ISBN: 978-3-95450-254-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS1991-S06SA01

Overview of the Next Generation of Fermilab[1] Collider Software

Brian HENDRICKS, Robert JOSHEL
Fermilab, P.O. Box 500, Batavia, IL 60510, USA

Abstract

Fermilab[1] is entering an era of operating a more complex collider facility. In addition, new operator workstations are available that have increased capabilities. The task of providing updated software in this new environment precipitated a project called Colliding Beam Software (CBS). It was soon evident that a new approach was needed for developing console software. Hence CBS, although a common acronym, is too narrow a description. A new generation of the application program subroutine library has been created to enhance the existing programming environment with a set of value added tools. Several key Collider applications were written that exploit CBS tools. This paper will discuss the new tools and the underlying change in methodology in application program development for accelerator control at Fermilab.

I. MOTIVATION

Digital VAXstations running X-windows under VMS have replaced the PDP-11s formerly used to control the accelerators. This more powerful platform coupled with the demands of more complicated Collider operation led us to move from an approach of single application-per-need to using fewer applications that draw on a large toolbox of resources. Application programs are now viewed as hooks into a pool of integrated tools. At the same time we must maintain compatibility, while encouraging migration to new tools, for the existing application programs which number in the hundreds. This is done by providing calling sequences which are not too divergent from the existing ones. The new console platform also opened the door for the use of C as an application programming language. Calling sequences are often provided both in call-by-reference for the FORTRAN users and call-by-value for the C users.

[1] Operated by Universities Research Association for the Department of Energy

II. OVERVIEW

All the new tools are layered on top of the older lower-level routines. These subroutines reside in a shareable image. This allows easy growth of a large number of routines without affecting the application programmer. This was necessary so that application development could be done in parallel with the maturation of the CBS environment. The tools handle file access, data acquisition, graphics screen management, window management, inter-program communication and error logging facilities. These utilities also provide their own logging and statistics that are viewable by the user during program execution. Tools that access centralized facilities, such as reading a database, cache information to reduce the load on the centralized processes and the network. Any of the tools with a visual interface follow standards. This provides a consistent user presentation to the operators. This is a more effective way to enforce user-standards, rather than administrative dependent approaches that have failed in the past.

III. DATA ACQUISITION

The first major component in the CBS utilities involves input and output to accelerator hardware as well as reading database information concerning that hardware. This involves reading, setting, controlling, and scaling values as well as handling alarms and miscellaneous device attributes. The previously available interface routines required separate requests for real-time raw data as well as stored data from the database in order to read or set data in engineering units. This required seven low level function calls to retrieve or set a single value in engineering units. In addition to the function calls, additional code was required to perform such necessary tasks as retrying data retrieval until the data is actually received. All of this functionality has been replaced by a single, simple function call. For lists of devices the procedure is only slightly more complex in that there is a function to build the list of devices and a second function to read or set the list.

The database information for scaling and necessary interface to front end software is cached locally. This reduces redundant database access and network traffic. The data acquistion routines perform the access and caching such that it is

Content from this work may be used under the terms of the CC BY 4.0 licence (© 1992/2024). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

ISBN: 978-3-95450-254-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS1991-S06SA01

Content from this work may be used under the terms of the CC BY 4.0 licence (© 1992/2024). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

transparent to the application programmer. The cached information exists for the life of the program run. The database is updated so infrequently that stale cached data is not a problem.

The programmer and end user can peek at the data acquisition activity through an application that polls embedded statistics modules in the data acquisition routines. This peeker program runs on the console concurrently with the program to be analyzed. The peeker will display counts of function calls and errors. It will also show the number of devices and lists of devices being read and set as well as the frequency of retrievals. Additionally, timing statistics are shown for the data accesses being performed.

IV. FILE MANAGEMENT

There is a large group of applications that use shared read/write access files. A set of routines was created to help the programmer manage opening, reading, writing, and closing files in a simple, convenient fashion. As with the data acquisition routines, file operations were simplified by reducing the number of function calls required to accomplish them. A file peeker program was written that is similar to the data acquisition peeker. It displays numbers of function calls and errors. It also displays which file and record was last read and last written and any error associated with the access. In addition, timing information is displayed for communications with the central file server process.

V. SCREEN MANAGEMENT

The VAX console environment provides three X windows for each application program. These windows are managed by two manager processes that perform all the direct X protocol. This was done for compatibility with existing applications. The main window permits only character cell access. The other two windows allow pixel addressing and are used primarily for plotting. Screen management facilities were created for each of these types of windows.

The previous interface routines for the alphanumeric window provided little in the way of managing subwindows. There were window create and delete routines which simply saved and restored blocks of text on the screen. Anything beyond that was handled directly by the application program. This led to crowded and confusing displays and a myriad of user interfaces as programmers grappled with the problem of displaying a great deal of information in a limited space.

The new routines support input and output to multiple tiled or overlapping windows. Input and output is clipped to the window being addressed. In the case of overlapping windows, text written to an occluded character cell is saved and is refreshed when and if that character cell becomes exposed. These windows may be moved or resized by the user and/or under the control of the application program. Routines were also created to make it simple for existing nonwindowed applications to make use of the new windowed routines. In the future this transition may be made seamless by modifying the underlying I/O routines.

Support of windows with vertical scrolling capability was provided to release the application programmer from the limits of the size of the alphanumeric window. Routines exist to create and manage a scroll bar and draggable indicator requiring no support code from the application program. Lines scrolled out of a window are buffered, freeing the application from needing to remember scrolled text.

In addition to the basic window support, higher level screen management tools have been added. Menu and menu bar routines, numeric and textual input dialog boxes, logical dialogs, and message windows have been provided. There are also utilities to create and manage logical switches as well as to handle highlighting of text regions.

Since the pixel addressable windows are used primarily for plotting, a suite of plotting interface routines were created. The routines allow the programmer to define a plotting window in terms of fractions of the background window. After the region is defined, the scaling function and scaling limits can be selected for both axes. Plot labelling and plotting attributes can also be defined. An application program was also created to allow the user to enter window definition parameters and view the resultant window interactively. Once the desired window is constructed, the program can be instructed to generate the source code to create the plot window displayed. Additional routines have also been created to save plotted data and then to perform statistical and fitting operations on the saved data.

VI. PROGRAM TAPE RECORDING

One of the primary applications which had to be created for the operation of the Collider was the Colliding Beams Sequencer. This program carries out all of the steps to load the Collider with particles and bring them into collision. Some of these steps are simple and are contained within the sequencer, but some are more complicated and require the sequencer to invoke other application programs to perform the task. Modification of applications to run under sequencer control in the past has often been complicated and time consuming. It was felt that a more flexible means was needed for running programs under the control of another program. It was also important to have a method which would allow for easy modification. The facility implemented was a software tape recording system. A user enters the desired program in a special recording mode and then proceeds to perform the desired task manually. All the steps are recorded automatically in a file and are assigned a unique file name. The file can then be

ISBN: 978-3-95450-254-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS1991-S06SA01

triggered by the sequencer (or the application itself), and the application program will perform the same steps originally carried out by the operator.

VII. ERROR HANDLING/LOGGING

Checking and reporting error conditions is one of the most important needs for application programs used for accelerator control. Operators need to know if settings and readings of devices are successful. In addition, it is useful to be able to reconstruct events that have occurred. The error handling routines check error codes, expand them to give text descriptions, and save errors encountered during execution of the program. Logging to circular log files can be done automatically by the error display functions. A program user can examine the log history in a scrollable window with many different text filter and search modes. The programmer can choose to incorporate a separate log for each instance of the program or a single log so that all uses of a program are shown in one log. Separate programs working together can also share a single log with time-ordered entries maintained by the logging utility.

VIII. INTERPROGRAM COMMUNICATION

To support the CBS utilities peeking activities and the Colliding Beam Sequencer communication, a set of interprogram communication routines were created. These use VMS mailboxes to allow queues of AST deliverable messages between indpendent processes. One simply creates a mailbox with an optional AST address and then sends or receives messages from that box. Programmers can now create program suites that can work independently or communicate directly with each other. The program used to smooth the Tevatron orbit will, if needing a lattice, start a program to generate the lattice. In turn it can then poll for the results. This allows large amounts of data to flow between cooperating programs at a high rate.

IX. UTILITIES

One of the CBS tools that demonstrates the integration of all the utilities is the Utilities Window. Through a menu driven interface the user can make screen copies, start plotting packages, invoke log displays and set timeouts for data acquisition and file accesses. The user can also interact with the error reporting buffer by clearing or viewing past messages. A window that allows display, reading and setting of accelerator parameters is also available. This parameter window is customized by the user for use with a particular program.

X. CONCLUSIONS

The results of recent accelerator studies indicate that the CBS approach has been successful thus far. The sequencer, orbit smoothing program, and the Tevatron ramp calculation and loading programs have been used successfully and have been well received by the user community. The consistency in user interfaces has allowed users to learn to use these complicated programs in a short period of time. In addition, programs have been developed using the CBS tools to create other applications not directly related to collider operation. Inexperienced programmers have been able to construct involved applications not only successfully, but also in short periods of time. The Collider applications have also been able to be extensively modified in response to user needs in a short period of time with little debugging.

The work continues on this project. There are many more topics to be addressed such as better handling of the data associated with the accelerator clock system and complicated table devices as well as context sensitive help. Ultimately, the goal is to bring all of this under the umbrella of a resource editor or code generation system to allow for even more rapid and error-free creation of accelerator applications.

XI. REFERENCES

[1] Cahill, K.J. and Smedinghoff, J.G. (1989). Converting the Fermilab Accelerator Control Consoles to X-Window Workstations, Proceedings of the International Conference on Accelerator and Large Experimental Physics Control Systems, pp. 442-445

[2] Cahill, K.J. and Smedinghoff, J.G. (1991). Exploiting the X-Window Environment to Expand the Number, Reach, and Usefulness of Fermilab Accelerator Control Consoles, these proceedings

Content from this work may be used under the terms of the CC BY 4.0 licence (© 1992/2024). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI