PFC/JA-96-22

MDSplus Data Acquisition System

J.A. Stillerman, T.W. Fredian, K.A. Klare[1], G. Manduchi[2]

June 1996

[1]Los Alamos National Laboratory, Los Alamos, NM 87545.
[2]CNR, Padua, Italy.

Submitted to Review of Scientific Instruments.

This work was supported by the U. S. Department of Energy Contract No. DE-AC02-78ET51013. Reproduction, translation, publication, use and disposal, in whole or in part by or for the United States government is permitted.

MDSplus Data Acquisition System

J.A. Stillerman, T.W. Fredian MIT Plasma Fusion Center; K.A. Klare Los Alamos National Lab; G. Manduchi CNR, Padua, Italy.

MDSplus a tree based, distributed data acquisition system, was developed in collaboration with the ZTH Group at Los Alamos National Lab and the RFX Group at CNR in Padua, Italy and is currently in use at MIT, RFX in Padua, TCV at EPFL in Lausanne, and KBSI in South Korea.

MDSplus is made up of a set of X/motif based tools for data acquisition and display, as well as diagnostic configuration and management. It is based on a hierarchical experiment description which completely describes the data acquisition and analysis tasks and contains the results from these operations. These tools were designed to operate in a distributed, client/server environment with multiple concurrent readers and writers to the data store. While usually used over a Local Area Network, these tools can be used over the Internet to provide access for remote diagnosticians and even machine operators. An interface to a relational database is provided for storage and management of processed data. IDL[1] is used as the primary data analysis and visualization tool.

Introduction

MDSplus provides a set of tools for performing data acquisition and analysis for pulsed experiments. It is designed to keep all experimental data, analysis results, experiment configuration and setup information together in an organized fashion. The data for each pulse of the experiment is stored in a hierarchical tree structure much like a file system. This experiment model, enables the users to organize very large data sets. Some of the MDSplus documentation is available on the Plasma Fusion Center WWW pages at ***http://cmod2.pfc.mit.edu/mds/mdsplus.html***.

Data Storage

All of the information about each instance of the experiment, or pulse is stored in a hierarchical structure which is accessible to the whole group of researchers. This comprehensive and open approach both enables and encourages collaboration among the scientists. Since all of the information is stored in one place, it is easy to keep track of exactly what was done. The hierarchy provides organization for all of these pieces of information. This becomes critical as the number of data items grows; the CMOD experiment currently has more than 30K nodes in its hierarchy.[2] Since all of these items are stored in the same way, one set of tools can be used to manipulate both setup information and results from the experiment.

After data acquisition for a pulse is completed the tree for that shot contains:

- The setup information for Diagnostics
- Calibration and geometry information
- The setup information for Data Acquisition
- The machine control settings
- Task scheduling
- The raw data from engineering systems and diagnostics
- The processed data

Paths, Members and Children

MDSplus trees contain two types of nodes, 'children' and 'members'. Children or branch nodes are used to define the structure of the tree. These nodes can not contain any data. They are analogous to directory files in a computer file system. The second kind of nodes, called members, actually contain the data. They are analogous to the files in a computer file system. These *members*, may also have members and children below them as well as containing data.

A node in a tree can be referred to by either an absolute or relative path. Absolute paths begin with a tag reference followed by a series of branch specifiers. A typical absolute path *\cmod::top.engineering.oh1:volt* contains a tag *\cmod::top* followed by a list of branches separated by dots *.engineering.oh1* followed by a leaf reference delimited by a colon *:volt*. Relative paths are relative to some current default location. If the current default is *\cmod::top.engineering.oh1* the above node could be referred to simply as *volt* and *\cmod::top.engineering.oh2:volt* could be referred to as *.-.oh2:volt*.

Data types

In addition to a wide selection of primitive data types (byte, word, long, float, double, complex, etc…) MDSplus supports complex types to store commonly used combinations of items.

- Signal - used to associate independent axes with arrays. They consist of result data, raw data and one or more descriptions of axes. Subscripting can be done on either the independent parameters or the array indices.
- Units - can be associated with any node in the hierarchy, or with any term of a complex data item. For example, a digitizer channel would have Volts, Counts, and Seconds associated with its' processed, raw, and first dimension parts.
- Range - composed of a start, and end, and an increment. They are used in the descriptions of clocks. Multi-frequency clocks are described by ranges in which the terms are arrays.
- Dispatch - stores a description of when a task should be performed and which server should perform it.
- Task - stores a description of something to be done. For example, a task could say to perform an ARM operation on a digitizer or to SUBMIT a particular batch job.

- Action - used to describe automatic data acquisition and analysis activities. They are made up of a dispatch and a task.
- Expressions - Any node in a tree, or any term in the above complex types can contain an expression instead of a primitive data object. These expressions are evaluated by the built-in expression evaluator when the node is referenced. This is the same expression evaluator, called TDI, which is described in the Data Analysis section of this paper.

Trees and Subtrees

A tree can be made up of a set of subtrees which are each stored in their own set of files. When a hierarchy is opened all of its subtrees are automatically opened, making it appear as one integrated tree.

The individual subtrees can also be accessed independently from the rest of the hierarchy. This means that one does not have to restore an entire 80 Mbyte shot from the data archive system in order to look at one or two traces. Using this subtree mechanism, write (or read) access to a subset of the tree can be restricted to an authorized group of users.

Editing vs. Traversing

Each node in the tree has three kinds of information associated with it. There are a set of read-only attributes including its name, any tag names associated with it, and its location in the hierarchy; a set of fixed length read-write attributes such as owner, date written, data type, data length, location in data file; and, if there is data associated with the node, a variable length record which contains the actual data. When a tree is accessed it can be opened in two ways. Normally, it is opened 'not for edit' which gives read-only access to the structure of the tree and read-write access to the data. In order to make structural changes to the tree, adding or removing nodes, renaming nodes, etc... the tree must be opened for 'edit'. In the normal mode, multiple concurrent readers and writers from all computers in the cluster are supported. Only one user can edit a tree at a time. For this reason 'editing' changes, as opposed to 'value' changes are normally done off line.

Two tools are provided for viewing and manipulating the trees. The TRAVERSER is an XWindows / Motif based application for browsing the hierarchy. TCL, or Tree Command Language, is a command interpreter with commands like ***'directory'***, ***'set default'***, ***'put'***, ***'show data'*** etc... See Figure 1 for a typical TRAVERSER screen.

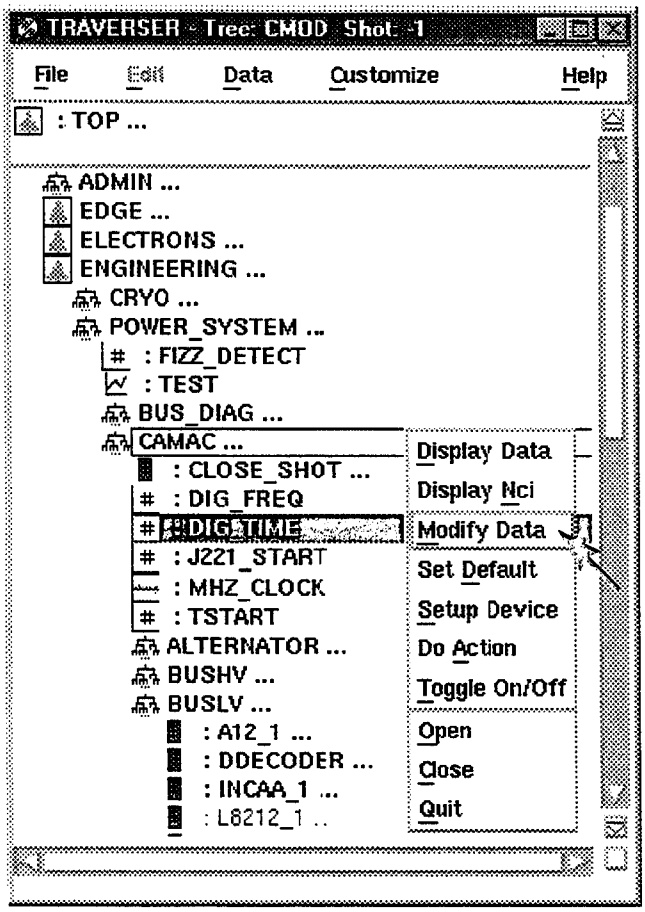

Figure 1 Traverser

From within the TRAVERSER, when 'modify data' is selected a data modifier appropriate to the data type of the node is displayed. Figure 2 shows a typical data modifier screen for a node containing a simple expression.

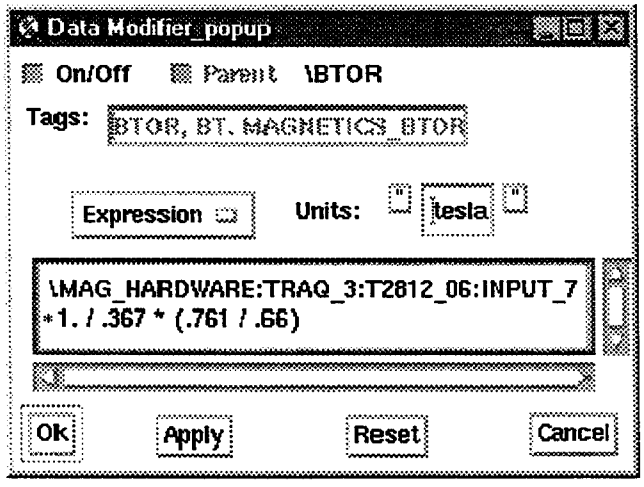

Figure 2 Expression Modifier

Data Acquisition

The MDSplus data acquisition cycle is driven by the action descriptions stored in the experiment model. These actions tell the dispatcher what tasks should be done by which servers and in what order. After the cycle completes, a complete record of what was done is stored back into the model along with the data from the experiment.

MDSplus currently supports more than 60 different data acquisition device types. New diagnostics composed of these can be added to the experiment model in just a few minutes using the TRAVERSER.

Adding support for new types of data acquisition hardware usually takes only one or two days of effort depending on how complicated the device is. One routine needs to be written for each operation the device will support, and a setup form needs to be assembled. Stubs for the routines are generated automatically by the GEN_DEVICE utility. A typical ADC requires INIT, TRIGGER, and STORE operations. The user interface can usually be implemented without any code, using the special widget set developed for this purpose and a graphical GUI builder.

Dispatcher/Server

The tasks which make up the data acquisition cycle are managed by the DISPATCHER. This application is an interpreter with a set of verbs for constructing pulse files, collecting together actions, and overseeing their execution by SERVER processes. It is implemented using the generic MDSDCL interpreter. DECNET is used for all inter-process communications.

Actions in the tree are scheduled by a phase and sequence number. An installation specific state machine causes the DISPATCHER to execute each phase of the shot cycle in turn.[2] A typical shot cycle might look like:

- INIT
  - copy model to pulse, and open it.
  - extract all actions from pulse into dispatch table.
  - dispatch all initialization operations, and wait for completion
- PULSE
  - fire high speed timing system
- STORE
  - dispatch all store operations
  - dispatch all analysis operations
  - dispatch 'close' to all servers

Traverser for Device Setup

The TRAVERSER, described above, is used to add diagnostics to the hierarchy and to configure existing devices. The setup form for a typical CAMAC module can be seen in Figure 3.

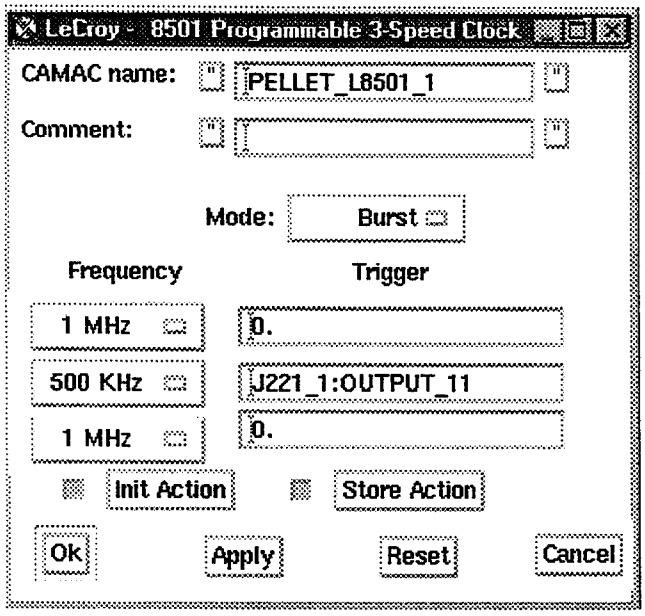

Figure 3 Programable Clock Setup

IDL for Custom Data Acquisition and Control

Custom interfaces to the experiment model are implemented in IDL. The user of a diagnostic may wish to specify their settings in a different way than they are stored in the experiment model. It is very easy, using the IDL widget and MDSplus interfaces, to put together a screen tailored to the task at hand. These range from fairly simple screens where the user sets their digitizing parameters (start time, rate, number of samples) to PCS which is used for plasma shaping and parameter control.[3]

Data Analysis

All of the interfaces to data in MDSplus are based on an expression evaluator called TDI. TDI expressions can be used in user analysis programs in C, FORTRAN, or IDL, interactive analysis and display programs like the SCOPE and are used to implement various aspects of device and shot cycle support.

Programming Interface

The IDL and compiled language interfaces to MDSplus data consist of four routines.

1. MDS$OPEN , experiment-name, shot-number
2. MDS$CLOSE [, experiment-name, shot-number]
3. MDS$VALUE, expression [, arguments]
4. MDS$PUT, node, expression [, arguments]

Given these four a user has complete read and write access to an MDSplus data store. The optional arguments to MDS$VALUE and MDS$PUT are values to be substituted into the expression when it is written to the tree. For example, given an array of data called *values* one could put an expression containing it in a node called temperature as follows:

```
MDS$PUT, 'TEMPERATURE', '$*1.76+1E4', values
```

One argument is substituted for each '$' found in the expression.

TDI transparently operates on arrays as well as scalars, and does automatic conversion to compatible data types. It includes control structures, function declarations, assignment operators and hundreds of built-in operations.
See ***http://cmod2.pfc.mit.edu/help/@mdsplus/tdishr*** for a complete description of its' functionality

.

DWSCOPE

The DWSCOPE is an XWindows-motif based data display application. Each DWSCOPE can display up to sixty-four traces in four columns. These panels can be updated automatically during the shot cycle when new data becomes available. For each panel the user specifies expressions for the X and Y axes and various display parameters. These DWSCOPE panels can be cut and pasted between DWSCOPEs. Figure 4 shows a simple DWSCOPE display. The mouse is used to zoom, pan, and point in the display.

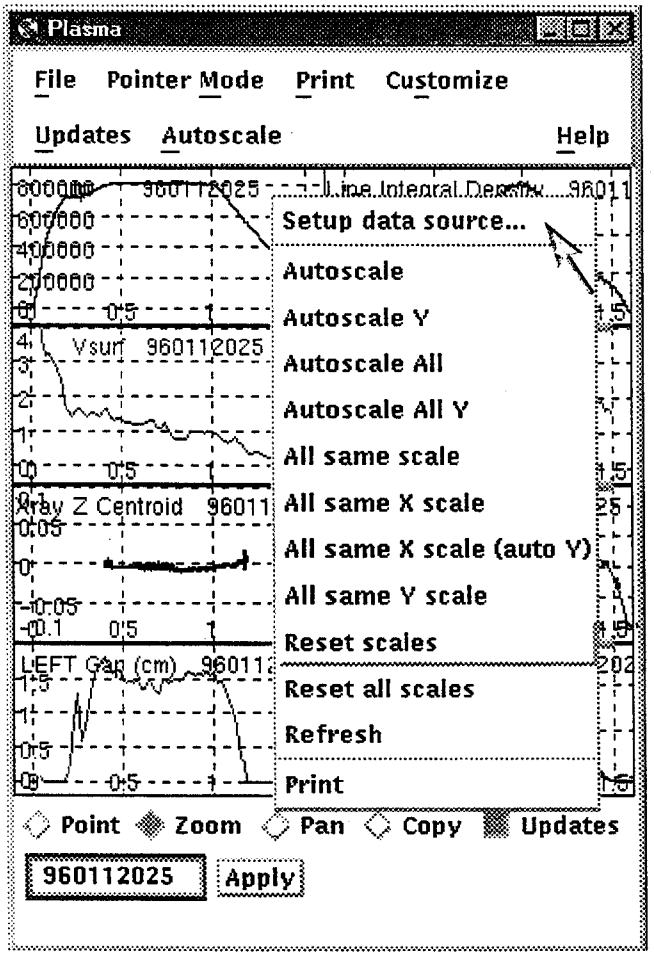

Figure 4 DWSCOPE Display

IDL (Research Systems Inc.)
IDL is a commercial data analysis and visualization package. MDSplus provides the interface to data described above and an SQL interface to get and put records to RDB relational databases. It has become the standard tool for data analysis and display applications in MDSplus.

IDL procedures are invoked from batch jobs during the shot cycle to automatically process raw experiment data and store analyzed data back into the tree. Figure 5 displays a typical automatic analysis routine. IDL's automatic vector arithmetic makes the computation of *poh* easy to express.

```
;procedure to calculate oh power
pro poh,shot,poh,t
  ;open the analysis tree, get efit data
  mds$open,'analysis',shot
  li = mds$value('\efit_aeqdsk:ali')
  majr = mds$value('\efit_aeqdsk:rmagx')
  ip = mds$value('\efit_aeqdsk:pasmat')
  psibry = mds$value('\efit_ssibry')
  t=mds$value('dim_of(\efit_geqdsk:pcurrt,2)')
  vs =6.283*deriv(t,psibry) ;surface voltage
  l = 6.283*majr*li*1.e-9   ;plasma inductance
  vi = l*deriv(t,ip)        ;inductive voltage
  vres = vs - vi            ;resistive voltage
  poh = ip*vres
  ;put results back into tree and close file
  mds$put,'\analysis::top.results:poh',$
          'build_signal($,*,$)',poh,t
  mds$close
  return
end
```

Figure 5 Sample data analysis routine

WLOGBOOK is an IDL application for accessing the electronic logbook implemented in RDB[4]. It also provides a general interface to tables in relational databases. Each site augments the tables which make up the logbook part of the database with tables which are relevant to their particular needs.

IDL is used to implement many site specific applications having to do with the data system. At CMOD, it is used for everything from playing the automatic audio announcements during the shot cycle to controlling the main plasma parameters for the machine.

Conclusion

MDSplus is currently in use at fusion sites in the US, Italy, Switzerland, and South Korea. Over its five years of operation, it has evolved with the available computer technology and the needs of its users. It has proven itself an effective tool for data acquisition, analysis and management. This system could not have been developed without the help and encouragement of the research groups at the collaborators sites.

---

[1] IDL is a registered trademark or Research Systems Inc.
[2] T. Fredian, J. Stillerman, M. Greewald, Data Acquisition System For Alcator CMOD, This conference.

[3] S. Horne, M. Greenwald, I. Hutchinson, S. Wolfe, G. Tinios, T. Fredian, J. Stillerman, Performance of the C-Mod Shape Control System, Proceedings of the IEEE 15th Symposium on Fusion Engineering, 782-785 (1993)

[4] Oracle RDB registered trademark of Oracle Corporation