SLAC-PUB-511
October 1968
(EXPI)

MIDAS: A MULTILEVEL INTERACTIVE DATA ACQUISITION SYSTEM*

R. D. Russell**

ABSTRACT

A system is described for using a medium scale computer in real time data acquisition, analysis, and control of high-energy physics experiments utilizing wire spark chambers.

Objectives

MIDAS is a control program for the IBM 1800 computer (see figure 1 for the SLAC configuration), designed as part of the IBM-Stanford Joint Study Agreement for real time data acquisition, analysis, and control in conjunction with the rho-zero experiment. The implementation had several objectives ...

1. to supply a set of standard functions required for any data acquisition system. These include run control, event control, buffer management, magnetic tape recording facility, data setup for analysis, CRT display control, listing control, message and keyboard control, periodic monitoring facility, and background task scheduling;

2. to provide convenient, table-driven interfacing between experimentally dependent tasks and fixed system functions;

3. to keep as independent as possible the tasks in the system, and to provide a hierarchical structure between them;

4. to permit a large degree of operator interaction with the experiment via an experimenter command post;

5. to provide a framework in which a complete analysis could be performed, free from time constraints and independent of the source of data;

6. to allow both FORTRAN and assembly language routines to be incorporated;

7. to operate in two distinct modes with no reprogramming

   (a) online for performing data acquisition, analysis, display, equipment monitoring, and equipment control in real time,

   (b) offline for analysis and display of data on magnetic tape.

Introduction

The basic operational entity in the system is called a task. Each task is designed to accomplish a single unit of work, and operates independently of other tasks. Since there is only one instruction processor, parallel operation of several tasks does not actually occur, but conceptually this is the best way to visualize their asynchronous operation. A priority scheme is used for allocating the processor to one task in the set of those active at any given time. Tasks remain dormant until explicitly activated by a request for their operation. There are two sources of request generation — other tasks; and signals or conditions in the external environment requiring action by the computer. These external conditions are made known to the computer either actively by an interrupt trigger, or passively by a specialized task in the computer, called a sensor or detector, which continually scans external equipment for meaningful changes in status. This event-driven organization seems the most natural for real time systems[1,2] and was utilized in the SPECTRE system implemented at SLAC on another computer.[3]

Interrupts

The usefulness of interrupt triggers in a data acquisition system depends on the existence of a priority interrupt scheme such as that on the 1800.[4] Triggers are assigned in groups called levels, designed so that processing on a given level precludes interruption by triggers at a lower level, but permits interruption for servicing higher level requests. This hardware is a convenient, effective queue for handling external demands on the computing resource, and enables the computer to remain continually "alive" to all requests. By assigning the most urgent demands to the highest priorities, the hardware allocates the processor to the most important tasks and enqueues other requests for subsequent execution in the proper priority sequence.

The implementation involves a mapping of each interrupt source into a unique item in a table that is filled with task names whenever an experiment is begun. This enables each experiment to assign tasks to interrupts and to arrange the relative priorities between tasks. Additional interrupts are incorporated by simply filling the corresponding slot in the table with a task name.

There are three major classes of interrupt triggers in MIDAS — (1) those involved in run dependent functions, (2) those associated with individual events, and (3) those connected to I/O devices.

Assigned to the first category are the highest priority interrupts which include a set of function pushbuttons required by the operator for run control purposes, signals for unusual error conditions, preset checks, and equipment malfunction triggers. These functions are infrequent relative to other triggers, but require immediate response when they do occur, since they directly affect everything else in the experiment.

* Work supported in part by the U.S. Atomic Energy Commission.

** Stanford Linear Accelerator Center, Stanford University, Stanford, California.

(Presented at 15th Nuclear Science Symposium, Montreal, Canada, October 23-25, 1968.)

The triggers assigned to the next levels are those associated with the data acquisition tasks. These signal the computer that an interesting event has occurred, that data is ready for transmission, or that a data transmission has been completed. They are utilized for equipment control and monitoring, and to synchronize computer action with the movements of active experimental devices. This class also includes the analysis, scope, and timer triggers, as well as interrupts from the high-speed magnetic tape on which raw data is logged.

The lowest levels are assigned to slow speed I/O devices, such as card reader, printer, typewriter, and to anything else which does not need fast, guaranteed response.

Tasks operating on the priority levels constitute the "foreground" activity in MIDAS. "Background" activity is supervised by a monitor which provides queuing and scheduling of the lowest priority tasks for which real time operation is not essential. Most of these tasks are involved with formatting and printing histograms, parameter tables, scaler readings, etc. Since these tasks are often large and slow, they are kept on the disk until requested, then brought into an overlay area of memory for execution. Response time is not critical, so the disk overhead can be ignored, but the space saving is a significant bonus. Requests for background tasks originate from typewriter commands, or from tasks on higher levels which have completed the urgent phase of their operation. Typically, the task on an interrupt level obtains data from an external device in real time, then enqueues a background task to format, print, and record the data as time permits.

The arrangement of individual priorities within the second class depends on the data rates and the amount of analysis attempted. The most urgent tasks are connected to the "next-event" triggers, to the "end-of-readout" triggers, and to interrupts from the logging device. These requests must be guaranteed service with minimal delay if the experiment is to be even attempted. Therefore they are assigned to the highest levels in this class, with the analysis routines immediately below them.

To obtain raw data from the input stream the data setup routines employ an event-sampling strategy that imposes no constraints on the time required to analyze a single event. The acquisition tasks on higher levels proceed independently of the analysis, so that any number of events can be acquired and logged while a single event is analyzed in the remaining time. Naturally the data setup routines are interlocked with the buffer managing routines to guarantee that data setup for analysis belongs entirely to a single event. In the rho-zero experiment the event rate was low enough and the analysis routines fast enough to perform an analysis on every event, so that the sampling method served as a buffer which enabled the analysis to "catch up" after short bursts of very rapid inputs. Other experiments with higher data rates have verified the usefulness of the event-sampling approach.

Display generation is assigned to the next level in order to keep information on the CRT current. An attempt is made to regenerate the display after every new event has been analyzed, which is feasible only when all processor time is not consumed by higher level functions. Were this not true, the scope functions would be placed on the same level as the analysis, in lieu of analyzing at a higher rate, or on a level above the analysis, in which case display updating would be triggered at a rate significantly below the analysis rate.

It is worth noting the advantages which our scope, an ordinary laboratory oscilloscope with a storage tube, has for small and medium scale computers. The pictures are drawn in a manner analogous with plotting on an x-y plotter. There is no need for complicated buffering or continual regeneration of the scope coordinates by the computer. This saves processor time and storage space — both valuable resources in smaller configurations. Because the storage tube retains images indefinitely, annoying flicker is totally absent, and displays can easily be superimposed on one another, a feature useful for comparisons and curve fitting. Image retention can also be used effectively for building up scatter plots involving large numbers of points, a task usually impossible for systems with limited memory and speed. Finally they are cheap when compared to other types of computer displays — the only other hardware components required are two digital-to-analog convertors and two amplifiers to drive the beam deflection plates in the CRT. These combined factors make it quite feasible to consider attaching several of these display units to the computer, although this has not been done in the current versions of MIDAS.

Sensors

Sensors are special tasks designed to detect changes in the environment which require repeated observations over a period of time and which are therefore not easily representable to the computer as interrupt triggers. Examples are switches and thumbwheel dials, DVM readings, etc. The sensor itself may be activated periodically, by external interrupts, or by conditions arising in other programs. Its operation consists of obtaining the current values and settings of external devices, then performing a series of tests on this information to determine whether or not activation of a new task is required. The test results are used to select the particular task from tables assigned to the sensor when the experiment is loaded into the computer. The table lookup enables each experiment to use the same detection mechanism for initiating completely different actions. Sensors form an integral part of any data acquisition system, since they constitute convenient, programmable interfaces for mapping external conditions into computer tasks. This is particularly desirable when using the same equipment in different experiments, because the sensor becomes an extension of the equipment as far as the programmer is concerned.

The MIDAS scope control facility is a good example of a sensor. Every 0.1 seconds this routine reads in the settings of the scope ID thumbwheel and the toggle switches on the scope control panel, then checks this information against the previous settings to select new display or print routines from the task name tables. If additional options, such as x-y plotter output, were desired, the same selection mechanism could be employed by adding another test in the sensor along with a corresponding task name table.

The keyboard input facility and begin-run/end-run interlock routines have also been implemented as sensors in MIDAS.

Operator Interface

The operator command post (figure 2) consists of the CRT, the typewriter/keyboard, a set of function pushbuttons with associated status lights, a panel of toggle switches, thumbwheel dials, and status lights, and a set of numerical display scalers. The intention is to make readily available to the experimenter as much information as possible, and to permit him a large degree of control of the experiment via the computer. The various input components of this interface (typewriter, scope ID selector, and pushbuttons) are table driven for easy use.

There are two types of CRT displays, those based on individual events and those showing summaries of event data. Event-by-event schematic plots of the spark chamber planes and hodoscope counters, with marks to indicate the location of sparks and counters which fired, are extremely useful for equipment checkout and monitoring. By inhibiting the screen erase between events and superimposing the successive displays, a distribution of sparks is built up in which any dead spots or malfunctions are easily discernible. Another display draws the tracks found by the analysis in positions relative to the chamber schematic.

The summary plots are histograms and scatter plots of various phenomena. These include results of the analysis (mass, energy, angular distributions), self-checks on the analysis (track deviations), and checks on the raw data (spark and hodoscope distributions).

Communication via the typewriter utilizes a simple command language in conjunction with an internal descriptor table (see appendix for commands). There are two basic classes of descriptors — task names; and parameter names — with commands to operate on each. The descriptors contain external names, internal names (storage addresses), and other information about the properties of the items being defined.

Tasks can be executed in the background once, or on an event-by-event basis, with any necessary parameters supplied from the typewriter. Parameters can be of four types — single and double precision integers, floating-point numbers, and character strings. They can be interrogated or changed by the experimenter at any time.

Explicit commands are part of the basic interface, but the more common command words can be dropped by making the command implicit in the name. Effectively each name becomes a command and the number of commands available is therefore unlimited. This is implemented by utilizing the additional information about the name in its descriptor. If the descriptor type is "task", entering the name obviously implies "execute the task", with any parameters needed by the task included after the name in the input. If the descriptor type is "parameter", the name alone indicates "type out the current value", whereas the name followed by a value means "change the parameter to this value." The descriptors also provide useful information when formatting parameter values both for typing them individually and for printing and logging the descriptor table as a whole. Finally the parameter descriptors can cause the background scheduler to execute an associated task whenever the value of the parameter is changed by the operator. This is especially useful for recomputing other parameter values derived from the one that was changed, or for readjusting control settings on external equipment as a consequence of the parameter's new value.

For debugging purposes the typewriter permits interrogation or modification of any word in storage, and dumping of any area of core, features frequently used in developing other parts of the system.

Additional Comments

The current system does not permit assemblies or compilations to be performed as background tasks, so that online reprogramming is not possible. FORTRAN routines can be incorporated, but they proved to be too large and too slow to be of any use in the rho-zero experiment. MIDAS components were written in assembly language for the same reasons, and the fact that FORTRAN is too restrictive and cumbersome for most system functions. However compatibility considerations and lack of time and manpower required us to utilize wherever possible the TSX operating system supplied by IBM for the 1800.[5] The result is a somewhat larger and slower resident nucleus than would be necessary had everything been written specifically for MIDAS. The TSX program loader builds and maintains all core loads on the disks in absolute format, including the overlay tasks. Therefore all routines which might be needed at any point in the experiment must be relocated and stored on the disk when the core load for that experiment is built. This speeds up subsequent loadings significantly, but modifications or additions require rebuilding the entire core load. This is bothersome during the initial phases of an experiment, when new routines are frequently being added and debugged, but is tolerable after the programming stabilizes if one considers the amount of storage necessary to maintain a resident relocating loader and accompanying symbol dictionary.

The hierarchical task structure and relative independence of the components has permitted a simple implementation of two modes of operation. Online the data acquisition tasks are activated to input the data, while offline equipment triggers are blocked and a tape reading task obtains data from magnetic tape. In either case the data is deposited in the same buffer chain, so that other tasks in the system need not know or care which particular data source is being employed. They operate identically in either mode. The event-sampling strategy for passing data to the analysis works quite well for analyzing all events offline because the data rate from tape adjusts to the rate at which the analysis can process events. Mode switching can be effected at any time by a command from the typewriter.

MIDAS includes a timer control facility for periodic activation of tasks at rates varying from several times a second to once every several minutes. This is used to initiate the scans of all MIDAS sensors, as well as a special monitor task every 10 minutes during a run. This monitor recomputes the chamber fiducials on the

basis of current data, reads all scalers and DVM's, dumps these along with the parameter table onto both printer and tape, and notifies the operator of any unusual conditions detected.

Although originally designed to use spark chambers as the principle acquisition equipment the basic system is not constrained to any particular device or data format, and has incorporated much additional equipment as the experiment progressed. A few changes in the input routine enable any device, digital and/or analog, to be included. The data rates attainable are a function of the device itself, the amount of data read per event, the amount of space allowed for buffers, and the event blocking factor for recording on tape. The availability of cheap selector-type data channels having direct access to memory permits I/O activity independent of the CPU, so that digital inputs, analog inputs, tape logging, and CRT displays can be in progress simultaneously.

Appendix: MIDAS Typewriter Commands

1. In the command definitions, a <name> can be either an alphameric name or an absolute core address, preceded by a slash if the address is in hexadecimal, e.g., XYX; 9572; /F04B.

2. A <value> can be either a decimal or hexadecimal integer constant, a floating-point constant, or a character string.

3. Command words can be typed in full, but the first letter alone will suffice.

4. Comments are entered by first typing a quote mark (') and then the text of the comment, e.g., 'This is a comment.

5. All commands and comments are free-field with the component parts separated by at least one blank.

6. Command definitions:

   TYPE <name>
   The value of parameter <name> is typed.

   HEX <name>
   The hexadecimal representation of the value of parameter <name> is typed.

   SET <name> <value>
   The current value of parameter <name> is replaced with <value>.

   XEQ <name> <value, if needed>
   Enqueues task <name> for execution once in the background. The <value> is supplied only if needed.

   OPEN <name> <value, if needed>
   Enqueues task <name> for execution in the background after the analysis of each subsequent event. The <value> is supplied only if needed.

   CLOSE <name>
   Removes task <name> from the background queue.

   ADDRESS <name>
   The hexadecimal core address of parameter or task <name> is typed.

   DUMP <name1><name2>
   The contents of core storage between locations <name1> and <name2> are dumped onto the printer in hexadecimal format.

Bibliography

1. Dahm, D.M., Gerbstadt, F. H., and Pacelli, M. M., "A System Organization for Resource Allocation," Communications of the ACM, Vol. 10, No. 12 (Dec. 1967), pp. 772-779.

2. Gaylord, Charles V., "Multiprogramming and the Design of On-line Control Systems," Data Processing Magazine, (May 1968), pp. 26-38.

3. Brown, R. M., Fisherkeller, M. A., Gromme, A. E., and Levy, J. V., "The SLAC High-Energy Spectrometer Data Acquisition and Analysis System," Proceedings of the IEEE, Vol. 54, No. 12, (Dec. 1966), pp. 1730-1734.

4. IBM 1800 Functional Characteristics, Form A26-5918-5, International Business Machines Corporation, San Jose, California.

5. IBM 1800 Time-Sharing Executive System Specifications, Form C26-5990-1, International Business Machines Corporation, San Jose, California.

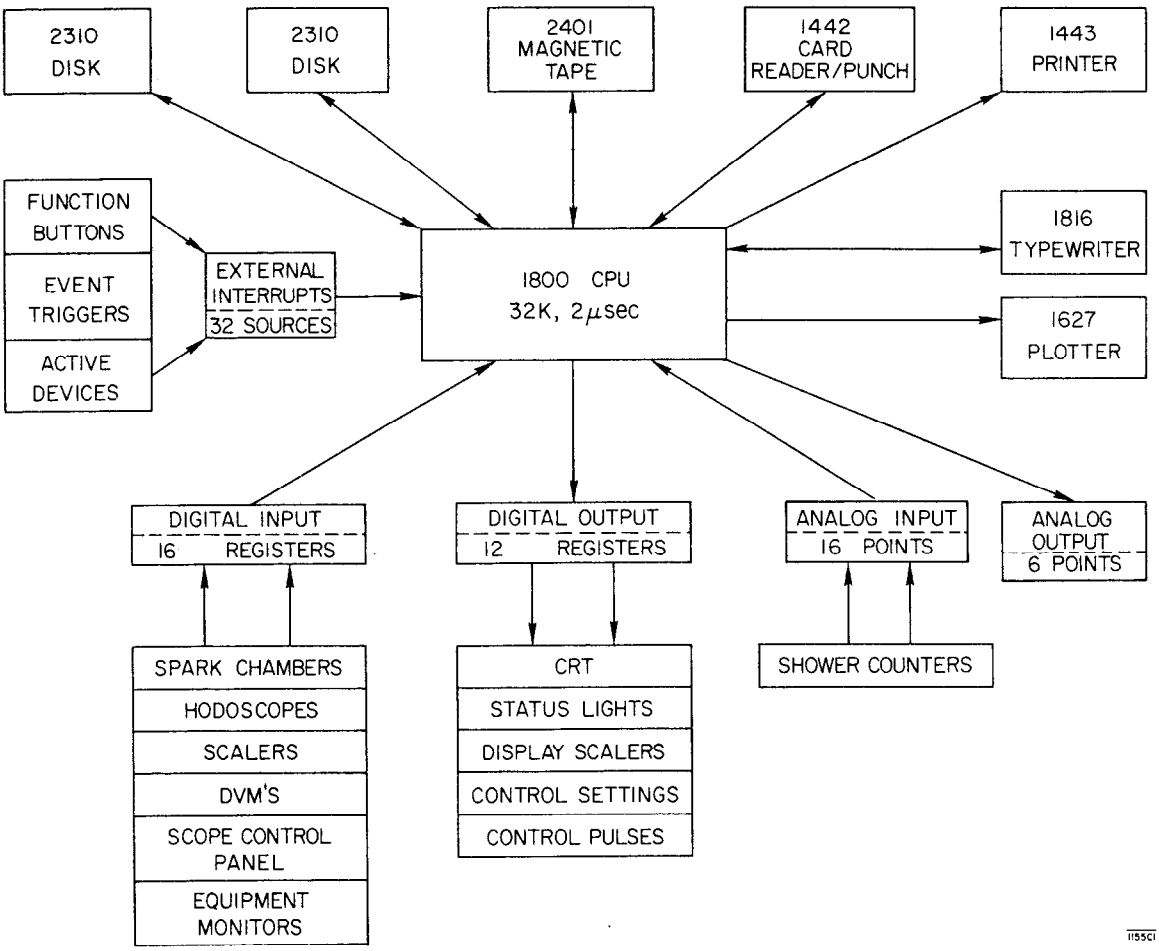
2310 DISK
2310 DISK
2401 MAGNETIC TAPE
1442 CARD READER/PUNCH
1443 PRINTER
FUNCTION BUTTONS
EVENT TRIGGERS
ACTIVE DEVICES
EXTERNAL INTERRUPTS
32 SOURCES
1800 CPU
32K, 2μsec
1816 TYPEWRITER
1627 PLOTTER
DIGITAL INPUT
16 REGISTERS
DIGITAL OUTPUT
12 REGISTERS
ANALOG INPUT
16 POINTS
ANALOG OUTPUT
6 POINTS
SPARK CHAMBERS
HODOSCOPES
SCALERS
DVM'S
SCOPE CONTROL PANEL
EQUIPMENT MONITORS
CRT
STATUS LIGHTS
DISPLAY SCALERS
CONTROL SETTINGS
CONTROL PULSES
SHOWER COUNTERS
1155C1

Fig. 1

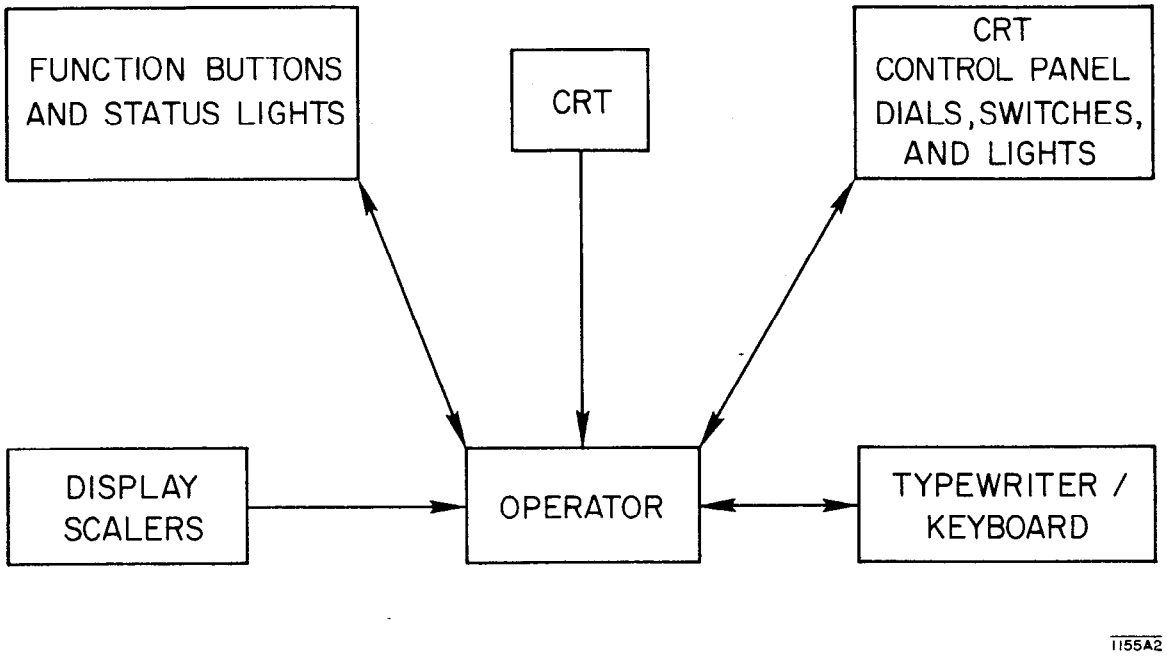
FUNCTION BUTTONS AND STATUS LIGHTS
CRT
CRT CONTROL PANEL DIALS, SWITCHES, AND LIGHTS
DISPLAY SCALERS
OPERATOR
TYPEWRITER / KEYBOARD
1155A2

Fig. 2