The High Level Programmer and User Interface of the NSLS Control System *

Y.N. Tang, J.D. Smith and S. Sathe†
National Synchrotron Light Source, Brookhaven National Laboratory, Upton, NY 11973

Abstract

This paper presents the major components of the high level software in the NSLS upgraded control system. Both programmer and user interfaces are discussed. The use of the high-speed work stations, fast network communications, UNIX system, X-window and Motif have greatly changed and improved these interfaces.

1 Introduction

The NSLS control system is undergoing a major upgrade and phase II has been completed [1, 3]. Many new capabilities and features have been brought in by the upgrade.

- The fast ethernet communication replaced the slow serial links and the powerful work stations are replacing the old central hosts and terminals. Many programs run 10 to 20 times faster than before. The ring down time for injection becomes much shorter. Many advanced diagnostic tools such as the real time orbit display are emerging.
- The graphical user interface is replacing the traditional textual interface. The X window and Motif have completely changed its look and feel.
- The UNIX excellent programming environment have greatly shortened the life cycle of the software development. Eight major libraries and more than 60 programs have been developed or converted since we started the project at the beginning of 1992.
- The rich set of the UNIX utilities and its flexible file system have helped us in developing a powerful interpreter and NSLS standard file format.
- The reliability of the system is very much enhanced. For example, there are many history processes running in the background at different scheduled frequencies. They never stopped and have not missed any scheduled readings under normal conditions.

Many people in the NSLS Department contributed to this project. Our special thanks go to S. Krinsky and John Keane. Without their encouragement, support and guidance, today's success of the upgrade would not be possible. Thanks also go to the ring managers N. Fewell and S. Kramer and many other people for their support and many very helpful suggestions, opinions and ideas. All members of the computer controls group made their contributions.

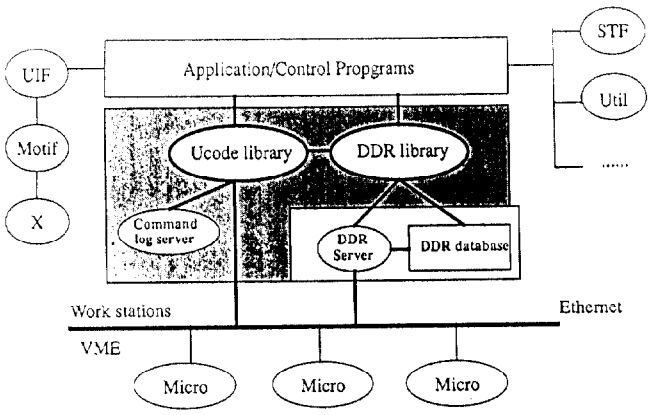
*Work performed under the auspices of the U.S. Dept. of Energy under contract no. DE-AC02-76CH00016.

†Ms. Sathe now in the AGS Department

2 Overview of Work station Software

The upgraded control system is a distributed system and its software consists of three layers (Fig. 1). The VME micros, which control the hardware, constitutes the lower layer. The high level applications talk to these micros through ethernet by calling functions in the ucode (user code) library. The controllable and/or readable system parameters and hardware signals are called devices, which are stored in a home made database — DDR (Device Data Record). The applications access the DDR database through the DDR library. The ucode and DDR constitute the middle layer of the control software.

Figure 1



3 Major Components of System Software

3.1 The DDR Database

We use logical and composite devices, whose records contain logically related parameters such as the setpoint, readback, command and device states, tolerance, high and low limits, error flag, error mask, lock status, in-process flag *etc.* A parameter mask field in the record masks out the parameters not used for a device.

Using logical records, which is a key to to a high-capacity and high-speed control system, drastically reduces the number of devices in our system and the network traffic. We have about 5100 devices now and it may increase to 10000 in the near future. If we did not use the compos-

ite records, the device number might be 10 times or more larger.

The DDR database keeps the addresses and other important parameters of all devices in the system. Any program accessing devices must consult DDR at first. The whole database is loaded into memory (now 0.5 Mbytes) at run time for the fast execution.

A complete set of DDR utilities has been built. A graphical browser/builder allows users to build, inspect, search, sort, save and print the DDR or displayed pages. The DDR library provides the programmer interface.

3.2 The Ucode Library

The ucode library is the interface between the high level programs and low level micros. Every program must call ucode functions when accessing devices. This library was carefully designed and developed because of its importance and is still evolving.

- To achieve high data rates is one of our major goals in the system upgrade. The UDP protocol is used to reduce the overhead caused by the TCP protocol. The reduction of the overhead in the network transport increases the effective bandwidth of the network. Though we spent much effort in the library and real-time monitor [2] to make the communication reliable, this effort is many times rewarded by the benefits we gained. For example, a program may read a set of devices at a 120 Hz rate or more. If the TCP protocol were used, we could only make about 70 Hz.

- To make the library as robust and easy to use as possible has been always emphasized because it s not only used by programmers, but also used by physicists, engineers and others. Generally, one ucode function is enough to write any settable fields of any number of devices or read any field, a commonly-used combination of fields or the whole records of any number of devices from any number of micros. The ucode automatically call functions in the DDR library and divides the user request in multiple packets (a UDP packet consists of at most 1024 bytes) if necessary. The implementation details are completely hidden and ucode does all the "dirty work" and decodes all the read back data. For example, it not only gives the state code of devices, it also encodes the code into "ON", "OFF" or whatever is appropriate. We provide both C and Fortran versions. A physicist or engineer needs only several hours to learn how to use it.

- For each ucode call of reading or writing devices, the ucode uses an internal buffer which is filled with the access status code for each device upon the completion of the call. The code is either SUCCESS (0) or an error number. If the function returns an error, one may acquire this buffer to check which devices went wrong and what went wrong and get an encoded message for each device.

- The ucode allocates and keeps all accessed devices in device lists. If the same set of devices are accessed again, the ucode does not need to search the DDR database once more.

- The ucode uses a special algorithm to assign each process an access class. Every process may read devices. However, whether a process may set devices or not depends on its access class and the system class at that time. Every device may be locked. Once a device is locked, no one may write to it unless it is unlocked at first. All the write command will be logged into a disk file[1].

3.3 The UIF Toolkit

The User InterFace Toolkit (UIF) based on X and Motif has been developed and it has greatly sped up the development of graphcal software, which constitutes a major portion of the important programs in the system.

The toolkit provides a consitant look and feel to the operators by providing a standard Menubar with the standard buttons namely Help, Tools and Quit. Other buttons on the menubar are user configurable. Below the menubar is the drawing area where the user can draw various graphics objects, plot graphs *etc.* The toolkit provides higher level abstractions such as a Matrix and a Plot. With the Matrix, it is very easy to create tabular displays. It has some features of a spreadsheet.

The toolkit provides a rich set of User Interface tools such as popup and pulldown menus, various kinds of buttons, lists, directory and file browsers, scrollbars, controls specific switch panels, text input areas, standard error popups, confirmation boxes *etc.*

All the X fonts and colors are available to the user. The toolkit deals with the graphics at an individual object level and lets the user interact with it. The user can group a number of such objects and can save them in a file.

The timer events are integrated with the toolkit. One can create an animation effect using this feature.

4 Orbit Monitoring and Display System

In addition to a complete and big set of orbit measurement, comparison, display, correction and history data collection and display programs, the following new features have been added in the upgraded system.

[1] we are developing a command log server to make the command log process more efficient and centralized.

- Real-time orbit graphical display: this is a sophisticated program which has numerous options. However, its greatest feature lies in the real time display with 4 to 6 Hz display rate, which is limited by the drawing speed of the X graphics. One may see the orbit moving or changing almost spontaneously when doing orbit correction or sending local bumps. It is one of the most useful tools in the system used in the studies and daily operations.

- Fast orbit history: this is another very useful tool. The fast history daemons run in the background and read a user-specified set of monitors at the user-specified rate (between 5 to 20 Hz). Several hours (specified by users) of the most recent readings are stored in a circular ring buffer. The program will dump a portion of or the whole buffer to disk files upon request, which is made through a signal sent by another process. The dumped data are in the NSLS standard file format and a general plot program then is used to look at and browse the data set. Many interesting phenomena have been detected by using the fast orbit history.

5 Standard File System and General History

In order to make the important data files transparent through the whole control system and to develop a general history and display program, we developed the standard file system.

A file in the standard format has a file header and segment headers which contain all the necessary information for application programs to read and display the file without prior knowledge. A C function library and several programs to inspect and display the data in the files of the standard format have been developed. The library functions provide an easy way to create, write, update, edit, search and read standard files. For example, One may read/write any rows or columns in the file or read/write any columns by their physical types or labels.

The standard file system now is used in many programs. One of its major use is in the general history system. Anyone may start his own history to record a certain set of devices at certain rate and write data to a specific directory in the standard file format. A general history graphical display may read and display any history files. At first, the display program gets all the device names from the segment header and prompts user to select some devices to display. Then it reads the data by device name or by column index. There are many other options in the program.

The ring managers and many users have started their own history processes. The general history system is widely and fully used in the Department.

6 The Interpreter

An interpreter has been developed by using the UNIX yacc utility. Since it was developed, more than 200 macros have been written by physicists, ring managers, control personnels and beam line users. These macros are used in the every day operations such as ramping, feedback controls, local and angle bumps, orbit corrections and other kinds of monitoring and controlling functions.

The interpreter may be executed interactively or run prewritten macros. It has all the read/write/control functions used in the control system plus many mathematical and scientific functions. It offers an unlimited layers of control flows and loops (both C- and Fortran-style if, while, for and *etc*) and macros may be nested to 16 layers deep. It has its own set of input/output functions, and the UNIX cursor and window package is implemented into it. In addition to the normal data types and float and character arrays, it has special data types to allow reading and writing control system parameters easily. For example, there is an array type for device names. The quotation marks are not needed when assign a device name to a variable. The built-in macro utility lets users list, inspect, edit and execute macros very easily. One may execute any shell commands and existing programs from the interpreter.

7 summary

We have presented the major components and features of the high level software. The upgraded system has been successfully installed and used in operations and is promising. Physicists, engineers and operators appreciate it very much. There are many important components and programs such as graphcal save/restore, pretune, softramp, bramp, runramp, automenu *etc*, which are not mentioned in the text.

Now we are evaluating comercial databases. The DDR naming server is under development. The phase III upgrade will start soon. We hope a more capable and more intelligent control system will emerge on the completion of the whole upgrade process.

I. References

[1] J.D. Smith, *et al.* , this proceeding (1993).

[2] S. Ramamoothy, *et al.* , this proceeding (1993).

[3] J.D. Smith, *et al.* , System Upgrade, NSLS, BNL (1990).

[4] There are many technotes written by members of the Computer Group on separate pieces of software. They are too many to list.