of program writing and storage, and of long-term data storage and retrieval are answered. The important considerations in the data processing step are interactive data processing, high-level languages, file management, and storage.

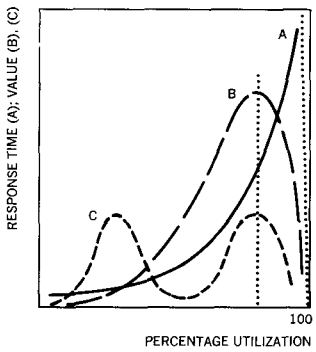
For example, the electrical resistance of a sample as a function of temperature may have been measured correctly and with proper precision in the data capture phase. The data processing steps may consist of calculating the natural logarithm of the resistivity as a function of the reciprocal of the absolute temperature. Straightline segments are then fitted to the calculated points. The experimenter then notes where the straight lines intersect. As illustrated in Figure 2, the experimenter may see an unexpected relationship because the calculated points are infor-

Figure 2 A. Data display
B. Graphic information display



mation instead of data. From the time the sample was first put into the furnace the experimenter has not needed to think further about the sample. His main objective has been to reach the data processing step as quickly as possible so as to take advantage of his cognitive skills. Again, there is no point in having information in the computer; it is only useful if it is returned to the experimenter as soon as possible, and in a form in which he can easily absorb it. Graphic output, rather than a tabular print out aids cognition. Graphic output in laboratory automation is just as important as data capture. Nevertheless, an experimenter uses all the functional blocks of Figure 1 in his laboratory automation system.

Figure 3 Value of a computing resource and response time as a function of percentage utilization



Distributed systems

Implementing the laboratory automation shown in Figure 1 implies a wide range of choice of components. For example, minisystems attached to each instrument may perform data capture; punched paper tape may be taken to the computing center and submitted with an analysis program for batch calculations; and the data printout may then be taken and hand plotted. With an IBM 1800 computer,[2a] we were able to provide these basic functions for about twenty laboratories in one central processing unit on an interleaved basis as follows: data capture through service interruption routines; data processing through FORTRAN programs in partitions; and storage scope displays in user locations driven by interactive display routines.

distributed system

Neither a set of unrelated customized installations nor a single large central processing unit is as attractive as a distributed system approach because each user needs all the function previously discussed. Each user, however, needs a different set of dedicated resources. General arguments for distributed systems are illustrated in Figures 3 and 4. In Figure 3, adapted from a discussion by D. Streeter,[7] we see that the general response time for responding to a new request in a computer system rises as a function of percentage of system utilization. This is indicated by curve A in Figure 3. The overall value of a resource, when all value factors are considered, is shown by curve B, for a larger system. Computing center managers need to be aware, for example, that pushing utilization from 80% to 90% may halve the value of the resource to the general user population. (By analogy, if all the lanes of a highway are full, the highway is of less value to the general population.).

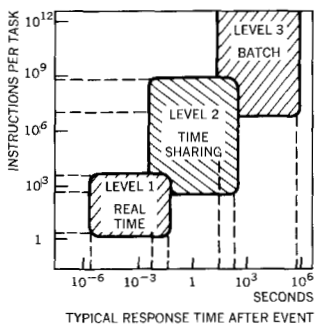
For a conversational computing installation, to which curve B can apply, there are also laboratory automation users who can fit

into this general population, as we discuss more fully in the following section. Most laboratory automation applications up to now, however, have had more severe real-time needs. Curve C of Figure 3 is a value curve for realtime users. The peak in value at low percentage utilization reflects the fact that this computer resource is part of the instrumentation, and the value lies in running the test, more than in running the computer. The computer must be ready when the test is ready, which can only be accomplished if the computer is largely idle. Thus, availability, timing, stability, and personal control are of greater value than high percentage of utilization.

comparative systems

On the other hand, the processing of the data can often be done within the responsiveness of a shared system. This fact is manifested by the second peak in curve C. The cost of a resource with low utilization, even though necessary, should be minimized. The general arguments of Figure 3 lead to the idea of coupled systems, each of which can be cost-effective for the function it is performing. The concept of regions of cost-effectiveness of system types is illustrated in Figure 4. Here the expected response time after an event (say an interruption) is plotted against the number of instructions to be executed in the task that responds to the event. Since every computer operates at a known maximum speed, it is usually impossible to execute instructions in response to an event that requires almost as much time as the repetition of the event itself. Figure 4 shows regions of real-time, time-sharing, and batch operations that are generally cost-effective today. It is usually not a cost-effective policy to keep batch or time-shared systems at the state of responsiveness needed for interactive operation. On the other hand, there are indications that small real-time systems, when not being used for data capture, can be used to carry out large calculations. Such use, however, becomes ineffective for the larger jobs, when measured against the ease of using larger systems.

Figure 4 Cost-effective regions of computing classes



Of course, the linking together of the capabilities shown in Figure 4 depends strongly on communication facilities, responsiveness of the different levels, and local engineering constraints. With low-speed telephone lines or sluggish response, more compute power—especially storage—is needed at each device site. The installation at the device site may well be a turn-key mini-system application. In a laboratory automation environment, however, it is unlikely that all requirements can be satisfied at the device site, or that change at this level should not be expected. Therefore ease of attachment to the next level, when needed, and ease of change, should be important choice factors. The primary area in which Research Center users at Yorktown make changes is in their analysis programs.

Real-time computing

The essence of laboratory data capture and experimental control automation is that these functions are integrated into a computing system. If the computing elements cannot respond in time, the whole operation may have to be repeated. Much more is at stake than that the experimenter has to return an hour later for his data print-out. For a technical point of view, he also expects stability and responsiveness in a system of high availability that is fast enough to do the job and sufficiently rugged to perform the operations that are expected of it. Psychologically, since the experimenter is responsible for the rest of the instrumentation, he feels a need for authority over the computer as well.

small local computer

Small individual processors, which often cost no more than other complex laboratory instruments, have been seen as satisfying this need and capturing data. They provide a sense of control, i.e., dedication to the task under exclusive control of the experimenter. A terminal that is connected to a larger system also offers a sense of control that batch processing does not. On the other hand, using a small system has meant being responsible for it and learning how to use it, which has often required a long-term investment of time. Ease of programming small dedicated systems has not been readily available. Also multiple data processing systems have to be used to do the whole analysis because of the dedicated nature of data-capture systems.

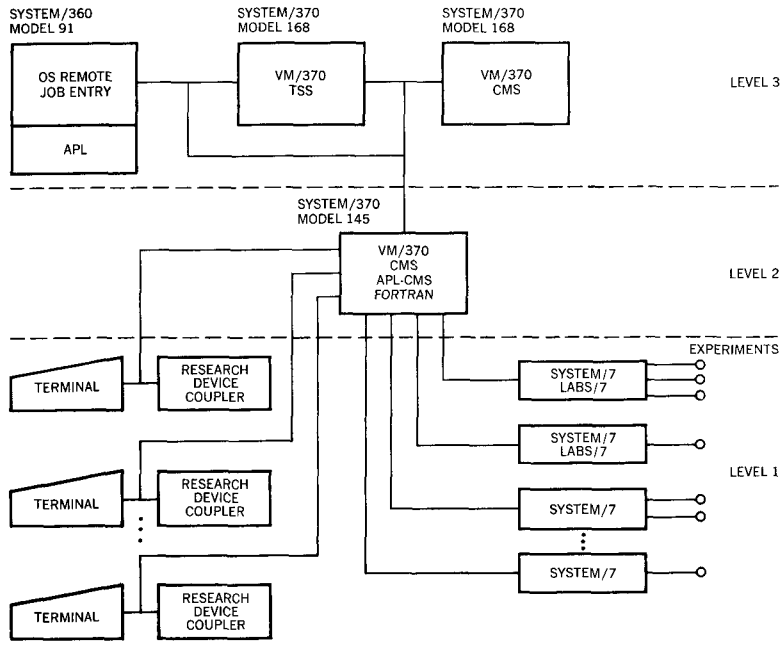
large shared system

The use of a large shared system often offers the advantages of support programs. There is also concern for system down times (system availability). Since all hardware—including other parts of the instrumentation—are down at times, the key parameter is the productivity value of running time compared to the loss value when not running. If down time is very costly, as it is in much of process control, then dedicated systems provide the redundancy necessary for high availability. For much research and development, a reasonable amount of down time can be tolerated. Therefore, for research and development, large-system shared resources offer efficiency. Further, in a laboratory with many potential applications, a hierarchy of computing systems offers a way of tailoring the ratio of dedicated to shared computing needed for each application.

A hierarchical system in a research application

The hierarchical system at the Research Center at Yorktown from the top down (or from batch processing out to data collection and display in the experimenter's laboratory) is schematically represented in Figure 5. A System/360 Model 91, running OS batch, Remote Job Entry (RJE), and APL is connected to a System/370

Figure 5 Research Center laboratory automation system



Model 168 running TSS which, in turn, is connected to a System/370 Model 168 running VM/370-CMS. This constitutes the computing center, which exists independently of the laboratory automation application. The computing center is primarily conversationally oriented toward supplying APL and VM/370-CMS services to several hundred terminals. Each system is connected to the other in such a way that jobs can be easily moved around. In the rest of this paper, VM/370-CMS is refered to simply as CMS.

Within the computing center, the laboratory automation system consists of a System/370 Model 145 host that operates under CMS, which supplies FORTRAN and APL service to terminals with Research Device Couplers listed in Table 1, and supports a number of IBM System/7s, each of which primarily uses the Installed User Program LABS/7 programming system[8]. The System/370 Model 145 is also connected to the network of other systems in the computing center. Table 1 summarizes the three-level hierarchy of computer resources available for laboratory automation at the Research Center at Yorktown.

The essential feature of this hierarchy for laboratory automation is that an activity on Level 1 initiates responses from Levels 2 and 3. That is, the experimenter at his site commands the whole hierarchy. He uses the hierarchy as though it is one system. The experimenter at the operator station of a System/7 views the

Table 1 A three-level hierarchy for laboratory automation

Level	*Type of service*	*Activity*	*System*
1	Time critical real time	Device control Data acquisition Data management Limited processing Display driver	System/7 with LABS/7
1	Low demand real time	Data capture Data display	Shared System/7 or Research Device Coupler
2	Interactive time sharing	Program writing Interactive data processing File management Display generation Storage and retrieval	System/370 Model 145 VM/370-CMS APL FORTRAN
3	Batch	Compilations Assemblies Large-scale calculations	System/370 VM/370 Batch (or transmit to System/360 Model 91)

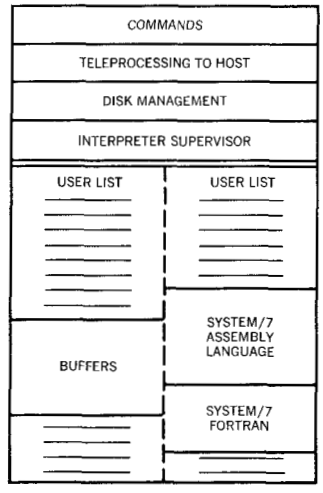
System/370 Model 145 and the System/360 Model 91 as part of his System/7. Neither the Model 91 nor the Model 145 can interact with the System/7 without having been requested. Conceptually, this is quite different from a computer network that is interconnected through a large central computer. In this research laboratory automation system, the small computer controls the hierarchy.

level 1: real time

The System/7 is an event-driven real-time system with an interruption structure that responds to the requests of attached devices. System/7 has adequate storage, good disk facilities, internal timers, and teleprocessing capability. For laboratory automation applications, the System/7 has sufficient front-end capabilities—digital in/digital out, analog in/analog out, and process interruption—that permit the attachment of instruments, devices, and displays. Since the System/7 has more capabilities than some of our projects require on a dedicated basis, the availability of a small operating system would ease the sharing of the System/7 among several independent concurrent users. This situation motivated the development of the LABS/7 operating system at the IBM Research Laboratory in San Jose. LABS/7 is now an installed user program.[8]

In brief, LABS/7 is a small monitor that permits multitasking and multiprogramming. This monitor also incorporates a programming language that permits an experimenter or supporting personnel to program data capture easily. The language also accepts blocks of Assembly/7 or FORTRAN/7 code. The LABS/7 user need not know how to program the System/7 to make data capture and display possible. The LABS/7 supervisor also provides disk and teleprocessing support to the Model 145-CMS host system. These features, shown schematically in Figure 6, make LABS/7 a very useful component of our hierarchical laboratory automation system.

Figure 6 LABS/7 structure



In Level 1 systems, there are occasional user interferences that are acceptable, especially on the smaller system where there is good contact between the individuals involved. However, if interferences occur often, one or more of the users may need more dedicated resources.

Most of the System/7s currently used for our Yorktown Research automation have 16K bytes of storage, a multifunction module for instrument attachment, two disk units, and teleprocessing facilities. LABS/7, used in a straightforward way provides high-speed response, handles the sensor-based input/output operation, handles disk and main storage management and teleprocessing for four or five concurrent users. A library of LABS/7 programs can be built up and saved on disk for each user. These programs are relocatable in storage assigned by the supervisor during program loading. Since LABS/7 is an interpretative operation, the programs—unless they contain blocks of Assembly/7 or FORTRAN/7—are short lists.

Users who share the System/7 outside the immediate vicinity of the machine may require additional terminals. Besides graphic output devices, individual requirements commonly include the support of such things as switches and lights. A display-keyboard terminal is another frequently used laboratory input/output device that is supported by LABS/7. Most of our screen display devices for graphics are storage scopes. Hard copies of plots are rarely needed because the data are stored in the computer, but they may be displayed and photographed for reports.

Experimenters who share a System/7 this way have a real-time processor available and a programming language that is reasonably easy to learn and use. Because System/7 is part of the laboratory automation hierarchy, program writing is done on level 2, the time-sharing level. Experimenters who have applications that demand a more customized programming may also use level 2 to support ASM/7 or other selected programming procedures.

level 2: time sharing

Experience with laboratory automation on a CP-67-CMS system[4] has pointed the way to many attractive features of a VM/370-CMS host for the level 2, time-sharing level in the hierarchy. VM/370 is a time slicing system, in which virtual storage gives each user an effectively large virtual machine. CMS provides such facilities as language processors, file management, and an editor. Under VM/370-CMS, procedures can be defined that can be called upon later by a single command. Also small special virtual machines can be created to handle special jobs.

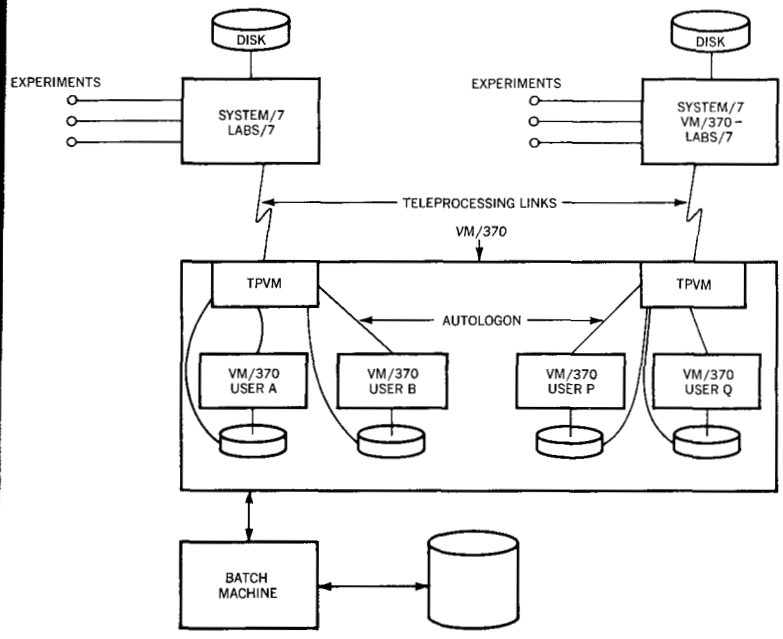
One such small machine, called TPVM, handles the communication line between the Model 145 and the System/7. This machine is shown in Figure 7. The TPVM remains in a listening state for the communication line, and uses one page of main storage. Functionally, the TPVM is transparent to the experimenter at the System/7, and stands ready to enable the level 2 tasks asked for by the experimenter. Since the System/7 is shared, the first task of TPVM is to identify the user. It then accesses his CMS file space, or accesses a procedure called AUTOLOGON (which is discussed in Reference 9). AUTOLOGON logs on a previously created CMS machine. Through TPVM, the user (or experimenter) at his station or terminal has available most of the CMS facilities. Thus a user can perform data capture with an executing program in his System/7, while writing a new LABS/7 program or CMS FORTRAN program on the Model 145 system. Earlier, he might have transmitted a large-scale calculation or an assembler job to the Model 91 for execution. At any one moment, an experimenter might have jobs going on all three levels. (Of course, he might not be running at all, which is statistically the more likely case.) In that case, he is not occupying any large computer resource, except file space and the TPVM machine. The System/7 may be idle or it may be used by other experimenters who share it. The resources of the level 2 system are available to the general set of users for laboratory automation or otherwise.

Since most of the System/7s are shared, experimenters who use any one System/7 are protected from each other by a user profile at the level at which the program assembling is being done. (Not all System/7s are shared, especially those used for such operations as crystal growing that demand a fully dedicated resource.)

Through VM/370 and LABS/7 (as shown in Figure 7), the System/7 experimenter can do one or more of the following operations more or less concurrently:

- Write data on his VM/370 disk space.
- Bring in a file.
- Request the activation of a previously written program.

Figure 7 VM/370—LABS/7 system structure



- Transmit a job to a batch processor.
- Inquire about the status of a job.
- Write a new program or modify an old program.
- Interact with a data display or with calculated results.

TPVM is discussed further in Reference 10.

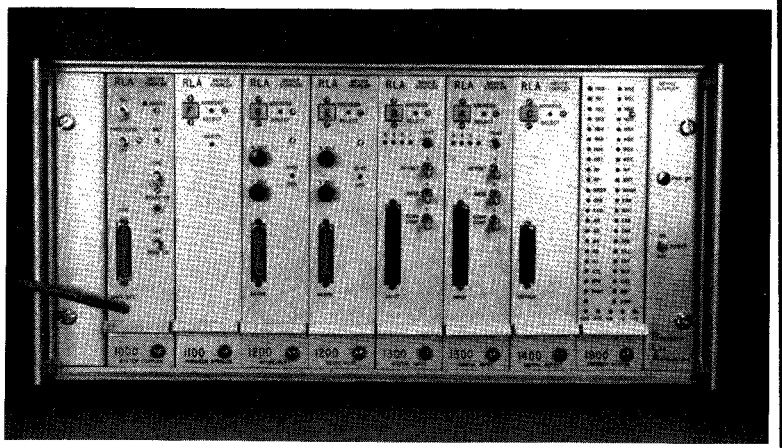
communication

The laboratory automation system discussed here has been implemented in the IBM Research Center at Yorktown, a single building. Thus it is possible to run cables through conduits and hardwire the machines together. The System/7s have been operating at 50K baud, using the start-stop 50K baud ports in the IBM 2701 Data Adapter Units on the Model 145. A sensor based adapter is being installed that can connect up to sixty-four System/7s by cable as far as a mile away that are transmitting at a total rate of 277K bytes per second. The CMS system has a sufficient response time so that a System/7 may use it during the program execution phase. This configuration allows data processing on level 2, so that little data processing is done in the System/7s. For conversational terminal support, the IBM 2703 or 3705 on the Model 145 provides for a 134.5 baud switched line or dedicated 600 baud and 1200 baud lines.

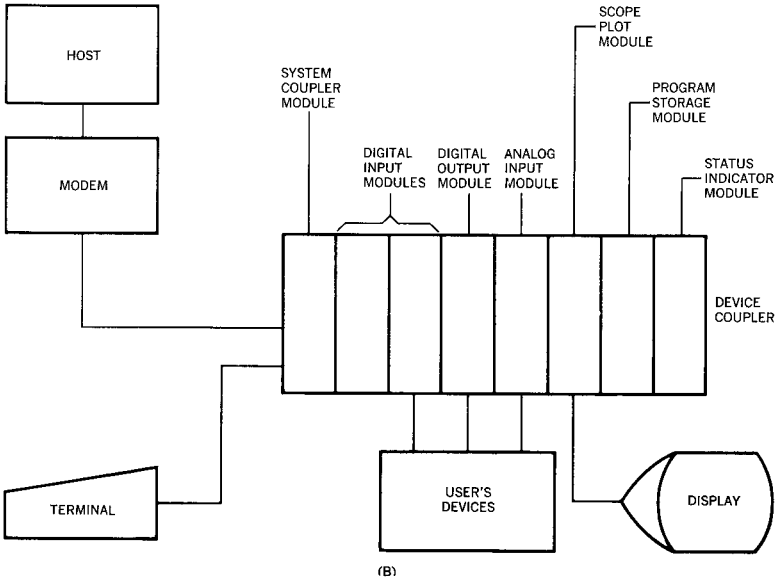
level 2: device coupler

Despite the adequacy of the hierarchy, potential laboratory automation users who could benefit from its use have been reluctant to make the commitment to the learning time required, even

Figure 8 A. Photograph of device coupler
B. Device coupler in laboratory automation application



(A)



(B)

though CMS-LABS/7 system have greatly reduced the effort. Often, however, potential users already know one of the conversational systems, such as APL, or FORTRAN under CMS. Conversational system use has been increasing because it is a part of a movement toward remote data entry, data base activity, and interactive computing.

It is appropriate, therefore, to build on this growing skill and do laboratory automation at terminals. A large number of sensor-based activities can be carried out within the responsiveness of a time-sharing system. Most instruments that today log their data

on paper using strip chart recorders or x-y plotters operate at speeds no faster than terminal operations.

Needed, of course, is a I/O hardware that can act in conjunction with a terminal, so that devices may be attached and used at terminal location as though they are part of the terminal operation.

device coupler

Experimental hardware called the Research Device Coupler, has been created to fulfill this need. The Device Coupler is a rack-mounted box of I/O-type function modules that permit experimental devices to be attached to a computer port. The experimenter's device may itself be a small computer, or individual pieces of instrumentation. The computer port most often used is a teleprocessing communication line so the device coupler appears to the time-sharing system as a terminal such as the IBM 2741 or the Communicating Magnetic Card Selectric Typewriter (CMC), the system does not sense the presence of the device coupler.

The main advantages from the experimenter's point of view—if he is already using a language at a terminal—are the following: doing data capture, data processing, and display with one high-level language. The Device Coupler is a low cost way of starting laboratory automation. The disadvantages are that it must operate at the speeds, responsiveness, and stability of the overall time-sharing system. We believe that the advantages to the experimenter warrant a time-sharing system for this use alone. Another advantage is that little more need be done than to run a stable time-sharing service. The Device Coupler is programmed by sending it messages. No change is needed in the system software.

In its modularity, the Device Coupler resembles such current devices as NIM or CAMAC, discussed in Reference 2d. The function modules of the Device Coupler, however, are a departure in architecture from such modules and from the front end of the System/7 and IBM 1800. A common configuration for the Device Coupler use is shown in Figure 8. One module of the set is the system coupler module. In this case, the Device Coupler acts as a start/stop asynchronous communication module. The CMC plugs into the system coupler rather than in the modem. The system coupler plugs into the regular telephone data set. In this configuration, the Device Coupler and the terminal use the same communication line. The Device Coupler is activated by messages that are specific to it. The messages to the Device Coupler request the other modules in the set to perform various I/O operations. The other modules perform such operations as digitial-in/digital-out, analog-in/analog-out (display driver), and operations store.

digital input function module

The digital input function module illustrates the concept. As input, this module can transfer up to sixteen binary coded decimal or hexadecimal characters as many times as requested. The important point, however, is that this module also contains a digital output point and a process interruption point. This reflects the fact that most digital devices, such as counters and digital voltmeters, usually need a start pulse, after which a wait is required before they have generated the digital information to be read. A simple message initiates a set of I/O activities. Most modules are not purely input or output but a mixture.

The digital input function module has attachment features such as polarity selection for the true state of the signal lines and interruption line. Decode selection matches the BCD code of the experimental device, and voltage off-set potentiometers are available to match signal levels. Thus the attachment of users' devices has been made as simple as possible.

In an alternative configuration, two computer ports are used, and the Device Coupler acts as an independent terminal. The terminal normally operates through a separate line at high speed. This is easily accomplished by an experimenter who is using CMS with FORTRAN by attaching the Device Coupler port through his program. The Device Coupler can also be used offline in a standalone data capture configuration.

A user, who has no terminal experience, has two choices in the Research Center laboratory automation system at Yorktown. He may learn to use a Device Coupler and go directly to level 2. Or he may learn CMS-LABS/7 and use a dedicated or shared System/7. Of course, high data rate, critical timing, or greater stability may require the use of a System/7.

We expect the user to carry on by himself after an initial learning period. Data analysis is clearly his own responsibility. Beyond that the available facilities make it possible for a small number of support people to aid the development, modification, and teaching of laboratory automation applications.

level 3: batch

Batch operations are understood in the Research Center of Yorktown laboratory automation hierarchy as those that the user passes on to the system to be done as soon as possible, but within the system priorities. Batch operations are typically compilations, assemblies, text printouts, large-scale calculations, and extensive file searches. With a CMS system, a batch VM/370 machine can be created to which these jobs are sent. Through the computing center network, these jobs are transferred to Model 91 and entered into the OS job stream. Using CMS procedures, which the experimenter can invoke, he may not be aware that the information transfer is taking place. As a CMS-LABS/7 user,

the experimenter views this resource as another part of his System/7. A further discussion of the Research Center computer network at Yorktown is given in Reference 11.

Continuing evolution of laboratory automation

The laboratory automation system in the Research Center at Yorktown is continuing to evolve. Ease of programming must still be improved. The goal is that a new user be able to start with a single part of his application without doing anything beyond operating function keys, much as he does now when he uses a hand held calculator. It is likely that the next step should include calculator functions at the local site to ease the user's transition period. As computing becomes more economical, special functions may be automated locally. Economy of scale is expected to continue through hierarchical system support of many basic applications.

Concluding remarks

The enhancement of research and development, product test, and process control by computer assisted data capture, analysis, storage, and display have been well demonstrated. Experience in laboratory automation at the IBM Research Center at Yorktown indicates that more users can take advantage of enhanced instrumentation and analytic capabilities as computing power continues to become more economical. A hierarchy of computer elements offers an opportunity to provide these resources with a degree of compatibility that minimizes the support required.

ACKNOWLEDGMENTS

Among the many people who have contributed to the laboratory automation systems discussed in this paper, the author particularly acknowledges the contributions of A. L. Bednowitz, A. A. Guido, L. B. Kreighbaum, J. I. Fortoul, H. Hulzsch, N. C. Hien, R. V. Dobransky, R. W. Martin, D. L. Raimondi, and P. M Grant. He also wishes to thank F. W. Chambers, H. M. Gladney, Y. Okaya, and H. Freitag.

CITED REFERENCES

1. S. P. Perone, "Computer applications in the chemistry laboratory," *Analytical Chemistry* **43**, 1288 (1971).
2a. H. Cole, "Growth path for computers in automated analysis," *Chemical Analysis of the Environment and Other Modern Techniques*, Plenum Publishing Company, New York, New York (1973).
2b. J. D. Swalen, et al., *Laboratory Automation*, RJ746, may be obtained from the IBM Research Laboratory, Monterey and Cottle Roads, San Jose, California 95193.
2c. *IBM Journal of Research and Development*, **13**, 1 (1969), special issue on laboratory automation.

2d. J. Birnbaum and M. W. Sacks, "Computers and nuclear physics," *Physics Today* **21,** 43 (1968).
2e. J. C. Bran, et al., "ARIEL, on-line experimental data acquisition and reduction," *IEEE Transactions on Nuclear Science* **NS-19,** 654 (1972).
2f. I. N. Houton and A. Lewis, "CAMAC in real-time computer systems," *IEEE Transactions on Nuclear Science* **NS-19,** 480 (1972).
2g. S. R. Smith, et al., "Intercomputer communication in real-time control systems," *IEEE Transactions on Nuclear Science* **NS-20,** 538 (1973).
2h. A. L. Bednowitz and F. W. Chambers, *A Literal User Language for Lab Automation*, RC 2755, may be obtained from the IBM Research Center, P. O. Box 218, Yorktown Heights, New York 10598.
2i. A. Segmuller and H. Cole, "Procedure to run an automated micro-densitometer on a shared computer system," *Advances in X-Ray Analysis* **14,** Plenum Publishing Company, New York, New York (1971).
2j. H. Cole, Y. Okaya, and F. W. Chambers, "A computer controlled x-ray diffractometer," *Review of Scientific Instruments* **34,** 872 (1963).
2k. M. A. Frisch and W. Reuter, "Automated evaluation of photographically recorded mass spectra," *Analytical Chemistry* **45,** 1889 (1973).
2l. F. Holtzberg and M. A. Frisch, "Solid-vapor equilibrium in the Sm-S system," *Revue de Chimie Minerale* **10,** 355 (1973).
3. J. W. Frazer, "A systems approach for the laboratory is necessary," American Society for Testing Materials, ASTM E-31, *Materials Research and Standards* **12** (1972).
4. A. A. Guido, *Laboratory Automation in a Virtual Machine Environment*, RC 3917, may be obtained from the IBM Research Center, P. O. Box 218, Yorktown Heights, New York 10598.
5. H. D. Plotzender, et al., *A Study of Productivity and Cost Associated with Automation*, RJ1328, may be obtained from the IBM Research Laboratory, Monterey and Cottle Roads, San Jose, California 95193.
6. Y. Okaya, "Interactive aspects of crystal structure analysis," *IBM Systems Journal* **7,** 3 and 4, 322–330 (1968).
7. D. S. Streeter, "Issues of management," *The Scientific Process and the Computer* John Wiley and Sons, New York, New York (1974).
8. LABS/7 supervisor and language were developed by a group under the direction of R. W. Martin at the IBM Laboratory, San Jose, California. Reports RJ 1184, RJ 1185, RJ 1186 and RJ 1187 may be obtained from that laboratory, whose address is Monterey and Cottle Roads, San Jose, California 95193. Installed user Program Manuals SH20-1363 and SH20-1364 may be obtained from the IBM Data Processing Division, 1133 Westchester Avenue, White Plains, New York 10604.
9. AUTOLOGON is a VM/370 procedure developed by M. Hefler that permits one VM/370-CMS machine to log on another VM/370-CMS machine as though it were a user at a terminal. Further information may be obtained from the author, H. Cole, at the IBM Research Center, P. O. Box 218, Yorktown Heights, New York 10598.
10. H. Hultzsch, et al., *Laboratory Automation in a Novel Computer Hierarchy*, RC4714, may be obtained from the IBM Research Center, P. O. Box 218, Yorktown Heights, New York 10598.
11. J. Myer and R. Nachbar, *CP67-OS/360 Network Link*, RC4113, may be obtained from the IBM Research Center, P. O. Box 218, Yorktown Heights, New York 10598.