AN INSTRUMENTATION AND CONTROL SYSTEM FOR THE AT-2 ACCELERATOR TEST STAND*

E. A. Wadlinger, D. B. Holtkamp, H. D. Holt, AT-2, MS-H818
Los Alamos National Laboratory, Los Alamos, New Mexico 87545 USA

Summary

A data-driven subroutine package, written for our accelerator test stand (ATS), is described. This flexible package permits the rapid writing and modifying of data acquisition, control, and analysis programs for the many diverse experiments performed on the ATS. These structurally simple and easy to maintain routines help to control administratively the integrity of the ATS through the use of the database. Our operating experience indicates that the original design goals have been met. We describe the subroutines, database, and our experiences with this system.

Introduction

We have created and implemented an instrumentation, control, and data acquisition (ICA) subroutine package to expedite activities on the ATS. The software package is a database-oriented system that stores all pertinent information required to run and acquire data from the ATS. The database, which is easy to access and modify, has complete information about the ATS diagnostic hardware. Most change requests by users are handled by modifying entries to the database.

Constraints were placed on the system by its designers to achieve a high-quality system in a minimum amount of time. A specification was written and followed, setting minimum standards for documentation and coding. The final result is a control and data acquisition computer code that is structurally simple and helpful in preventing mistakes and in correcting errors. In addition, the software maximizes the speed for performing the experiments and keeps manpower costs to a minimum. The system knows the location of all the hardware on the ATS and prevents hardware conflicts. Extensive testing and error checking was done before moving the code to the ATS. Generally the errors found during the final check-out on the ATS were minor and were easily fixed.

The following sections describe various features of the ICA system beginning with a system overview followed by a brief description of the user-callable routines. We then present and explain one example of a user program. We describe the features of the database and database editor and finally describe the device handlers that are used by these user-level routines to do the actual I/O to the CAMAC modules.

System Overview

Figure 1 shows the hardware for the ICA system. The test stand is interfaced to the CAMAC modules in the crates shown in the figure. The CAMAC crates are tied together through byte serial fiber-optic U-ports to a Kinetic Systems 2050 serial driver, which attaches to the Unibus of a VAX 11/750. The fiber-optic highway electrically isolates the various CAMAC crates, including the crate in the 100-kV high-voltage column.

Figure 2 is a block diagram of the entire ICA system. We show the diagnostic and accelerator devices connected to the ICA hardware (fig. 1) that is controlled by the ICA software described in this paper.

*Work supported by the US Department of Defense, Defense Advanced Research Projects Agency, and Ballistic Missile Defense Advanced Technology Center.

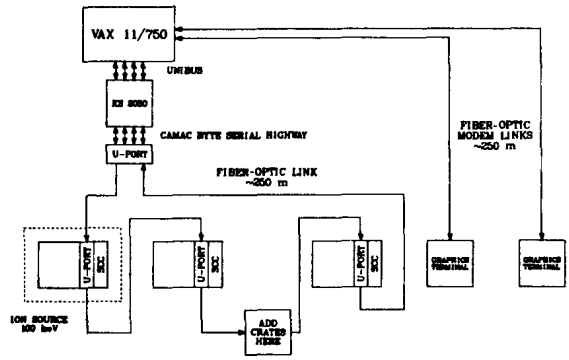

Fig. 1. ICA system hardware.

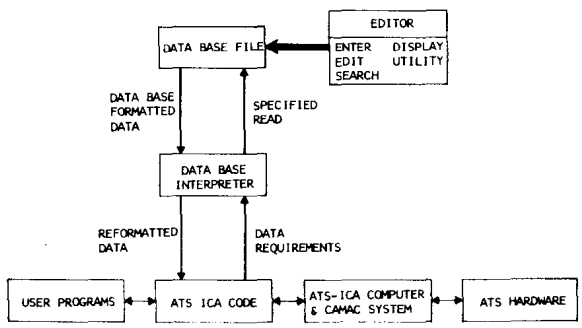

Fig. 2. ICA software functions.

The user controls the ATS with his own program that calls subroutines in the ICA package and gives a command for a specified device such as an emittance scanner. The ICA software obtains the necessary information from the database to perform the command and then executes the command. A database editor was written to create and modify the database for the ICA subroutines. Using this editor guarantees the integrity of the database and forces the user to input the required information needed when defining a new device.

User Routines

The software is designed so that from a user's point of view it is very easy to implement and change programs. The software consists of a set of subroutines each of which performs a specific task on a named device. One may read information from or write information to a device, move a device to a specific location, or check the status of the device. All subroutines in the package contain an error flag that, if not equal to zero on subroutine return, indicates an error condition. An informative message is simultaneously sent to each user's computer terminal. Table I gives a short description of the user callable subroutines.

TABLE I

USER SUBROUTINES AND CALL PARAMETERS

SUBROUTINE CALL LIST PARAMETERS

NAME	VARIABLE TYPE	VARIABLE DESCRIPTION
ARRAY	REAL	Data passed in subroutines AREAD and AWRIT
CHKPOS	REAL	Variable denoting the measured position of a device
COMENT	CHARACTER	Description of the error
DEVNAM	CHARACTER	Variable giving the device name
DIMEN	INTEGER	Dimension of the variable array ARRAY
ERROR	INTEGER	Subroutine error code (0 on return indicates no error condition)
LOCAT	REAL	Location where device is placed
PARAM	REAL	Value corresponding to parameter PARNAM
PARNAM	CHARACTER	The name of a parameter
SUBNAM	CHARACTER	Subroutine name where the error occurred
WORD	INTEGER	Data word passed in subroutines WRECV and WSEND

USER SUBROUTINES

SUBROUTINE AREAD(DEVNAM,ARRAY,DIMEN,ERROR)
This subroutine reads and scales an array of values from the device DEVNAM using the appropriate CAMAC device handler

SUBROUTINE AWRIT(DEVNAM,ARRAY,DIMEN,ERROR)
This subroutine scales then writes the ARRAY values to the device DEVNAM using the appropriate CAMAC device handler

SUBROUTINE DVIN(DEVNAM,ERROR)
This subroutine moves the device DEVNAM to the in limit location. The current position word is updated to the in limit position

SUBROUTINE DVINIT(DEVNAM,ERROR)
Load the appropriate common blocks from the data base for the given device name DEVNAM

SUBROUTINE DVLOCA(DEVNAM,LOCAT,CHKPOS,ERROR)
This subroutine takes a device named DEVNAM and places it at location LOCAT. The device is moved from its current location directly to the desired location. The current position datum is updated

SUBROUTINE DVOUT(DEVNAM,ERROR)
This subroutine moves the device DEVNAM to the out limit location. The current position word is updated to the out limit position

SUBROUTINE DVSTAT(DEVNAM,PARNAM,PARAM,ERROR)
This subroutine returns a parameter value PARAM for a device DEVNAM and parameter PARNAM indicating the current value for this parameter

SUBROUTINE ERRMSG(SUBNAM,COMENT,MODE)
Write error messages to the user terminal. Proceed, pause, or stop after typing the message depending on the value of MODE

SUBROUTINE WRECV(DEVNAM,WORD,ERROR)
This subroutine reads and scales a value from the device DEVNAM using the appropriate CAMAC device handler

SUBROUTINE WSEND(DEVNAM,WORD,ERROR)
This subroutine scales then writes a value to the device DEVNAM using the appropriate CAMAC device handler

Example

We give an example of the procedure involved in applying the ICA software to a new device. The sample application program is listed in fig. 3. The device of interest is an electrostatic emittance scanner.[1] This device is stepped across the beamline and at a given position in the beam it scans the beam divergence.

```
      PROGRAM SCAN
C
C     PURPOSE
C
C           THIS PROGRAM PERFORMS AN ELECTROSTATIC EMITTANCE
C           SCAN ON THE ATS
C
C     VARIABLES
C
C           DEVNAM  ; CHARACTER*6 DEVICE NAME VARIABLE
C           ERROR   ; INTEGER ERROR CODE
C           LOCAT   ; ARRAY OF DESIRED EMITTANCE DEVICE LOCATIONS
C           FLAG    ; (1.0) = (ENGAGED   DISENGAGED) LIMIT SWITCH
C           ISTEP   ; NUMBER OF LOCATION STEPS ACROSS
C                     THE BEAM LINE FOR EMITTANCE SCAN
C           ARRAY   ; REAL ARRAY OF ADC DATA FOR BEAM
C                     DIVERGENCE SCAN
C           DIMEN   ; INTEGER DIMENSION OF ARRAY
C
C           ****************************
C
      CHARACTER*6 DEVNAM
      REAL ARRAY,LOCAT,FLAG
      INTEGER ERROR,ISTEP,DIMEN
      DIMENSION LOCAT(100),ARRAY(1024)
C
C     READ REQUIRED DATA
C
      TYPE *,' READ IN DEVICE NAME, NUMBER OF STEPS IN SCAN'
      ACCEPT *, DEVNAM,ISTEP
      TYPE *,' READ DESIRED STEP LOCATIONS IN EMITTANCE SCAN'
      ACCEPT *, (LOCAT(I), I=1,ISTEP)
C
C
C     INITIALIZE THE EMITTANCE SCANNER
C
      CALL DVINIT(DEVNAM,ERROR)
            IF(ERROR .NE. 0) GO TO 200              ! ERROR CHECK
C
C           MOVE THE SCANNER TO THE OUT LIMIT
C
      CALL DVOUT(DEVNAM,ERROR)
            IF(ERROR .NE. 0) GO TO 200              ! ERROR CHECK
C
C     PERFORM THE EMITTANCE SCAN
C     MOVE THE SCANNER THROUGH THE BEAM AND SCAN AT EACH
C     POSITION.  STOP THE SCAN WHEN THE IN LIMIT POINT IS REACHED.
C
      DO 100 I = 1,ISTEP
C
C           MOVE TO THE DESIRED LOCATION
C
            CALL DVLOCA(DEVNAM,LOCAT(I),CHKPOS,ERROR)
            IF(ERROR .NE. 0) GO TO 200              ! ERROR CHECK
C
C     TAKE DATA FROM THE 1024 CHANNEL ADC
C
            CALL AREAD(DEVNAM,DATA-ARRAY,ARRAY-DIMENSION,ERROR)
            IF(ERROR .NE. 0) GO TO 200              ! ERROR CHECK
C
            ANALYSE THIS SET OF DATA
C
            CALL ANALYS(DEVNAM,ARRAY,DIMEN,ERROR)
C
100   CONTINUE
C
C
C     COMPLETE THE ANALYSIS AND END THE PROGRAM (USER SUPPLIED)
C
      STOP
C
200   CONTINUE
C     ERROR HANDLING SECTION
C
      CALL ERRMSG(...)
C
      END
```

Fig. 3. Application program for emittance scan.

We determine the hardware configuration for the scanner, then run the database editor to update the database. This editor prompts for all the required information including the device name, address location of the various CAMAC modules, CAMAC module identification name, scale factors, and names of other devices to check against for conflicts.

There are three CAMAC modules associated with the emittance device: a 1024-channel ADC, a stepping-motor controller, and an input register to look at the status of limit switches. The program outlined performs the emittance scan. The procedure uses the subroutines DVINIT (obtain the required information from the database and initialize the CAMAC modules), DVOUT (move the emittance gear to the out location), DVLOCA (move the emittance device to a desired location), AREAD (read and scale the array of data from the ADC at the given location), ANALYS (user-supplied subroutine for data reduction), and ERRMSG (error-message routine). Except for ANALYS these are all ICA routines available for inclusion into any user's application program.

ATS Database and Database Editor

Figure 2 includes a functional diagram of the ATS database system. The ICA software maintains named FORTRAN COMMON areas in the user task where parameters are stored. These parameters are necessary for the successful execution of the ICA routines. To load this storage area during device initialization, the interpreter/processor is called and asked to supply specific information from the database file. The interpreter knows more about the database organization than the ICA software does, so it knows specifically where to look for the requested information. The data are read from the database file in character format, then the interpreter reformats the numerical data into floating-point values that the working software expects to receive. At execution time, the data in the database file is fixed; at other times, however, the information it contains may be updated, augmented, or removed. This is the task of the database editor.

The ATS database editor is a menu-driven editor for the database file. It is composed of six functional modules each with its own menu. The module ENTER allows the user to build a record with complete prompting for every record field. Once all information is submitted, the contents are presented for approval. At this point, any of the record's information may be edited. The old input is always listed before the user is prompted for the new. If the record is perfect, the record may be entered into the database file.

The module EDIT enables the user to edit any field in a record. Access to a particular record is by device name and procedure handler. Old field input is presented before new is entered. This section can also delete a record. When edited, the record may be entered back into the database. The module SEARCH will search the database file for records with common entries in any of the four active fields. It is possible to do a logical "and" with multiple entries. A scratch file is written by SEARCH, and it may be displayed on the terminal, saved, or deleted. The module DISPLAY will type the contents of the entire database to the terminal or print it to the line printer.

The last module UTILITY contains tools to effect specialized tasks. Here one may type a list of input/output status-error explanations to the terminal, eliminating look-up in a system manual. It is possible to rename a device by changing the DEVNAM field in all of the records having the old device name to the new device name. We can remove records having a particular device name from the database file. To aid in adding new devices that are similar to existing ones, there is a device copy option that duplicates a device's records with a new device name. This avoids re-entering several records at the expense of some editing of parameter values.

It might be necessary to focus on the module addresses contained in the database. For this reason, one may print to the line printer or type to the terminal a listing of the ordered CNA (crate, station number, and subaddress) addresses followed by the device name, procedure handler, and CAMAC handler. The parameter information is not listed. This tool provides a way to find out what devices use which hardware modules. If one of those modules is moved to another crate, a given address field may be changed globally to a new address.

Camac Device Handlers

There are several features of ICA standard devices handlers that are quite general and apply to all handlers. The handler is the most primitive class of routines in the ATS software package. It makes the calls to the DOE standard CAMAC subroutines[2] (CDREG, CFSA, etc.) that do the actual I/O to the modules. In general, these are the only type of subroutines called from a handler, with the exception of the standard error message routine handler, ERRMSG. The CAMAC driver for the Kinetic Systems 2050 was modified for use on the VAX by Peter Clout.[3]

The handlers are re-entrant in the following sense: If local variables are generated and defined by one call (say, an initialization call) and they are needed in subsequent calls, then these local variables are always passed back to the routine as part of the call. For example, two devices having different names use a particular module for different functions. The first logical device accesses the module with a call to the handler and the second device does the same. The CNA information must be passed each time the handler is called, since the handler does not know which logical device it is servicing, or indeed if there is one or more of these particular modules in the system. This way only one software handler needs to be linked to the main program to service several identical CAMAC modules.

We normally use interrupts (LAMs) in device handlers because they are an efficient tool for optimizing overall system performance. A good example of how to use LAMs is in using a stepping-motor controller module. The module is loaded with a certain number of stepping-motor pulses, then the handler suspends the task, freeing the CPU while waiting for the motor to stop. When the controller is finished, a LAM is generated that wakes up the suspended task. Using a LAM is more efficient than polling the stepping-motor controller continuously for a response to find out when the motor stops.

Conclusion

The ICA system has exceeded the initial expectations. We had a working system within 8 man-months from the time we began to write the specifications. The system has expanded significantly in the two years since it was first implemented, but this anticipated expansion has always occurred within the original guidelines. The system is ideally suited to the changing needs of an experimental environment.

References

1. Paul Allison, Joseph D. Sherman, and David B. Holtkamp, "An Emittance Scanner for Intense Low-Energy Ion Beams," IEEE Transactions on Nucl. Sci., 30, No. 4, 2204 (August 1983).

2. "Subroutines for CAMAC," ESONE/SR/01, DOE/EV/0016 (1978).

3. Peter Clout, Los Alamos National Laboratory, Group AT-3, MS H808, Los Alamos, NM 87545.