CONTROL OF STOCHASTIC COOLING & RF SYSTEMS USING INTERACTIVE GRAPHICS

Glenn Mayer, Stephen Beck, Ralph Pasquinelli

Fermi National Accelerator Laboratory[1]
P.O. Box 500 Batavia, Illinois 60510

Summary

The anti-proton source at Fermilab uses an interactive color graphics system to control the stochastic cooling and RF systems. This general purpose graphic control system provides an intuitive, powerful, and flexible person machine interface to the accelerator systems. The graphics interpreter translates user generated files which specify the location and type of symbol, and displays a live schematic of an accelerator system on a color graphics monitor. The system provides both static symbols, which can be used to generate a schematic, and dynamic symbols which display real time data and provide control. Experience has shown that this method of control has many advantages over simple text displays and has the flexibility necessary to be a an important tool in accelerator control.

[1] Operated by University Research Association Inc. under Contract with the United States Department of Energy.

Introduction

In October of 1983, the stochastic cooling systems for the anti-proton source were being built and the control system interface was being designed. Since the stochastic cooling systems were clearly destined to be complex, the user interface was closely examined. The standard interface in use at that time was the parameter page. This page displayed up to 20 lines of data at one time. Each line consisted of an eight character device name, a forty character device description, and the value associated with that device. Device parameters could also be controlled from the parameter page. If more that 20 devices existed in a system, the system was displayed using multiple pages. The parameter page worked well when controlling a small number of devices or a large number of similar devices. For example, to control a large number of magnets that need to be all on or all off, you could quickly page through all of the magnets and insure that they were all in the appropriate state. The stochastic cooling systems would have many

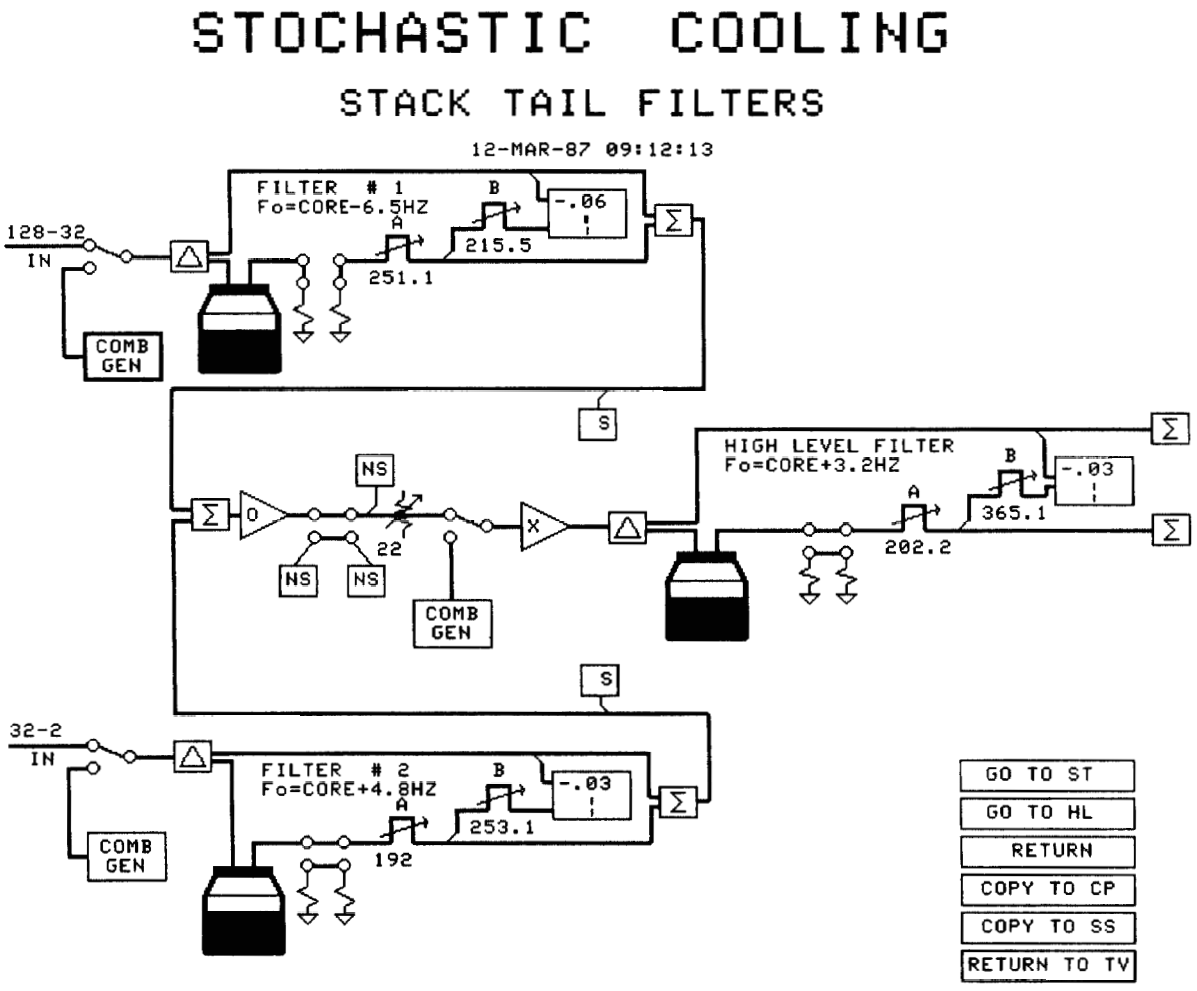

Figure 1 Typical Graphic Display

CH2387-9/87/0000-0738 $1.00 © IEEE

different types of devices, multi-pole switches, amplifiers, transfer switches, attenuators, trombone lines, phase shifters, etc. Not only were the interconnections between these devices going to be complex, but no two systems would be alike. It was feared that no one would be able to operate the system without a schematic in front of them. It was decided to implement some form of graphic interface similar to the Apple Lisa[2].

Basic Design Concepts

The primary difference between the interface that the Apple Macintosh[2] (the current Apple version of the Lisa) uses and a "typical" computer, is that the Macintosh interface is "intuitive". That is to say, the actions necessary to perform common operations are immediately obvious to a beginning user. The classic example of this is file deletion. If a beginning user is presented with a file which needs to be deleted and a picture of a trashcan, his first thought would be to move the file into the trashcan. The beauty of the Macintosh interface is that he would be right. There is no other logical alternative, therefore the user will select the appropriate course of action. Contrast this to a more conventional system which would require typing;

"DELETE,filename" or
"REMOVE filename" or
"PURGE, filename" or even
"DELETE, filename/OPTION=EXPUNGE"

In order to design an interface which is intuitive, one must first find a body of knowledge that all of the potential users of the system are likely to already possess. One then must design the system such that the assumed body of knowledge is sufficient for a new user to be able to operate the system without extensive training. For example, the Apple Lisa was targeted for the office environment. The potential users were familiar with the office environment, i.e. documents go into file folders, folders go into file cabinets, new things appear in the IN basket, garbage goes into the trashcan, etc. The Lisa interface used a "desktop metaphor" for all operations thereby allowing people to use existing knowledge instead of requiring them to learn new concepts.

Schematic Metaphor

The stochastic cooling systems are controlled by physicists, engineers, and operators, who are all familiar with schematic drawings. If the stochastic cooling interface looks like a schematic drawing then people will be able to recognize components without needing to know that "AL" in the 5th and 6th character positions of the name means that the device is an amplifier. They will know that all of the triangular shaped things are amplifiers from past knowledge and experience.

The final design for the stochastic cooling controls interface was a graphic image of the system that looked similar to the schematics from which the system was built (See Figure1). The appearance of a device reflects the devices current state in the real world, i.e. a switch which is schematically represented as a bar connecting two circles has the bar actually connecting the circles if the real switch is closed, or the bar tilted up if the real switch is open. Color is also used to indicate the state of a device. A device in its' operational mode is shown in green, standby mode is magenta, and a fault mode is shown in red. Analog devices have their current value displayed below the device. If the analog value if greater than a preset maximum, the value is displayed in red. If a user selects a device by pointing at it, additional information is provided about that device. An "interrupt" button is provided that allows the user to control devices, for example, if the user points at a switch and then presses the interrupt button, the switch will go from its current state to the opposite state. A knob is provided for changing analog values. If the user points to an analog device and turns the knob, the value will changed accordingly. Analog values may also be typed in.

[2] ™ Apple Computer Inc.

Hierarchical Organization

The total number of devices in most systems exceeds what can be reasonably put on the 14 in. color graphics screen. In order to display all data in an organized manner, a hierarchically organized set of graphic displays is created for each system. The first level displays the key components of the system. For any complex components, the user may interrupt inside the component and it will expand to a new, more detailed view of that one component. For example, the stack tail momentum system has a box labeled "FILTER", which is the cryogenic notch filter system. If the user interrupts inside a filter box, it will expand to show all of components of the filter, such as trombone lines, transfer switches, helium levels, etc. The graphics may be nested to any number of levels. The current stochastic cooling graphics have up to four levels of abstraction.

Implementation of the System

The interactive graphics system is implemented on the standard ACNET (Accelerator Control NETwork) console. This console provides the following resources;

Alphanumeric color TV screen
Color graphics screen (512x640, 8 colors)
Black and white graphics screen
Keyboard
Trackball with interrupt button
Knob

The schematic is displayed on the color graphics screen. The alphanumeric display is used to display error messages and additional information about a selected device. The trackball is used to move the cursor on the graphics display.

The graphics system is composed of three major components: the graphic definition file, the symbol definition routines, and the graphics interpreter (See Figure 2).

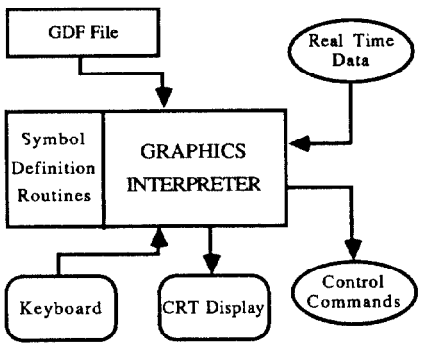

Figure 2 Graphics System Structure

Graphic Definition File

The graphic definition file (GDF) is an ASCII file that is created by the user for each graphic image. Each line in the file specifies one symbol on the screen, Each line has 8 integer parameters and one 8 character alphanumeric parameter. The first integer parameter is always a symbol number. All of the other parameters are interpreted according to the symbol definition for that symbol. A line beginning with a "!" is a comment and is ignored by the interpreter. The following is a short example of a GDF.

```
! A SAMPLE GDF FILE
!1112222333344445555666677778888AAAAAAAA
! THIS LINE PRINTS THE TEXT HELLO
! AT COORDINATES X=100,Y=200 IN GREEN(2)
! IN THE STANDARD SIZE (1)
   5 100 200   2   1   5          HELLO
!
! THIS PUTS THE VALUE OF A:IBEAM UNDER HELLO
  11 100 180                      A:IBEAM
```

A typical graphic definition file is about 300-500 lines long. The GDF file can be entered using a text editor, in fact the first stochastic cooling graphics were all entered using a text editor. For a far more detailed description of GDF files see reference [1]. Since direct entry of text is a time consuming process, an interactive graphic editor was developed. This editor runs on a ACNET console and allows the user to move, enter and delete symbols directly. See reference [2] for more information about the interactive editor.

Symbol Definition Routines

For every type of symbol that appears on the graphic, a symbol definition must exist. A symbol definition consists of three subroutines. The INI routine defines the necessary initialization for that symbol type. This usually consists of drawing the symbols initial shape on the graphics screen and requesting that the appropriate data be sent to the console's data pool. The PER routine defines the proper procedure for refreshing the symbol. This usually consists of getting data from the datapool and drawing the appropriate symbol over the last symbol. The KBI routine defines the action that occurs when the user presses the interrupt button while the symbol is selected. This action is differs widely for different symbols. For a switch symbol, the current state is obtained from the data pool and a command which will set it to the opposite state is sent out.

There are 20 words of global data area allocated to each symbol. This area is used differently by each symbol. The types of things stored are: symbol number, lower left and upper right hand corner of the valid interrupt area of the symbol, pointers to the live data in the datapool, pointers to the correct data scaling routines, etc. Each of the three symbol routines has access to this global data.

A set of 30 symbol definitions is linked to the graphics interpreter to form a version of the interactive graphics system. For example, the stochastic cooling version of the graphics system is composed of 15 general purpose symbol types, and 15 symbol types unique to stochastic cooling. Memory limitations of the console CPU prevent all of the symbols from being linked into one program. The symbol number (1-30) is defined at link time. For example, if the trombone line symbol is the 22nd symbol linked into the system, it becomes symbol 22. See reference [3] for details of symbol definition.

A general purpose library of symbols exists so that graphics systems can easily be built. The general purpose library consists of; lines, boxes, circles, triangles, text, cursor transfer to the TV screen, hardcopy of graphic image, global setting of devices, open a window to a new graphic, time/date, analog value, basic status, basic control, bar graph of analog value. Reference [1] has a complete description of the general purpose symbol library. A control graphic for many systems can be built from these basic symbols, however a few custom symbols usually make the final product much more esthetically pleasing.

Graphics Interpreter

The graphics interpreter's basic function is to schedule the execution of the symbol definition routines. The following is a brief sequence of events (See Figure 3).

1) Initialize all variables

2) Get a GDF file

3) Go through the GDF file a line at a time, for each symbol call the appropriate INI routine passing the parameters in the line to the routine.

4) Go into a loop, calling the PER routine for each symbol as often as possible. If a keyboard interrupt occurs, find the currently selected symbol and call the KBI routine for that symbol.

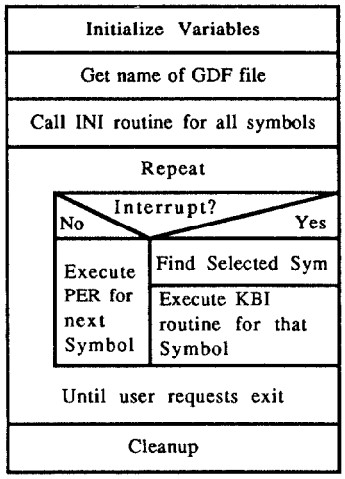

Figure 3 Logic for Graphics Interpreter

Operational Experience

The primary problem encountered with the interactive graphics system was the tendency to overload the data collection capacity of the control system. The first graphic page consumed 80% of the total data collection capacity of the p-bar source. The first graphic page (stochastic cooling stack tail momentum) had about 200 devices on it compared to a standard parameter page which displays at most 20 devices. There was no real solution to this problem other than to exercise restraint when designing graphics.

The decision to use three subroutines to define a symbol type has proved to be a good one. Many new symbols have been designed and implemented since the original system has been released. There has never been a problem or limitation which prevented a person from doing exactly what they wanted to do. One symbol designer created a water pump symbol that regularly moved the impeller position a fraction of a turn if the pump was on. The result was an animated symbol.

The final lesson learned was that in using a powerful and flexible facility such as this it is not only possible to create informative and easy to use graphics that are pleasing to the eye, one can create cluttered, unorganized graphics that are virtually impossible to follow. In fact, graphics of the second category are much easier to do than the first. The Apple Macintosh team included a full time artist on the staff. Any large scale system (SSC?) planning to use a graphical interface should certainly have high caliber artistic talent included in the design team.

Acknowledgments

The interactive graphics interface could not have reached its final form without the help of many people. Kevin Cahill and Jim Smedinghoff freely provided system support, vital to this project. Dave Johnson, Wally Kissel, Don Rohde, and Duane Voy provided feedback in the implementation of the current system and ideas for future development. John Marriner provided ideas and feedback during the design phase of the project.

References

[1] G. N. Mayer and S. Beck, "Basic Interactive Graphic Interface", Software Documentation Memo No. 78, FERMILAB, December 9, 1985.

[2] S. Beck and G. N. Mayer, "The Interactive Graphics Editor", Software Documentation Release No. 86, FERMILAB, December 8, 1986.

[3] G. N. Mayer and S. Beck, "Advanced Interactive Graphic Interface", Software Documentation Memo No. 82, FERMILAB, January 17, 1986.