SLAC-PUB-5511
May 1991
(A)

DESIGN OF VAX SOFTWARE FOR A GENERALIZED FEEDBACK SYSTEM*

F. Rouse,[a] S. Castillo,[b] T. Himel, B. Sass, and H. Shoaee
Stanford Linear Accelerator Center, Stanford University, Stanford, CA 94309 USA

Abstract

Fast feedback in the Stanford Linear Collider (SLC) not only works, but is necessary. We have several examples of currently running systems that have greatly improved the performance of the accelerator. In order to increase the number of feedback loops, it has become necessary to redesign the system to allow a database description of any feedback loop. We use digital control theory to formally describe each feedback loop in terms of a matrix equation. Then a new feedback loop requires only an update to the database, and perhaps the installation of a inter-micro communications link. This paper details the design of the VAX software required to implement the new system.

INTRODUCTION

The SLAC Linear Collider (SLC) is a novel accelerator designed to produce e^+e^- collisions at center-of-mass energies up to 100 GeV, i.e., around the mass of the neutral intermediate vector boson Z^0. The collisions occur between electrons and positrons produced on every crossing as opposed to being stored for an extended time, as in electron-positron storage rings. Currently, the SLC has feedback loops that stabilize the energy of the machine, the orbit through a set of collimators near the end of the linear accelerator, and one that maintains the beams in collision. These feedback loops are essential to the operation of the SLC. The software for these feedback loops resides on both a VAX 8800 and a series of INTEL 80386 microprocessors. The 80386 processors actually control the devices that accelerate and control the beam.

We have designed a new system that replaces the current software with generic, database-driven software. We rely on the SLC database to specify each different loop. This is possible because the action of any feedback loop can be cast into a series of matrix equations in the formalism of digital control theory [1].

The problem of closed-loop feedback can be described by a series of matrix equations. We therefore use the term vector to refer to the vectors of measurements, state variables, and control elements used in the feedback loop. We use the term matrix to refer to the matrices that connect the vectors together into an equation [1].

The SLC database specifies the matrices and describes the vectors the matrices act on. The database also contains the complete description of what needs to be measured and how to affect the actuators (usually magnets) to carry out the changes required to stabilize the loop. We design the matrices and specify the loop in the database, add the hardware for the network linking the different micros in the loop, and reboot the micros to start up a new feedback loop in this new system.

* Work supported by Department of Energy contract DE-AC03-76SF00515.

(a) Present address: University of California at Davis.
(b) Present address: Apple Computer Corporation.

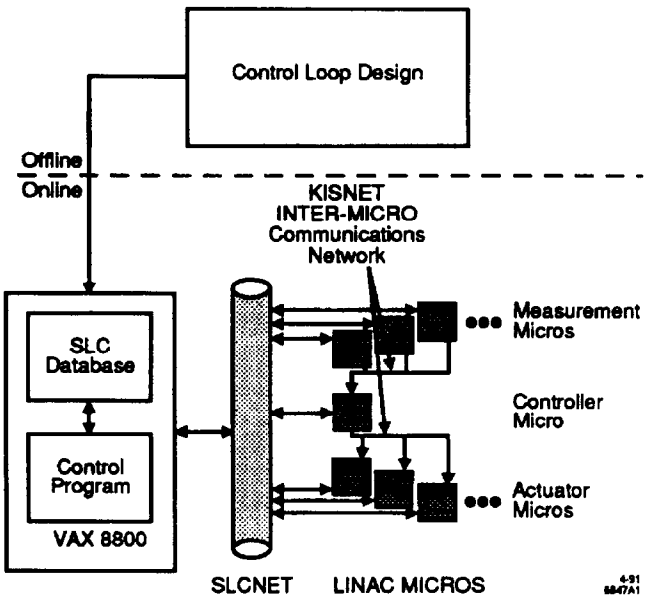

Figure 1. Overview of the components for one feedback loop.

Figure 1 shows the basic components needed for one loop. Matrix design is done offline [1]. The VAX orchestrates how each and every feedback loop works. The INTEL 80386 microprocessors actually do the work of the feedback loop: measure, compute the corrections for and control the hardware devices for the beam. The microprocessors communicate among themselves via a new network called KISNet [2].

Primarily, the VAX orchestrates the new feedback by providing the initialization of all microfeedback jobs, the user interface for analysis and operator diagnostics, and the management of the feedback database.

ELEMENTS OF THE VAX SYSTEM

The new feedback system is first and foremost a database driven system. We built a linked-list system that classifies relevant information together. The objects chosen are shown in Figs. 2 and 3. The feedback structure is the highest ranking linked list. Each linked list consists of pointers to the lower ranking lists, along with the name (used as a key) and other pertinent information. The microstructure mainly points to descriptors associated with each micro attached to the loop. There are also plotting and analysis structures used by the user interface.

The model hierarchy shown in Fig. 3 stores the information associated with the vectors and the matrices. Ring buffers store pulse-by-pulse data from the microprocessors. Data mainly consists of vectors and the status of each vector element. We store information describing each element in the vector and the matrices used by the feedback loop.

We separated the design of the hierarchy of the feedback loop and the interaction with the SLC database from application code. This allows changes in the precise representation of the feedback hierarchy or changes

Presented at the IEEE Particle Accelerator Conference, San Francisco, CA, May 6-9, 1991

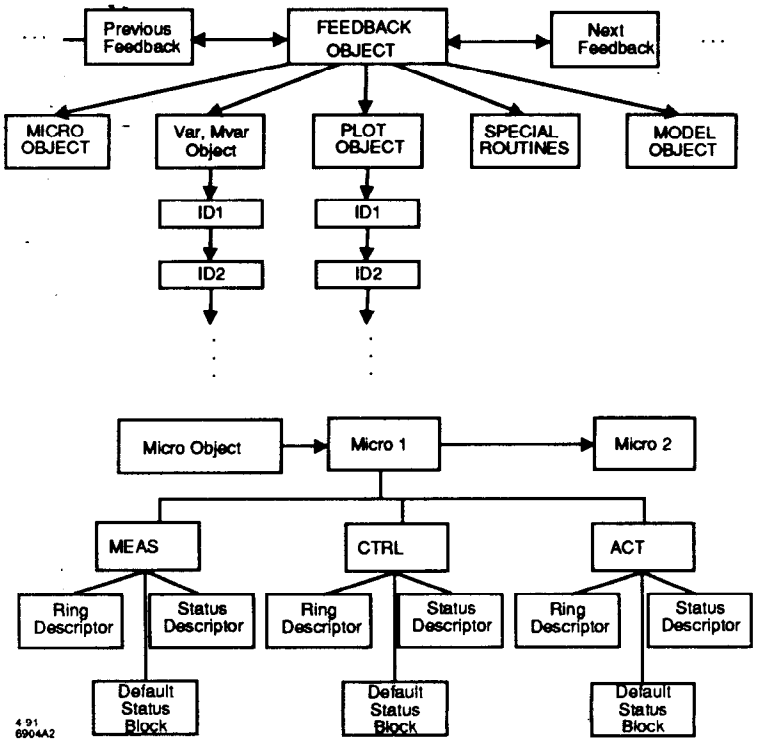

Figure 2. Diagram of feedback hierarchy.

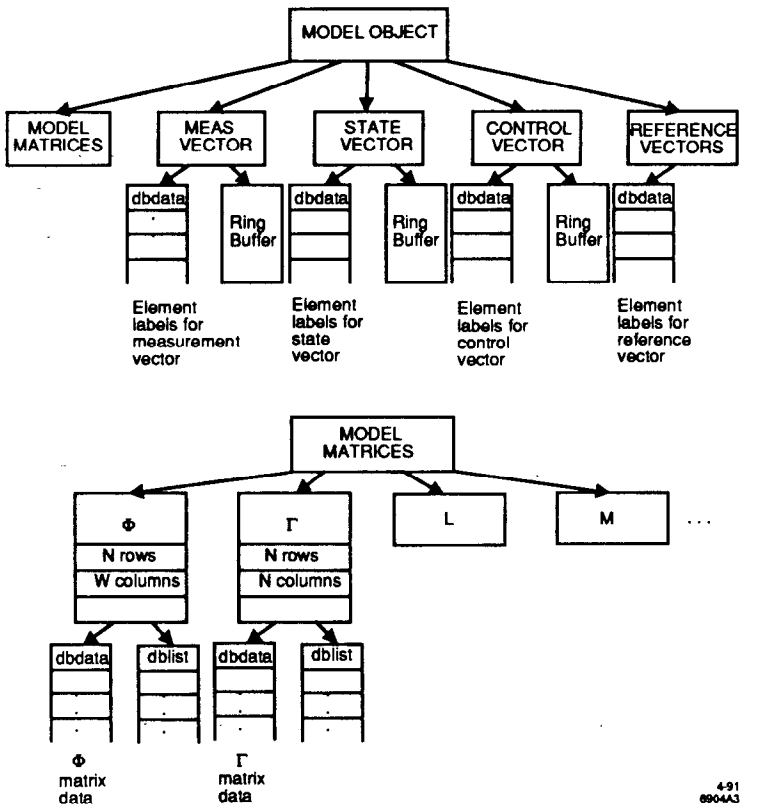

Figure 3. Diagram of model hierarchy.

in the SLC database without having to change the application code. The feedback code is layered as shown in Fig. 4. The outermost layer is the ***SCP*** layer. This layer manages all user interfaces. We request actions via touch panels or terminals, the SLC's standard interface devices. Actions include loop control and calibration, diagnostic interventions, display of recent feedback data (measurements, states or actuator settings) as a function of time, and listing of pertinent loop information. Displays are shown on a color graphics monitor.

The next layer is called the ***application utility*** layer. This layer buffers calls from the application code layer to the ***kernel*** layer. The most important layer is the ***kernel***. Here the feedback hierarchy is stored. The particular representation of the hierarchy is a series of linked lists.

The user interface in the new system allows for arbitrary numbers of feedback loops, micros, and plots. Additionally, the user interface handles all common interface functions, such as changing the state of a feedback loop; turning on or off a particular device; listing the status of a feedback loop; analysis and histograms; and listing the feedback loops, micros, plots, or ring buffer elements.

SLC DATABASE

The information in the database for the feedback system consists of two classes: feedback loop information and display information. Feedback loop information includes a loop name, descriptions of the micros carrying out the measurement, controller and actuator tasks of the loop and the communication links between them, the feedback matrices, and the vectors the matrices act upon. We also specify the state vector that the controller uses to compute the actuator settings. The display information consists of the plot names, windowing for specified plots, and variables. We key off of the feedback loop name for all information pertaining to the loop.

The matrices are generated offline by modeling the action of the feedback loop along with the model of the accelerator. The matrices are then loaded into the SLC database by the offline program. They are stored in a sparse format.

The vectors must have specific device information, since the measurement and actuation drivers need CAMAC control words and locations in order to read out or set their respective devices. Typically, feedback routines only need a pointer to specific device information. The device information is already in the database to allow control of the accelerator by preexisting applications. Each vector element has a corresponding label that includes the keywords required for unique database access. Finally, the database also describes physical and display units, tolerances, axis labels, etc., for each vector element

OBJECTS

The basis of the categorization system is the linked list. Each linked list contains different information, but movement from node to node along

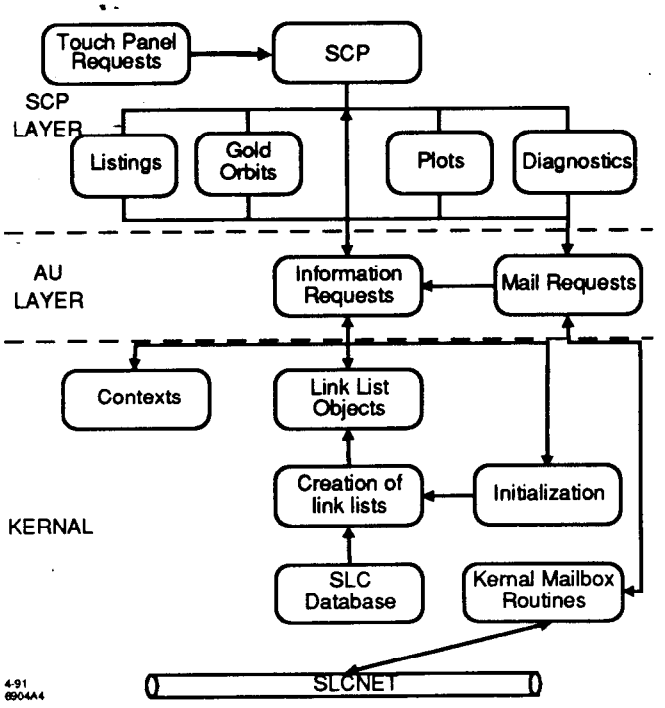

Figure 4. Layering of the VAX feedback code

the link list is a common action. Additionally, the major user interface is through programmable CRT touch panels. Often we do not know how many buttons will be required on the panel. The action initiated by pushing any one button may, of course, be different. The action of pushing a feedback loop button is different than pushing a button that describes the element in a vector. Yet the act of selection or deselection of a button, or the printing of a screen can be described by common code.

The use of ***"objects"*** [3] can elegantly represent our problem. We allow our objects to inherit instance variables and internal methods to give a slightly different behaviour to our various linked lists and button actions [3].

Let us consider our linked list class in particular. We have linked lists of feedback loops, micros, plots and plotting variables within a feedback loop (see Fig. 2). This linked-list class is an example of a container class [3]. The class handles movement from node to node along the list, listing each node in the class, adding nodes to and removing nodes from the list, and creation and deletion of the entire linked list. Each subclass implements its own particular behavior.

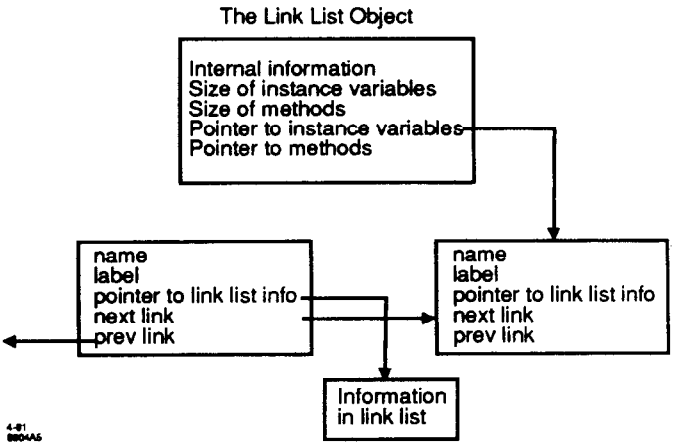

Figure 5. The structure of an object. This example shows the linked list object. The instance variables are also shown. The methods are the common actions for all linked lists.

The actual structure of the linked list object that we implemented is shown in Fig. 5. The description of the object is only required in a fully objective system, though we use the SIZEOF fields to create copies (new instances) of objects. The important part of the object are the pointers to the instance variables and to the methods.

CONCLUSIONS

We have described the VAX code used by a generalized feedback system at the Stanford Linear Collider. The system categorizes various classes of information by the use of linked lists. The lists use a primitive form of an objective-C object in which a container class describes all linked lists. This allows us to have an arbitrary number of feedback loops whose behaviour can be globally modified by changing common code.

ACKNOWLEDGMENTS

We thank John Zicker for his early work on this problem. We also thank Lee Patmore, Phyllis Grossberg and Bob Hall for their efforts on the VAX code.

REFERENCES

[1] T. Himel et al., "Use of Digital Control Theory State Space Formalism for Feedback at the SLC," Proc. 1991 IEEE Particle Accelerator Conf., San Francisco, CA, 1991.

[2] K. Krauter and D. Nelson, "SLC's Adaptation of the ALS High Performance Serial Link," ibid.

[3] B. Cox, *"Object Oriented Programming, The Evolutionary Approach,"* (Addison-Wesley, 1986.)