Summary

SLAC is consolidating its two control rooms by utilizing existing control computers in each, connected by communications link. Except for video signals, all of the controls and data of one (Central Control Room) will be sent to the operators in the other (beam switchyard control room) using this system. The paper is concerned with the software aspects of this project. The major program components are described.

Introduction

By late summer of this year, SLAC plans to have consolidated its two existing control rooms. The SDS-925 computer, located in what will become the Main Control Center, is mainly responsible for the operator/computer interaction and for bringing the control over a data link, drives most of the accelerator interfaces from the old Central Control Room. A paper in these proceedings discusses the main hardware component and another goes into detail on the touch panel displays, which the operator uses to control the accelerator. This paper discusses the software aspects of the system. A software block diagram, Fig. 1, shows the basic building blocks. Control panel designs, made up by operators or other nonprogrammers, are described in a source language. This is converted by a 360/91 program into tables which are loaded into the 925’s "touch panel library." The basic SDS-925 system program has been augmented to include conversational input, the display and touch panel interface, as well as the link handler. The data link itself is a lower priority task, creating a new layer of dynamic variables that will be used when an accelerator status change is received. Additional programs, such as analog and text handlers, which are erased when the task ends. The disadvantage of such systems is that it is difficult to vary the priority of the task as it goes to completion because its dynamic variables are at a fixed layer in the push down stack. This is a serious drawback in our application because most tasks involve real time delays (to wait for relays to close, or for data on the disk, to name two examples).

The above considerations led to the development of DS, a multitasking operating system on the PDP-9. Its resident section includes a disk relocatable loader, interrupt driven programs, and a task scheduler. The rest of memory is occupied by areas for user program blocks and dynamic memory of the tasks. A task includes such information as execution time, current program location, and user variables, but does not include the user program itself. In fact, a task may go from program to program as it progresses. While a user task is in execution the processor is in user mode; otherwise it is in a system break, controlled, and deleted by a set of macros provided in DS.

The events may be defined, as above, or user specified. By coding the macros the programmer has complete control over occurrence of system breaks within his task. He must protect his variables against total system breaks (by putting them into dynamic storage because the program may be re-entered by other tasks before a given task resumes. Within any one user period, variable protection is unnecessary.

Operating Systems

A partial panel inside the 925 is a data table containing display control specifications for a rectangular array of accelerator buttons. The tables are made up of five basic subsections: fixed text, analog signal list, accelerator status signal list, button specifications, and computer generated text. One or more of these tables, linked together, define a panel. The linkage name each constituent partial panel along with its geographic position on the panel. To utilize the tables, there are various processing programs. A panel select program obtains a panel from the library and puts it on display (i.e. – onto the currently active list). A button processing program monitors and decodes push button interrupts, using graph data to initiate the appropriate response functions, if any. Another program uses a similar procedure when an accelerator status change occurs. Tasks are initiated, controlled, and deleted by a set of macros provided in DS. The events may be defined, as above, or user specified. By coding the macros the programmer has complete control over occurrence of system breaks within his task. He must protect his variables against total system breaks (by putting them into dynamic storage because the program may be re-entered by other tasks before a given task resumes. Within any one user period, variable protection is unnecessary.

Off-Line Panel Compiler

Because the touch panel system will be comprised of hundreds of different panel displays, and therefore, hundreds of different data tables to make up the displays, it is necessary to have a means to quickly and easily produce the data tables. A language was devised whereby a desired panel display could be described in a way that is easy for a nonprogrammer to write, but that contains all the necessary information. This special language is used as input to a panel generating program that operates on the IBM 360-91. The output from this

*Work supported by the U. S. Atomic Energy Commission.
program is a punched deck of binary cards to read into the
ouch panel library on the SDS-925 drum.

A push button panel may cover up to three TV monitors
which make up a single operator console. For example, a
panel (or partial panel) may be a row of buttons or 20 lines
of binary status.

A panel may consist of many partial panels, and a given
partial panel may be a part of different panels. The panel
generating program has two functions: to generate tables for
partial panels and to create linkages which define panels. In
the partial panel source language, there are five basic in-
struction types, corresponding to the subsections given above.

If a user wishes buttons to appear on his partial panel,
he includes a "button" type instruction in his input deck for
each button. He must include the following information: lo-
cation of the button relative to the origin of the partial panel,
function of the button when pushed, function when released
(if any), type and shape of the button. If the user wants text
on his panel, like the label of the button, he inserts a "Text"
instruction in his deck for each line of text. Many items
have defaults, like letter size will be normal height and nor-
mal width unless otherwise requested, but certain informa-
tion must be supplied, like the element for which to display
the binary status, the units and magnitude of the analogs, the
field width of computer generated messages, and positions.
The program also converts all x, y coordinate references of
the source language into the 256 x 256 coordinate system of
the TV display system.

Control Functions and Data Logging

Each control room had its computer before the concept
of consolidation was considered. The BSY computer was in-
stalled during the construction of the control room. The
CCR computer is relatively new, first going on-line less than
two years ago. Initial on-line usage was concerned largely
with data logging, used both as a maintenance aid (as, for
example, continuous recording of the faulting of each of the
245 klystrons) and for the operations log (e.g., total on time
and number of pulses delivered to each experimenter). This
was soon followed by control functions, such as switching on
klystrons to replace those that had faulted, and setting the
(~ 50) accelerator quadrupoles to pre-assigned values. The
computer is also used to test the complicated interlock net-
works used for the protection of personnel and the machine.
Consolidation will have little effect on this software, except
that most of the operator interaction with the computer will
be via the touch panel/link system rather than the teletype as
at present. Little closed-loop control of the accelerator has
been accomplished, largely due to the lack of the necessary
interfaces. Consolidation has accelerated the schedule of
building these interfaces. However, plans for closed loop
control have been deferred until the more basic direct con-
A

References
1. K. Breymer et al. , "SLAC control room consolidation
using linked computers," this proceedings.
2. D. Fryberger and R. Johnson, "An innovation in control
panels for large computer control systems," this
proceedings.
3. S. Howry et al., "The SLAC beam switchyard computer,"
IEEE Transactions on Nuclear Science (June 1967).

FIG. 1--SLAC control consolidation -- software diagram.