JOURNAL OF COMPUTATIONAL PHYSICS 2, 255–273 (1968)

A Simple Time-Sharing Monitor System for Laboratory Automation

H. M. GLADNEY

IBM San Jose Research Laboratory, San Jose, California 95114

ABSTRACT

The requirements for time-sharing a computer to control and monitor multiple experiments in a solid state physics and physical chemistry laboratory are discussed. A monitor system, which has been implemented on the IBM 1800, to control multiple spectroscopic experiments and collect their data is described. It provides flexible analog and digital input/output routines which, using a multilevel interrupt system, operate in a background mode once they have been initiated from a foreground program, and a priority queuing system to sequence these foreground programs. In addition, provision is made for the utilization of the computer on problems not associated with any connected experiment. A simplified sample application is presented.

INTRODUCTION

Research in solid state physics, physical chemistry, and analytical chemistry has been sharply influenced in the last decade by the introduction of instruments capable of collecting relatively large amounts of data. Very often, the rather sophisticated and expensive instruments involved remain idle for a large fraction of their time because the researchers are engaged in simple but tedious tasks of preliminary data reduction and amelioration to separate the meaningful data from instrumental corrections and noise. It has long been advocated by numerous voices that these tasks are readily accomplished on a digital computer and that in addition one can envision quite sophisticated feedback and control. This paper describes rudiments of a simple monitor system which is sufficiently powerful to accomplish most of these tasks for a chemical physics laboratory and which, moreover, relieves the experimenter of almost all the tricky interface programming. The system has been designed to take full advantage of the flexibility offered by a moderate sized process control computer.

Before detailed description of this system is undertaken, it is pertinent to describe the laboratory for which it has been designed and the problems with which the computer will be required to deal. Most of the experiments which are being tied to the computer occur in the physics departments of this laboratory. These include

581/2/3-4

spectroscopy in a number of energy ranges (electron paramagnetic resonance, far-infrared Fourier-transform spectroscopy, infrared spectroscopy, visible-absorption and fluorescence spectroscopy and vacuum ultraviolet reflection spectroscopy), studies of electrical transport in photo- and semiconductors, single crystal and powder x-ray diffraction, and atomic and molecular cross section studies in an ultra-high vacuum manometric experiment. The applications of instrument control in our chemistry department are those of a conventional analytical chemistry laboratory (infrared spectroscopy, vapor phase chromotography, potentiometric studies). In addition, the laboratory has a very active study of industrial process control which involves, inter alia, studies in long distance communication between computers and simulation of processes on an analog computer connected to the digital computer. It is desirable that none of these projects conflict with any other, within the data processing capabilities of the computer employed (i.e., no application should be in a pending state at any time that the central processing unit is idle).

Multiple experiments running concurrently require either multiple small data processing systems or time-sharing in moderate to large data processing systems. Since for many applications the small computers will at one point or another require service from a more powerful machine, the former approach does not in fact evade the time-sharing problem but merely postpones it. The general time-sharing problem in which the number of users is potentially very large and in which "instantaneous" service is to be given to each user, is a highly complex problem which is being actively studied in a number of centers. To date no simple comprehensive system which can be transferred from machine to machine, mutatis mutandis, has appeared. If laboratory automation is to be made feasible, it is necessary that the time-sharing problem be brought to manageable proportions.

Our approach to this problem differs from those employed in other laboratories. In the first instance the experiments have been analyzed to identify their common features and to isolate whatever simplifications occur within their timing requirements. We feel that the ease of programming in FORTRAN more than offsets the consequent inefficiency, but that later recoding of key portions of algorithms may be necessary. There exists for the IBM 1800 a monitor system designed for large-scale process control. With some extension, it seemed suited to laboratory automation and was adopted.

Below, following an analysis of the experimental requirements and their implications for the monitor programs, the most relevant features of the IBM 1800 computer and the TSX (time-shared executive) monitor system are sketched. The four programs that constitute the core of the laboratory submonitor are followed by a generalized and simplified flow chart particular to the author's own experimental field—electron paramagnetic resonance spectroscopy—as an illustration of a user's program.

The Experiments and Their Requirements

For spectroscopic experiments, a number of significant simplifications do occur. The purpose of a spectroscopic experiment is generally to find the functional dependence of some dependent variable(s) on some controlled variable(s). Very often one or more parameters, e.g., temperature, is to be held constant for portions of the experiment and changed to new values for subsequent portions. The controlled variables are swept in time, but a sweep may be delayed for quite long intervals without detriment to the experiment if it is resumed from the same value later. The data do not need interpretation during the course of a sweep but may be processed after a sweep is completed (a sweep could comprise a very short data list if processing is required very soon after data collection). Analog integration to improve signal-to-noise ratio is a common feature of spectroscopy and introduces a time constant characteristic of the experiment. Fast sampling does not confer any statistical advantage for experiments which are intrinsically long-term.

These numerical and timing characteristics are typical of many types of experiments besides conventional spectroscopy; they may be all classified as "slow" in the time scale appropriate to digital computation. Other common experiments seem to divide fairly sharply into those with high data rates (Kc/sec) and very high data rates (Mc/sec). It should be recognized that the number of data collected in any data-gathering cycle has natural upper bounds so that any experiment requiring high data rates will be of short duration. It is not unreasonable to dedicate the "foreground" of the computer to that experiment during its data-collection phase, since this does not conflict with the slow spectroscopic experiments. Experiments with very high data rates are beyond the power of commercial computers presently available.

With these simplifications in mind, the design objectives and the method of their implementation become almost self-evident.

(1) That no experiment should use more of the facilities of the central processing and arithmetic units of the computer than it requires implies a time-sharing system.

(2) That any experiment has an associated time constant implies that the collection of data may be under the control of a timer internal to the computer, rather than on the receipt of an interrupt signal at the computer from the experiment. However, it would be quite simple to extend the monitor described below to arrange that some experiments could control the time that their data are gathered, and that others depend on the computer for timing.

(3) Internal to any data-gathering operation, accurate timing is unnecessary for spectroscopic experiments. Data points should be taken at intervals equal to or greater than the time constant of the experimental apparatus. If the mean interval

is greater than the output time constant, the only effect is that the duty cycle of the spectroscope is not one hundred percent. For those experiments for which real time is the independent variable it is generally adequate to take the data at roughly equal intervals and to provide in output tables accurate interval information. Should an experiment in fact require accurately set timing intervals, it does lie outside the programming of the monitor operative in this laboratory. However, extension to cover this possibility is only a change in data buffer logic, which would be neither difficult nor time consuming.

(4) A system should be open-ended with respect to data set size. This criterion is not directly met in the system described; however, only a very small number of simple modifications are necessary to modify the size of the data set collected in any single collection cycle. Presently, any experiment may make at most 3500 accesses to the computer and collect up to 14,000 data in any collection cycle. These limitations occur because the system uses fixed buffer areas on disk storage.

(5) The data-collection programs should be open-ended with respect to inclusion of elementary data-gathering and control operations. The entire monitor system should not preclude process-control functions outside those functions explicitly provided for.

(6) Whatever calculation time is not used in the data-collection and control functions should be available on a time-shared basis to programs processing the output of completed data-collection cycles or to nonprocess programs. This is accomplished by assigning the data-gathering program to a permanent area of magnetic memory and having it interrupt the job stream whenever a timer expires. As a consequence, the data-collection program must occupy minimum possible magnetic memory.

(7) Within the limitations implied by finite processing speed and limited core memory and disk storage, the inception and accomplishment of any data-collection cycle (any experiment) should proceed in such a fashion that the experimenter is not aware of the status of other experiments.

(8) In these early stages of laboratory automation, flexibility and ease of programming are of extreme importance. For almost every case when flexibility and efficiency were in conflict, efficiency with respect to either processing speed or the use of memory has been sacrificed in the system described. Fortunately, for the IBM 1800 used in the present implementation, a monitor system called TSX (time-sharing executive) and a FORTRAN compiler were available. In only very few instances did we find it necessary to operate outside the facilities that TSX provided. It is common that a large fraction of the computer time is involved with a small fraction of the code, so that when operating experience is more extensive it will probably prove profitable to eliminate most of the inefficiency of FORTRAN by recoding key portions of the programs in symbolic machine language.

(9) For the experiments considered, ten samples per second seemed the fastest data-rate required, and one sample per second was more typical. The data collection program is capable of sampling multiple experiments at up to 20 samples per second for any or all experiments and has been tested at 50 samples per second. At the latter rate, timing begins to be erratic.

(10) It should be possible at any time to run a nonprocess program by action taken at the console of the computer. Further, when all process activity is complete, the computer should revert automatically to the nonprocess job stream until more process activity is requested.

(11) If a computer is to service a laboratory with multifarious, continuously changing experiments, it should be possible to modify old programs and enter new programs without interfering with running experiments. Under TSX, these activities are not distinct from those of the previous point.

(12) To avoid interleaving outputs or inputs of different users, the data processing I/O facilities (printer, card reader, plotter) may be accessed from variable core only. Interrupt programs should not use these facilities. However, typewriters, which are relatively inexpensive, can be dedicated to particular experiments or sets of experiments and located at those experiments rather than in the vicinity of the computer. They may be used by any program associated with the experiment(s).

Our system is not open-ended with respect to a number of experiments simultaneously in a data collection phase. In San Jose we expect that about fifteen experiments will eventually be connected to the computer and that it will be unusual that more than six will be in a data-collection phase concurrently. The system is designed so that any six of the fifteen may be simultaneously recording experimental data and so that any other requests for input activity will be deferred until one of the previous six terminates its data collection.

IBM 1800 Computer and TSX Monitor System

For details on the IBM 1800 computer or the TSX system the reader is referred to the IBM Systems Specifications manuals. However, in view of the large number of options available, a very brief description of the system installed in the San Jose Research Laboratory is pertinent, particularly since this hardware configuration is adequate, without many superfluous resources, for the number of experiments that the monitor is designed to handle. The machine is a general-purpose digital computer, with a 2-μ sec cycle time and 32K core storage memory with word size 16 bits, supplemented by a multilevel priority interrupt system and a number of process input/output terminals for external communication. The machine is equip-

ped with a card read-punch, a line printer, several typewriters and a keyboard, a Calcomp plotter, 1.5-million words disk storage, and 3 timers (1/8 msec, 1 msec, and 8 msec). Nine data channels, operating on a cycle-steal basis, control both the data-processing I/O equipment and the process I/O terminals. The latter consist of eight groups of 16 bits each of digital input, 16 groups of digital output, two 8-bit pulse counters, 16 analog outputs, and 64 analog inputs. The 96 priority-interrupt terminals are distributed among 24 interrupt levels.[1] In addition to external interrupts, there are interrupt signals generated by each I/O device, by the timers and by programs explicitly. In addition, the machine is equipped with a System/360 channel adapter for direct high-speed machine-to-machine communication. Although the programming is not complete, we anticipate using an IBM 360/50 as a slave to the IBM 1800 to do those calculations for which the smaller machine is too slow, or for which better software is available on the bigger computer.

The TSX monitor system provides most of the control functions necessary for a time-shared laboratory automation monitor. It consists of two submonitors, a nonprocess monitor system which includes an assembler, a FORTRAN compiler, and a set of disk utility programs, and a process monitor which controls the sequence of tasks in the CPU. The process system director program:

(1) controls the sequencing and initiates the loading and execution of process programs;

(2) controls the time-sharing of variable core between process and nonprocess programs;

(3) automatically determines the type of each interrupt and transfers control to the proper servicing program;

(4) provides error messages and executes a specified recovery procedure whenever an execution error occurs.

In addition, the system provides an extensive set of subroutines for simplified control of the process input/output hardware. Many of the TSX functions may be referenced explicitly from users' FORTRAN programs. The most important of these are listed in a simplified form in Table I, to illustrate the facilities already available in TSX.

When the IBM 1800 is operating under TSX, the core memory is divided logically into two sections, called skeleton and variable core. The skeleton contains the portions of the operating system which must be permanently available, those interrupt subprograms which may be time-shared with the mainline program and a skeleton COMMON area. Variable core is used for the job-stream of mainline

[1] 48 interrupt terminals distributed among 8 levels would be adequate.

programs. Interrupts are processed either by the skeleton interrupt routines, without overwriting variable memory, or by temporarily replacing the current mainline program with an interrupt core-load.

TABLE I

TSX SUBROUTINES WHICH MAY BE REFERENCED FROM APPLICATIONS PROGRAMS

CHAIN (NAME)	The program "NAME" is loaded and executed.
QUEUE (NAME, P)	The program "NAME" is entered into the process job stream queue with priority P for future execution.
VIAQ	The highest priority program in the queue is loaded and executed.
LEVEL (I)	An interrupt at level I is entered.
INTEX	In an interrupt servicing program, completion of the service is recorded.
TIMER (SUB, I, J)	After the timer "I" has run "J" intervals, the subprogram "SUB" is executed.
CLOCK (I)	The real time "I" is recorded.
SHARE (I)	Devote an interval "I" to the nonprocess job stream.
ENDTS	Terminate action in the nonprocess job stream at once.
AIS (I, J)[a]	Analog input several points sequentially.
DO (I, J)[a]	Digital output to several addresses.
DAC (I, J)[a]	Analog output to several addresses.
DI (I, J)[a]	Digital input from several addresses.

[a] Here "I" is a control parameter, and "J" is an array containing the input or output addresses and the data transmitted.

THE LABORATORY AUTOMATION SUBMONITOR

The primary functions of a computer monitor program, supervision of priorities and sequencing of programs, and provision of assemblers, compilers, and program loaders, are adequately met by the TSX monitor. In addition, a special-purpose monitor should provide flexible input/output control programs and a powerful subroutine library appropriate to the problems anticipated. It is extremely important that the subroutines be both flexible and easy to use if most of the users are scientists with limited experience and less interest in programming. This paper deals most particularly with the data-collection functions of the nucleus of a new laboratory automation monitor with only limited discussion of the subprogram library which has been and is being prepared in this laboratory.

Four important programming principles—dynamic allocation, preservation of a status record for each time-shared program, concatenation, and modularity of tasks—are fundamental to the input/output routines described below and are extremely useful guides in the related programming to be done by individual users. First of all, there must be a dynamic allocation algorithm for those computer resources that are in short supply. Timers, core buffers, and disk record buffers are allocated on request and made available for future requests as soon as possible after the completion of a timed input/output sequence in the monitor described here. The allocation of each of these facilities is maintained in a small data array in the skeleton area of core storage. The logic of these arrays is indicated in lines 78 to 111 of Fig. 1. The subroutine which collects the data for multiple experiments must remember the current status of each collection cycle. These records are maintained in the COMMON region denoted at the bottom of the figure.

Without a detailed knowledge of each experiment, it is not possible to fix in advance the sequence of operations required, the timing or number of accesses to the experiment, or a number of other parameters associated with any access to an experiment. However, one can easily identify the small number of operational modules from which, if concatenation of modules is allowed, sufficiently flexible strings can be constructed to satisfy quite complicated sequences of input/ouput operations. In fact, such a logical structure is very similar to modern techniques in analog logical circuitry, in which a relatively small number of operational units (adders, integraters, multiplying circuits) are hooked up to provide more complicated transfer functions. The design of digital computers itself uses the same principles of modularity and concatenation. The modularity confers the additional advantage that, should the program not provide some necessary function, it is quite easy to add appropriate modules. In Fig. 1, lines 31–72, some of the modular elements are identified. The first group of elements provides the facility to sweep a set of independent variables with equal increments. The second set permits output from tabular data prepared before the initiation of an I/O cycle. Operation 8 is designed to reset a motor equipped with a sensing potentiometer with the computer closing the servomechanism loop. Operations 9 and 10 provide for analog input, with and without time-averaging [1]. The next group of operations is provided to supplement the time-averaging data collection by allowing an output variable to be incremented only after a specified number of data points have been collected. The final operation provides an interval timer to give precise sampling intervals to experiments in real time.

The core of the laboratory automation sub-monitor is a set of four programs operating within TSX. The central routine, DOIO (Do Input/Output), is a skeleton subprogram entered when a timer or timers expire and accomplishing a series of process input/output operations determined by a command string stored in the

```
      SUBROUTINE INIO( IOPER,IDEV,ICORE)

C...  IOPER    IO OPERATION SEQUENCE DEFINITION

C...       UP TO THREE OPERATIONS IN SEQUENCE IN THIS VERSION
C...            IOPER(1)        NUMBER OF CYCLES
C...            IOPER(2).GT.0  TIME INTERVAL IN MILLISECONDS
C...            IOPER(2).LT.0  TIME INTERVAL IN SECONDS
C...            IOPER(3)        TYPEWRITER FOR OUTPUT MESSAGES
C...            IOPER(4*L)      OPERATION CODE FOR 'L'TH PART OF SEQUENCE
C...            IOPER(4*L+1)    WORD COUNT FOR 'L'TH ITEM
C...            IOPER(4*L+2)    DISK DATA FILE FOR 'L'TH ITEM
C...            IOPER(4*L+3)    RECORD POSITION FOR NEXT DATA TRANSFER
C...                      RECORD LENGTH MUST EXCEED STRING OF ONE CYCLE
C...

C...  ICORE    PROCESS CORE LOAD TO BE QUEUED ON COMPLETION

C...            ICORE(1)        INTEGER IDENTIFIER OF CORE LOAD
C...            ICORE(2)        EXECUTION PRIORITY OF THAT CORE LOAD


C...  IDEV     DEVICE OPERAND AND DIRECT DATA TABLE.  THE INTERPRETATION

C...       OF THE 'L'TH STRING OF OPERANDS DEPENDS ON THE OP CODE AND
C...       WORD COUNT OF THE 'L'TH PART OF THE OPERATION SEQUENCE.
C...       SUCCESSIVE STRINGS ARE CONCATENATED.  UP TO 15 ENTRIES IN IDEV


C..   OP 1      INCREMENT AND RESET ANALOG OUTPUT
C..   OP 2      INCREMENT AND RESET DIGITAL OUTPUT
C..   OP 3      INCREMENT AND RESET CONTACT OPERATE
C..        IDEV(1,4,... ) STARTING VALUE
C..        IDEV(2,5,... ) INCREMENT
C..        IDEV(3,6,... ) MULTIPLEXER ADDRESS

C..   OP 4      ANALOG OUTPUT
C..   OP 5      DIGITAL OUTPUT
C..   OP 6      CONTACT OPERATE
C..   OP 7      PULSE OUTPUT
C..        IDEV(1,2,... ) MULTIPLEXER ADDRESS

C..   OP 8      RESET ELECTROMECHANICAL
C..        IDEV(1)   REFERENCE VOLTAGE
C..        IDEV(2)   REFERENCE M.A.  -   ANALOG OUTPUT
C..        IDEV(3)   POSITION M.A.   -   ANALOG INPUT
C..        IDEV(4)   CONTACT OPERATE M.A.
C..        IDEV(5)   BIT POSITION TO START MOTOR FORWARD
C..        IDEV(5)+1 BIT POSITION TO START MOTOR REVERSED
C..        IDEV(6)   DESIRED POSITION
C..        IDEV(7)   TOLERANCE
C..        IOPER(2)  TIME FOR MOTION OF 1 VOLT ( OR 10 MV. )

C..   OP  9     ANALOG INPUT SEQUENTIAL AND CAT 'N' DATA POINTS
C..        IDEV(1)   STARTING MULTIPLEXER ADDRESS
C..        IDEV(2)   NUMBER OF CYCLES AT EACH POINT 'N'

C..   OP 10     ANALOG INPUT SEQUENTIAL
C..        IDEV(1)   STARTING M.A.

[illegible]
C..        IDEV(1,4,... ) STARTING VALUE
C..        IDEV(2,5,... ) INCREMENT
C..        IDEV(3,6,... ) MULTIPLEXER ADDRESSES
C..        IOPER(4*L+2)  NUMBER OF CYCLES AT EACH VALUE, 'N'

C..   OP 14     INTERVAL TIMER
C..        NO ENTRIES IN IDEV TABLE ARE NECESSARY

C....           FOR THE IBM RESEARCH LABORATORY, SAN JOSE 1800, THE
C.... FOLLOWING OPERATIONS HAVE BEEN DISABLED.      2,5,7,12
C..............................................................


C...  TIMER OPERATIONS
C...        TIMERS ARE ASSIGNED AND RELEASED AS NEEDED.  THE TIMER
C...  NUMBER IS THE EFFECTIVE IDENTIFIER OF THE TASK SEQUENCE.

C...       IF MSK(I).LT.0, THE 'I'TH SOFTWARE TIMER IS RUNNING
C...       IF MSK(I).EQ.0, THE 'I'TH SOFTWARE TIMER WILL SOON EXPIRE.
C...       IF MSK(I).EQ.1, THE 'I'TH SOFTWARE TIMER IS DEMANDING SERVICE.
C...       IF MSK(I).EQ.2, THE 'I'TH TIMER IS QUEUEING A CORE LOAD
C...       IF MSK(I).GT.2.AND.LE.30, THE 'I'TH TIMER IS EXITING WITH AN
C...  ERROR.  THE VALUE INDICATES THE NATURE OF THE ERROR.
C...       IF MSK(I).GT.30, THE 'I'TH TIMER IS AVAILABLE.


C...  BUFFERED DISK I/O OPERATIONS
C...        BUFFERS ARE ASSIGNED AND RELEASED AS NEEDED.  THEY ARE
C...  ASSOCIATED WITH PARTICULAR SOFTWARE TIMERS BY THE ARRAY 'NBUF'.
C...  THE BUFFER SIZE IS THE SAME AS THE DISK SECTOR SIZE, 320 WORDS.
C...  IN EACH DATA TRANSMISSION, THE DATA FOR N ENTRIES OF DOIO ARE
C...  COMMUNICATED, WHERE  N*WC .LT. 321 .
C...        LOGICAL DISK FILES ARE ASSIGNED ON A ONE-TO-ONE BASIS WITH
C...  THE BUFFERS.  THEY ARE, IN FACT, SETS OF SECTORS OF A MODERATELY
C...  LARGE FILE USED ONLY FOR THE REPETETIVE ASYNCHRONOUS TIMED I/O
C...  SERVICE.  EACH LOGICAL FILE HAS 3200 WORDS RESERVED TO IT.  WHEN
C...  THE BUFFERING IS TAKEN INTO ACCOUNT, THIS IMPLIES THE LIMIT OF
C...  3520 WORDS TO BE TRANSMITTED TO ANY SINGLE EXTERNAL FILE.

C...      KBUF(I).NE.0   IMPLIES THAT THE 'I'TH BUFFER IS NOT AVAILABLE.
C...      NBUF(1,I,IT)   NUMBER AND PURPOSE OF BUFFER FOR TIMER 'IT'
C...                          .GT.0      INPUT
C...                          .LT.0      OUTPUT
C...                          .EQ.0      NO BUFFER ASSIGNED
C...      NBUF(2,I,IT)   NUMBER OF ACCESSES REQUIRED TO FILL BUFFER
C...      NBUF(3,I,IT)   NUMBER OF ACCESSES CURRENTLY COMPLETE


      INTEGER  BUF( 322,8 ), NBUF( 3,3,6 ), STATE(16)
      INTEGER     IOPER(15),ICORE(2),IDEV(15)
      EXTERNAL TIMES
      EXTERNAL TDUMM
      COMMON/INSKEL/MSK(6),KBUF(8),IO(4,6),IOP(4,3,6),ID(15,6),ICOR(2,6)
     1 , BUF, NBUF, STATE
```

FIG. 1

skeleton common area of the main computer memory. Subroutine TIMR, also stored in the skeleton area, uses one of the hardware timers to provide DOIO with multiple software timers. Each software timer may be set to expire at integral multiples of the basic timing interval, 50 msec. Subprogram INIO (Initiate Input/Output) is automatically loaded with the user's mainline program. It provides a very simple interface to initiate DOIO, assigns a timer and buffers as necessary, checks the validity of the input data, and stores in the skeleton common area the data required by DOIO. The final routine, EXIO (Exit from Input/Output), terminates any I/O sequence gracefully, releases the timer and buffers associated for use by subsequent requests for service, and enters into the process job stream queue a user's program which presumably interprets and outputs the data collected. In these programs the primary identifier of the tasks to be done when any software timer expires is the software timer number itself. Associated with each timer is a short memory block in the skeleton area. This block contains a fifteen-word

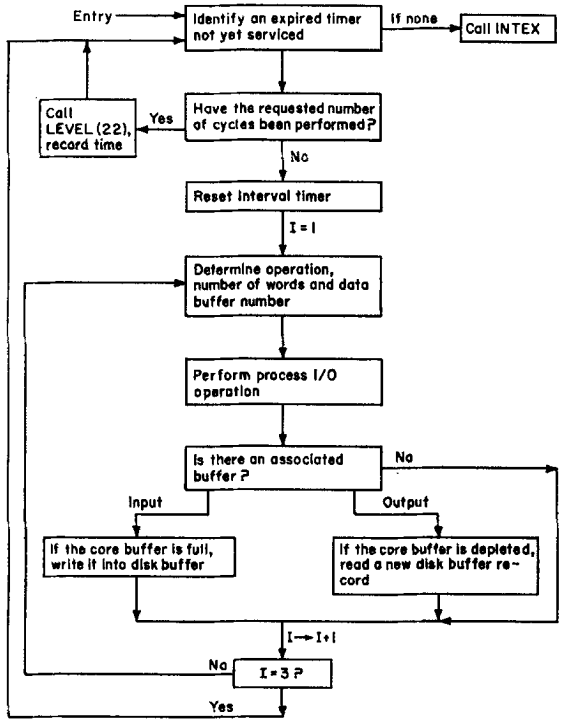


FIG. 2. Simplified logical flow for DOIO (Skeleton interrupt subroutine servicing programmed interrupt level 12) DOIO is entered when one or more timers expire.

command string, a fifteen-word device-address and direct-data table, and a two-word identification of the program to be queued at the end of the cycle. In addition, a nine-word table is used to associate memory and disc storage buffers with the timer and to maintain a current status record of the data being processed by the buffer. Simplified flow charts of the four programs appear in Figs. 2–5. Details of the input data for the initiation program occur in Fig. 1.

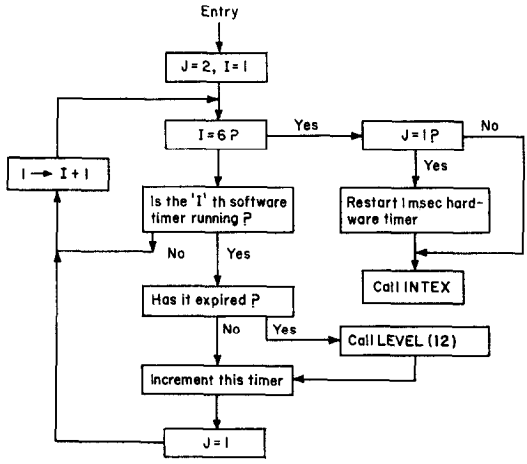


FIG. 3. Logical flow for TIMR (Skeleton subroutine at interrupt priority level 0) TIMR is entered whenever the 1-msec hardware timer expires.

Besides the repetitive timed I/O package, the submonitor includes a set of quite simple programs to permit forced changes in program sequencing from the computer console. Normally, the sequencing is entirely automatic; at the completion of any mainline program the next one in the program queue is initiated. If there is no process program in the queue, a few minutes is devoted to the nonprocess jobstream. However, from the console we may start interrupt programs which:

(1) abort the present nonprocess job in favor of the next nonprocess job;

(2) terminate nonprocess work immediately;

(3) devote a few minutes, starting at the completion of the current process job, to nonprocess work; or

(4) enter into the process job-stream queue a desired program with a desired priority.

Of course, whenever either process or nonprocess work is forcibly terminated, its current status is saved in disk storage, so that it can be resumed later.

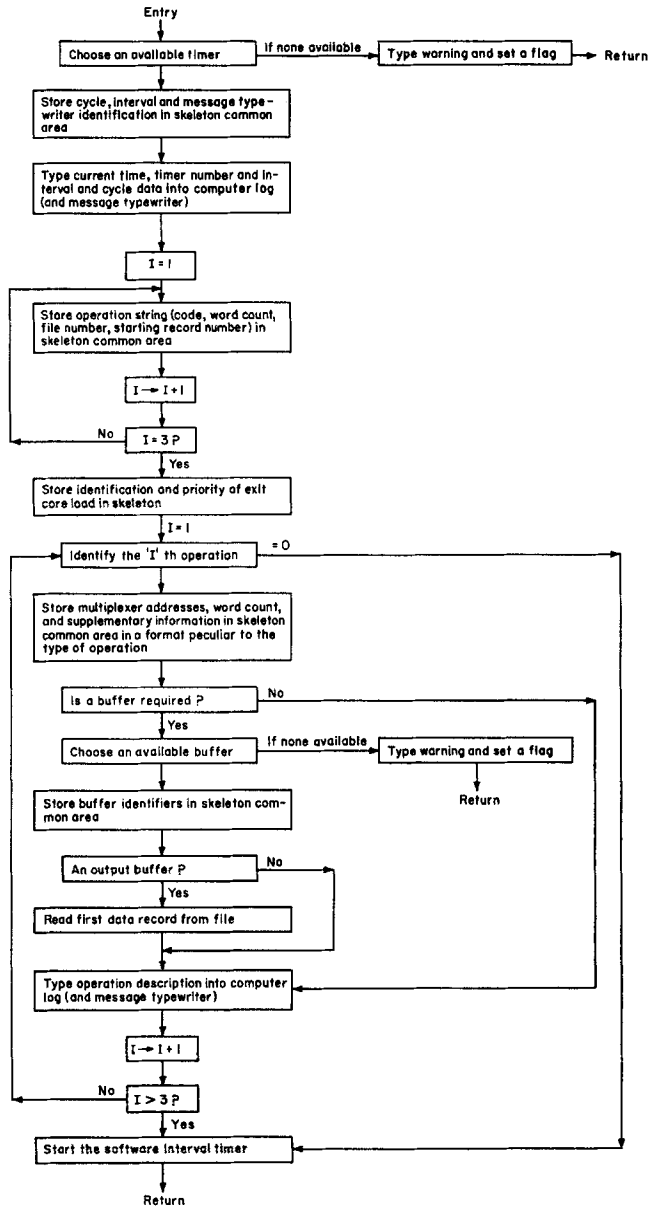


FIG. 4. Simplified logical flow for INIO (Subprogram loaded with program in variable core).

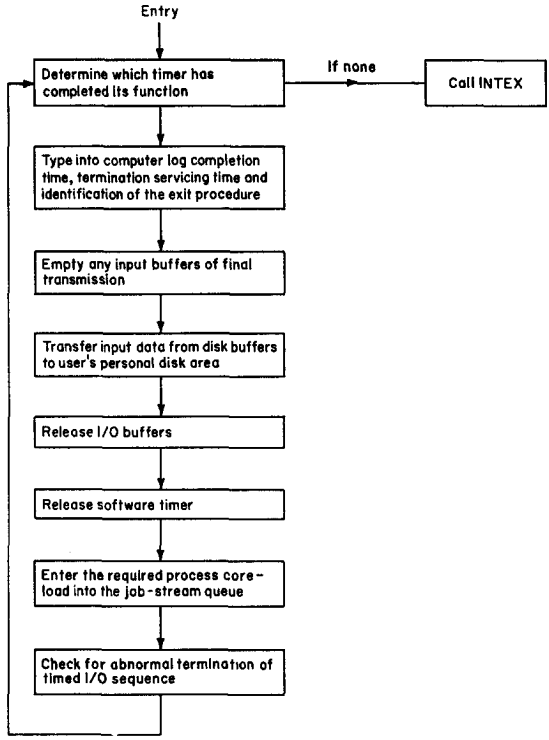


FIG. 5. Simplified logic flow for EXIO (Interrupt core load servicing programmed interrupt level 22) EXIO is entered when one or more repetitive timed I/O sequences is complete.

A USER'S PROGRAM

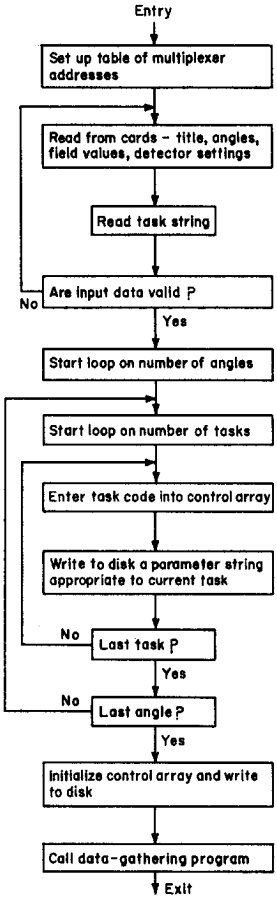
As an example of an application of this time-sharing monitor system, we present drastically simplified logical flow charts of programs to accomplish the tedious initial phases of data collection for electron paramagnetic resonance (EPR) of magnetic centers in single crystal samples. These preliminary experiments are performed to ascertain whether or not a sample has a magnetic impurity and to determine the chemical nature and location of the magnetic site by observation of the angular dependence of line positions. As it has been performed manually in the past, a typical experiment involves a hundred or more sweeps of 10–30 minute duration each, with detection time constants of the order of one second. Between sweeps it is often necessary to adjust the controls of the output detector circuits or to change the orientation of the applied magnetic field. Single crystals seem to

accept desired magnetic dopants only reluctantly, so that a negative experiment is more common than a positive one. We seek to escape these uninteresting phases of the researches. Since only rudimentary feedback from the results is involved in adjusting the detector, this problem was one for which significant effort could be saved without elaborate programming. Much more significant than the extra ease of the experiment is that the computer makes it possible to maintain a uniformly high-quality experiment with more complete records than have been common in the past.

The interface between the spectrometer and the computer is relatively simple compared to the large number of controls available on the spectrometer. It includes all those signals and controls which are commonly modified during the course of an experiment to present and optimize the output data. These include digital control of the magnetic field, analog sensing of the output of a phase-sensitive detector, digital control of the gain, time constant and modulation amplitude of the same detector, a positioning mechanism for a 12-in. rotating electromagnet, and analog outputs to an X–Y recorder in the laboratory. Most of these controls are used in the simple search for signals, but we anticipate that they are also adequate for some of the more sophisticated control programs planned for the future.

The organization of this user's program is quite similar to that of the laboratory automation submonitor itself. The tasks to be performed include data-collection cycles carried out by DOIO. During these it is unnecessary that the main program be resident in variable core. Consequently all data and control arrays are not presumed to be saved in core memory but are stored in disk files, and the program uses variable core only during its data-processing phases. For this program the command string is conveniently read from a set of input cards. Since it is not possible to predict when the next elements of the command string may be required, the entire string is stored in disk storage by an initiation program which is completed before the first task is undertaken. The initiation program (Fig. 6) provides the additional opportunity of generation of a relatively long command string from a compact set of data entered in a format convenient to the experimenter. For the EPR spectrometer a typical data input deck has 12 cards which generate commands for 48 hours of searches for spectra.

A more detailed description of the EPR system will be deferred for future publication. In its present form, the program (Fig. 7) includes modules to reset the magnetic field, to record a spectrum with or without time-averaging, to plot a spectrum, to adjust for base-line drift, to locate absorption lines, to readjust the detector circuits on the basis of previous results, and to set the magnetic field direction. Of course, this program is open-ended so that other modules of instructions may be added if and when desired. Consequently it could, *mutatis mutandis*, control spectroscopic experiments other than electron paramagnetic resonance.



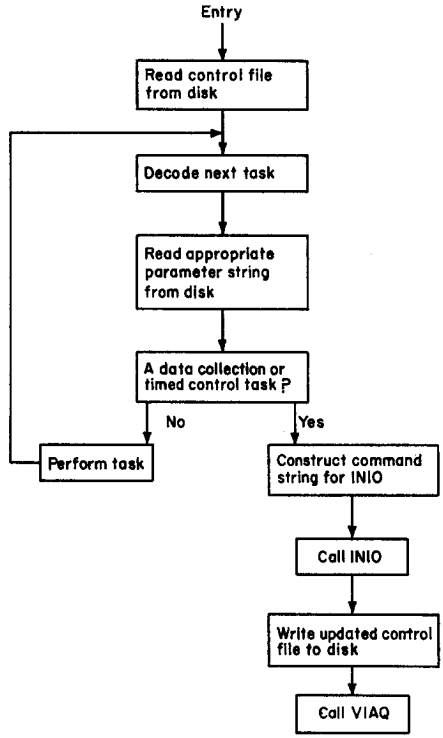


FIG. 6 (left). Logic flow for EPR preliminary data collection initiation.
FIG. 7 (right). Logic flow for EPR preliminary data collection execution.

A COMPARISON

It is pertinent to compare the present system to an extant time-sharing laboratory monitor. This is much to indicate the similarities of these systems, which may be characteristic of many of the systems which will be designed in the near future, as to illuminate the significant differences. Beaucage *et al.* [2] have described a time-sharing monitor implemented on an SDS 920 for the control of multiple neutron-diffraction spectrometers at Brookhaven National Laboratory. The most striking

change in the present system, that the individual experiments are multifarious, is quite superfical from the point of view of the computer since, given programming with the flexibility of a modular system, all the experiments we have considered are very similar. However, most of our applications are for instruments with the data characteristics of a scanning spectrometer, which are very different from those of crystal-diffraction spectrometers. The latter were, even before the impact of digital computation, intrinsically digital devices, whose data handling requirements may be contrasted with the requirements described in the second section of this paper. The primary scaling in a neutron-diffraction spectrometer is done in a counter external to the computer, so that the time scale for transferring data can be measured in seconds or longer, rather than in fractions of a second. However, when a "point" (angles plus count) has been collected, it is usually necessary to perform some short computations before resumption of the experiment. The total data transmitted to or from the experiment is measured in hundreds or thousands of words, rather than the tens or hundreds of thousands of words transmitted by an EPR spectrometer. Consequently, in the system at Brookhaven, interrupts associated with data transfer generally originate in the experiment. At San Jose, since it is not common that much control action need be taken before a file of data are collected, timed interrupts become normal and externally generated interrupts the exception rather than the rule. At the present our external interrupts are usually associated with programs that require core-load interchanges which may themselves take of the order of a second to perform. The timed interrupts originate in a timer internal to the computer. Should it prove necessary, it would be quite simple to replace one or more of these with external interrupts, which could either be external timer(s) or other device(s), so that to make this system operate in a similar fashion to that at Brookhaven would be quite feasible, although perhaps not very efficient.

The sequencing between data reduction tasks is quite different in the two systems. The Brookhaven facility sequences between main-line programs by polling. Since the polling operation itself is quite fast, it is probably not significantly different in practice from the more sophisticated queuing operation provided by TSX as far as the efficiency of the computer is concerned. It does not, however, provide a priority scheme for main-line core loads, which may not be a serious deficiency for experiments as similar as the ones at Brookhaven, but is an undesirable restriction in the environment of San Jose. The limitation of 30 seconds for each user program is quite intolerable at San Jose (e.g., to plot, with axes and labels, a large data file might take 5 minutes). A most important feature of the priority queuing system is that we are able to provide a researcher who wishes to run a nonprocess calculation from the console with a very high priority, on the assumption that his time is more valuable than that of a laboratory instrument which may be running untended. While he may be engaged in debugging a new program in the

foreground, all those data-collection and control functions which have been started will continue in the background, but the intervening data-reduction programs will be delayed until the programmer no longer requires operations at the console. Alternatively, if any experiment requires data-reduction and initiation of new data-collection phases with priority exceeding that of console operation, this type of service can be provided with an interrupt core-load.

In two respects the computer hardware described by Beaucage seems inadequate for the control of multiple scanning spectrometers. The core memory and drum storage allocation for each experiment is roughly what we require for a single spectral scan. The price of the ease and flexibility of FORTRAN programming is that the programs tend to become quite bulky, both because of compiler inefficiency and because we become more elaborate with easier programming. Our monitor system is designed to run with a two-disk system, so that we will have a million words of pseudo-random access memory available for programs and temporary data files. We plan to handle large permanent files either by transferring them to an IBM 360/50 via a channel-to-channel adapter or by transferring them to a temporarily mounted third disk, at the option of the experimenter. Because many of the programs servicing interrupts are much more extensive than those described by Beaucage, they necessarily contain output messages, so that we find it necessary to provide, as well as the line printer, multiple output typewriters. While these are relatively slow, they are inexpensive and may be placed in the laboratories, which are as far as 500 feet from the computer.

DISCUSSION

A laboratory automation monitor system which provides a powerful timed data-collection and control facility for multiple experiments and most of the time-sharing bookkeeping to maintain a parallel conventional batch-processing job-stream has been described. It has been designed primarily for experiments of the type characterized by scanning spectrometers, which represent a large class common to physics and chemistry. At the same time, it does not preclude that a variety of experiments outside this explicit scope be time-shared with the other activities. In particular, experiments with higher data rates may be handled in a burst mode. The contrast between the sequencing method for slow experiments and that for fast experiments is analogous to that between multiplexor and selector channels on modern digital computers.

Inevitably, in the design of a system with the complexity of the present one, there arise a number of alternatives whose consequences are either not clear or not very different. As these were encountered, they were dealt with in such a manner that to

interchange alternative solutions is a simple exercise. For the external interface operations modularity combined with concatenation provides the necessary facility. Many of the parameters of the timed I/O programs, such as the number of timers and the number of buffers, can be easily changed. We have chosen to enslave the experiments during their data-collection phase to timers internal to the computer, but this choice may be replaced by data collection controlled by interrupt signals originating at the experiment. It is not, however, possible to guarantee to any experimentalist that his interrupt signals will be instantaneously serviced. This possibly unpalatable restriction does not depend on the system described above, but is an inevitable consequence of time-sharing a computer for multiple tasks.

The advantages and difficulties of time-sharing a single computer are quite well known. The most serious danger is that of spurious interaction between unrelated experiments. Certain timing delays are inevitable and we expect that programming mistakes on the part of individual experimenters will provoke heated interchanges in the first few months of operation of our system. In any installation it is necessary that the allocation of physical resources, e.g., data-file areas on disk, and interrupt and queuing priorities are under the control of a single individual who can correlate and compromise conflicting requirements. One disadvantage indicated by Beaucage [2], that individual spectrometers are dependent on the reliability of the central computer, can in many cases be avoided easily by maintaining the present stand-alone capability for the spectrometers. However, it is not unlikely that the aid lent by computers will be so significant that experimenters gradually will become reluctant to continue the experiment until an ailing computer is rejuvenated. One of the most insistent arguments against time-sharing a single large facility has been the estimate that it might take many man-years to provide the necessary programming. With the FORTRAN compiler and the TSX monitor system already available, the system described in this paper has required roughly two man-months to complete.

All too often computer systems have been designed without an adequate background or understanding of the problems or systems to which they are directed. The balance between insufficient generality and overelaborate proliferation continues to be elusive. I have attempted to reach what I consider to be the middle ground with a system designed neither for my own experiment alone, nor for all experiments under all circumstances, but for the set of experiments and circumstances prevalent today in this laboratory. Insofar as it is typical of other laboratories, the system may be useful in them. Possibly the programs described here have a range of usefulness beyond the disciplines for which they have been provided. In any case, the programming principles that made this project so easy and which I have attempted to identify explicitly should have a very general applicability.

ACKNOWLEDGMENTS

For programming advice and patience I am indebted to Mr. Wayne Elwood and Dr. Robert Durbeck. The advice and support of Dr. James Eaton and Dr. J. D. Swalen have been invaluable.

REFERENCES

1. For a detailed discussion of "time-averaging" to improve the signal-to-noise ratio, see R. ERNST, *Rev. Sci. Instr.* **36**, 1689 (1965).
2. D. R. BEAUCAGE, M. A. KELLY, D. OPHIS, S. RANKOWITZ, R. J. SPINRAD, and R. VAN NORTEN, *Nucl. Inst. Methods* **40**, 26 (1966).