LA-UR-85-1757

CONF 850577 --O

RECEIVED BY OSTI

JUN 07 1985

Los Alamos National Laboratory is operated by the University of California for the United States Department of Energy under contract W-7405-ENG-36.

TITLE: A DATA ACQUISITION COMMAND INTERFACE USING VAX/VMS DCL

LA-UR--85-1757

AUTHOR(S): R. V. Poore
D. M. Barrus
G. Cort
J. A. Goldstone
L. Miller
R. O. Nelson

DE85 012747

SUBMITTED TO FOURTH BIENNIAL CONFERENCE ON REAL-TIME COMPUTER APPLICATION IN
NUCLEAR AND PARTICLE PHYSICS, CHICAGO, ILL. MAY 20-25, 1985

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

By acceptance of this article, the publisher recognizes that the U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or to allow others to do so, for U.S. Government purposes.

The Los Alamos National Laboratory requests that the publisher identify this article as work performed under the auspices of the U.S. Department of Energy.

Los Alamos Los Alamos National Laboratory
Los Alamos, New Mexico 87545

DISSEMINATION OF THIS DOCUMENT IS UNLIMITED

25W

A DATA ACQUISITION COMMAND INTERFACE USING VAX/VMS DCL

R. V. Poore, D. M. Barrus, G. Cort,
J. A. Goldstone, L. B. Miller and R. O. Nelson
Los Alamos National Laboratory

Abstract

The user interface to a data acquisition system is being developed at the Los Alamos Weapons Neutron Research Facility using the VAX/VMS command language interface DCL. Commands are being implemented which provide for system initialization and control functions and FASTBUS diagnostics. The data acquisition system incorporates the concept of a data acquisition "state" (running, halted, etc.) where a certain subset of input commands is allowed.

Introduction

The Weapons Neutron Research Facility supports basic and applied research in condensed matter and nuclear physics using neutrons produced by the 800 MeV proton beam from the Los Alamos Meson Physics Facility (LAMPF). The commissioning of the new Proton Storage Ring at LAMPF with its higher beam intensities requires a new data acquisition system. Consequently, efforts are under way in the computer group to design such a system utilizing DEC VAX computers in conjunction with FASTBUS and CAMAC. As designed, the system will consist of 8-12 VAX computers linked together with Ethernet, and a large VAX as a central data analysis machine. In this system the computer serves as a user interface to the real-time portion which is confined to the FASTBUS and CAMAC. This has been described in an earlier paper [1].

Design and implementation specifications were determined after the generation of a requirements document by the users [2] and a design and implementation specification by the computer group [3]. Numerous meetings attended by the users and the computer group refined the basic goals of the new data acquisition system. In trying to develop more user friendly and flexible commands for this system we established the following basic goals for the new data acquisition command interface:

- 1) The commands must be usable by all people conducting experiments at the facility. They should not be unique to each experiment.
- 2) The addition of new options or commands must not impact the functioning of the system.
- 3) Error and message handling facilities must be centralized.

- 4) Help facilities must be provided for on-line assistance to the experimenter.
- 5) In order to reduce the programming and maintenance effort as much vendor supplied software as possible must be utilized.
- 6) An automatic method must be available to control the experiment through the use of command files.
- 7) Documentation for the system must be complete and accurate.

We feel that the above goals are best met by implementing the command interface within VAX/VMS DCL. In particular, the fact that DCL allows each command to be contained in its own stand-alone program provides isolation between commands therefore allowing additional commands to be implemented without substantially affecting existing ones. The main area of possible interaction occurs when the experiment data base [4] (resident in a VAX/VMS global section) is accessed.

The current design of our data acquisition system software integrates locally written commands in Pascal with the existing VAX/VMS DCL commands as supplied by DEC. This has resulted in other desirable benefits. All of the commands and features of DCL as developed by DEC are available for use during data acquisition, we receive the manufacturer's enhancements, and we are relieved of the maintenance of the command language interpreter as well as the effort required to write such an interpreter.

While not directly a part of DCL the centralized VAX/VMS error and message facility is flexible and utility programs are provided by DEC which allow user creation of messages. The errors produced by a program and signaled using the facility are accessible in a DCL command file.

Likewise the VAX/VMS help facility also allows user written help files to be integrated into the system help package and invoked using the VAX/VMS DCL HELP command. Without this feature we would have been forced to create our own on-line help capability. We have used this to provide help for every command and every error message.

The fact that the DCL command language interpreter may be incorporated into user written processes has allowed us to implement command files for automatically controlling and monitoring the experiment while at the same time allowing the experimenter's terminal to be free for other functions. Although this resembles the VAX/VMS batch facility, we have incorporated several features to interact with the data acquisition environment which are not available in batch. This is discussed in more detail in a subsequent section.

The overhead associated with the integrating of local commands into the the DCL command tables can also be significant. This problem is particularly severe for cases in which the DCL tables change and all

local commands must be added to the new tables. By automating this operation we have reduced the overhead to a minimum.

Data Acquisition States

In order to provide a more foolproof system for data acquisition we have incorporated the concept of "data acquisition states" into the software. There are four possible data acquisition states in the system: INITIALIZE, HALT, RUN and PAUSE. This structure has been designed to provide protection from commands which could destroy the integrity of the experimental data if executed at an inappropriate time. For example, hardware setup commands are restricted to the INITIALIZE state. However, recognizing that situations will arise when this type of protection will have to be bypassed, each data acquisition command may employ the qualifier /PERMITTED which allows the command to be executed in any data acquisition state. The use of /PERMITTED on a command, however, must first be enabled by a privileged user.

Each data acquisition state is designed to have a specific subset of data acquisition commands operate within it. The structure of the data acquisition states and the method of going from one state to another is illustrated in Figure 1.

Fig. 1 The data acquisition states and the commands which change from one state to another.

The INITIALIZE state is entered when the data acquisition system is activated and is used primarily for experiment setup. Here the various parameters of the modules are defined, inserted into the experiment data base and the modules are loaded with their operating parameters. Memory which is required for data acquisition is allocated in this state from either FASTBUS histogramming memory or from a VMS global section.

In the HALT state the experiment is inactive between runs. In this state the experimenter can modify experimental parameters in preparation for the next run.

The experiment acquires data only in the RUN state. While in this state the experimenter is not allowed to change any of the operating parameters of the experiment. Only commands which read the experiment parameters from the data base to list them and the command which stops the data acquisition are allowed.

The PAUSE state is used to momentarily stop the data acquisition. It is from this state that the experimenter is required to save his data on disk or specifically execute a command which allows the experiment to be ended without saving the data. One of these operations must be done prior to entering to the HALT state. Here experimental conditions may be examined if a problem is suspected. The user can perform intermediate saves of the data from this state. Saving of data while running is not permitted because data acquisition buffers in the hardware have not, in general, been emptied.

Command Structure

We have adopted a command structure which we expect will not conflict with DEC commands, current and future. Since DEC usually uses verbs as command names we have chosen to use nouns as commands with a qualifier (referred to as a "syntax" qualifier) that determines the action. Each of these nouns with a syntax qualifier becomes, in effect, a command which invokes a program to perform the function. Other qualifiers may also be present in addition to the syntax qualifier.

In designing the structure of the data acquisition commands we have elected to avoid strings of positional arguments, i.e. parameters in the DEC terminology. The positional dependence of parameters tends to be confusing and error prone. Thus we have attempted to use parameters sparingly (usually only one) preferring instead to convey most of the command options with qualifiers. Except for the syntax qualifier which must immediately follow the command name, qualifiers are position independent on the command line. In addition the name of the qualifier provides some documentation of its function. For example, the qualifier /SUBADDRESS=4 refers to a CAMAC subaddress.

To illustrate a typical command which defines a FASTBUS or CAMAC module type the command consists of a command name which is the same as the name of the module and only one of several syntax qualifiers which determine the basic functions. For example, the time of flight module with four multiplexed inputs is called a TOF4. The command has the following basic forms:

```
$ TOF4 /ALLOCATE
$ TOF4 /INITIALIZE
$ TOF4 /LIST
```

```
$ TOF4 /MODIFY
$ TOF4 /NEW
$ TOF4 /VERIFY
```

The /ALLOCATE syntax qualifier establishes space in the data base for a description of the module and places the module's values there. These parameters include the module's logical address, the geographical address, the address space reserved for the module and the user defined module name. The use of the /ALLOCATE qualifier is restricted to the INITIALIZE state.

The /INITIALIZE syntax qualifier loads values from the data base into the module. Typical FASTBUS parameters include the logical address and other control space registers specific to that type of module. In the case of CAMAC modules the initialization step is much simpler. The /INITIALIZE qualifier is allowed only in the INITIALIZE and HALT states.

The /LIST syntax qualifier lists module parameters from the data base to a file or on the user's terminal. No values are read from the module. The /LIST qualifier may be used in any state.

The /MODIFY syntax qualifier allows the user to change the various parameters characterizing a previously allocated hardware module, such as the user defined name for the module or its geographical address. The /MODIFY qualifier only changes the data base. In order to reload the module with the new parameters from the data base the /INITIALIZE qualifier must subsequently be used with the command. The /MODIFY qualifier can be used in the HALT and INITIALIZE states.

The /NEW syntax qualifier is used to erase all previous definitions in the data base belonging to this module type. This is normally done before setting up the experiment. The /NEW qualifier may only be used in the INITIALIZE state.

The /VERIFY syntax qualifier is used to compare the contents of the data base with the actual values in the module. The module is read and the hardware values are compared to those in the data base. Any discrepancies are reported to the user. We expect this option to be particularly valuable when the user detects anomalies in the data and wishes to check the module's operating parameters. The /VERIFY qualifier may be used in the INITIALIZE, RUN and PAUSE states.

Example

An illustration of the command structure and command file usage is given in the example below which progresses from the beginning of an experiment through to completion. Commands with their appropriate syntax and qualifiers are given in uppercase while parameters and string input on qualifiers is in mixed case.

```
$!
$! Example experiment command file
```

```

$!
$ HEAP /NEW
$ BEGIN INITIALIZE
$!
$ SCALER /ALLOCATE /CRATE=1 /SLOT=2 /SUBADDRESS=1 Time
$ SCALER /ALLOCATE /CRATE=1 /SLOT=2 /SUBADDRESS=2 Starts
.
.
$ TOF4 /ALLOCATE /SLOT=12 Left_Detectors
$ TOF4 /INITIALIZE Left_Detectors
.
.
$ MEMORY /ALLOCATE /DIMENSION=2048 /CLASS=FASTBUS-
/TITLE="Background Data" Background[3:4]
$ MEMORY /ALLOCATE /DIMENSION=2048 /CLASS=SCRATCH-
/TITLE="Analysis of 12C data from run D667" Carbon_Data[3:4]
$ MEMORY /ALLOCATE /DIMENSION=2048 /CLASS=SCRATCH-
/TITLE="Normalized carbon data" Normalized_Data[3:4]
.
$ END INITIALIZE !Enter HALT state
.
.
$ MEMORY /CLEAR /ALL !Clear data storage
$ SCALER /CLEAR /ALL !Clear all scalers
$ IDENT/RUN=D668 !Define a new run

$ IDENT/TITLE="Boron time-of-flight data"

$ BEGIN RUN !Start taking data
.
.
$ PAUSE !Enter PAUSE state
$ SAVE /DATA /ABSTRACT PARAMETERS=ABSTRACT.ADF /CATALOG
$ OPERATION "Normalized_Data=2*(Carbon_Data/Background)+10"
$ END RUN !Enter HALT state

```

Fig.2. An example of an experiment command file.

The first command HEAP/NEW initializes the experiment data base. The command, BEGIN INITIALIZE, places the experiment in the INITIALIZE state. No hardware action is taken.

The next command, SCALER/ALLOCATE, allocates records in the data base and places the information specified on the command line in the data base. The scaler is located in CAMAC crate 1, slot 2, subaddress 1 and is given the name "Time". Subsequently another scaler is allocated with the name "Starts" in CAMAC crate 1, slot 2 and subaddress 2.

The TOF4/ALLOCATE command allocates space in the data base for the definition of a TOF4 FASTBUS module with the name of "Left Detectors" and places the parameters for such a module in the allotted space. The /SLOT qualifier indicates the FASTBUS geographical address is 12. The command TOF4/INITIALIZE loads the internally assigned logical address into the module and enables the module.

The first MEMORY/ALLOCATE command reserves space in the FASTBUS histogramming memory for 3 groups of 4 histograms, each of size 2048 channels. The name "Background" and title "Background data" are associated with this group of histograms.

The last two MEMORY/ALLOCATE commands allocate 3 groups of 4 histograms, each of 2048 channels as scratch data areas from a VMS global section. The names of the areas are "Carbon_Data" and "Normalized_Data", respectively.

END INITIALIZE completes the initialization phase of the experiment and places the data acquisition system in the HALT state. END INITIALIZE does not perform any action on the hardware.

After the data acquisition environment is defined the data acquisition is initiated. The MEMORY/CLEAR/ALL command clears all the data storage and the SCALER/CLEAR/ALL command clears all scalers which have been defined. The run is given the identification D668 and the title "Boron time-of-flight data". The BEGIN RUN command starts data acquisition.

At the end of the run the experiment is stopped and placed in the PAUSE state with the PAUSE command. The data are saved on disk with SAVE/DATA. Here the qualifier /ABSTRACT PARAMETERS indicates that the experimental parameters (such as scalers etc.) to be saved along with the data are enumerated in the file ABSTRACT.ADF. The /CATALOG qualifier specifies that the data be automatically sent from the data acquisition computer over the network to the data archive on central computer. At this point the data comes under the protection of the system and cannot be deleted by the user. The user may avoid archiving the data by specifying /NOCATALOG on the command. In this case the data is saved on the local disk only.

After the data is saved the OPERATION command performs the channel by channel operation of dividing the data named Carbon Data by the data named Background, multiplying by 2 and adding 10. The results are placed in the histograms of the storage area called Normalized Data. Since no particular group or histogram is specified, the operation is performed for each of the 3 groups of 4 histograms. The OPERATION command supports algebraic syntax for data area arithmetic.

The experiment then enters the HALT state with END RUN and the command file terminates. At this point the experimenter would normally change the experimental conditions to prepare for the next run.

Additional Features

A method of running data acquisition commands using detached processes has been developed which allows the experimenter to specify a list of DCL command files to be run to control the course of the experiment. This technique frees the user's terminal for other uses. Users are notified on the terminal when errors occur. At all times the user has manual override of this automated process through DCL commands. The

internal structure of this feature of the system is described in another paper given at this conference [4].

Two commands control the submission of DCL command files for automatic execution. The command `QUEUE RUNS/EDIT` allows editing of a file known as the "Pending-runs-queue" which contains the names of the command files which, when run, determine the course of the experiment. This editing is performed by using a standard text editor, normally the EDT editor. The command verifies that all entries are syntactically valid and that files which are entered actually exist. While the Pending-runs-queue is being modified no command files will be submitted.

The command `EXECUTE RUNS/START` activates a detached process which has access to the DCL interpreter and executes the DCL command files. This process executes command files from the Pending-runs-queue until the queue is empty. The `EXECUTE RUNS` command provides additional options which allow the user to stop, suspend or resume the execution of the queue. Since this operation is a detached process it can continue regardless of the activity on the user's terminal, and even if the user logs out.

FASTBUS diagnostic commands to perform the FASTBUS primitive operations of arbitration cycle, address lock, FASTBUS read and write, release address lock and reset bus have been implemented in DCL. In addition, commands to perform complete FASTBUS read and write cycles have been written.

Summary

In developing the commands for the data acquisition system we have attempted to follow standard DEC DCL usage whenever possible in order to make the system easier to understand by our experimenters. Documentation within the data acquisition system is covered in another paper given at this conference [5].

We have found that the use of DCL has saved an enormous amount of time compared to the effort which would have been expended in writing our own command language interpreter, help facility and message facility. Also the DEC enhancements of VMS and DCL which occurred with the introduction of VAX/VMS V4.0 have resulted in a simplification of our programs, particularly in the area of input validation.

DCL execution is slow as are almost all interpreters. However most experimenter activity on the terminal and in automatic command execution occurs at non-critical times in the experiment, usually at the end of a run when experimental parameters must be changed. Also the VAX computers in our system is really not involved in real-time activities. These tasks have been relegated to the FASTBUS environment.

Acknowledgments

This work was supported in part by the U.S. Department of Energy.

References

- [1] R. O. Nelson, D. M. Barrus, G. Cort, J. A. Goldstone, D. E. McMillan, R. V. Poore, and D. R. Machen, "FASTBUS Data Acquisition System for Neutron Time-of-Flight Measurements", IEEE Transactions on Nuclear Science, Vol. NS-32, NO. 1, February 1985.
- [2] M. W. Johnson, J. A. Goldstone and A. D. Taylor, "Requirements for Data Acquisition and Analysis for Condensed Matter Studies at the Weapons Neutron Research/Proton Storage Ring Facility", Los Alamos Report LA-9099-MS (1982).
- [3] D. McMillan R. Nelson, R. Poore, M Meier, and R. Woods, "P-9 Proposed Weapons Neutron Research Facility Data Acquisition System for the PSR Era", 1982.
- [4] G. Cort, J. A. Goldstone, R. O. Nelson, R. V. Poore, L. Miller, and D. M. Barrus, "Dynamic Data Structures and Concurrency in a real-time Data Acquisition System," presented at the Fourth Real-Time Conference on Computer Applications in Nuclear and Particle Physics, Chicago, Illinois, May 20-24, 1985.
- [5] G. Cort, J. A. Goldstone, R. O. Nelson, R. V. Poore, L. Miller, and D. M. Barrus, "A Development Methodology for Scientific Software," presented at the Fourth Real-Time Conference on Computer Applications in Nuclear and Particle Physics, Chicago, Illinois, May 20-24, 1985.