SLAC-PUB-2681
February 1981
(A)

PRESENT SLAC ACCELERATOR COMPUTER CONTROL SYSTEM FEATURES*

V. Davidson and R. Johnson
Stanford Linear Accelerator Center
Stanford University, Stanford, California 94305

Abstract

The current functional organization and state of software development of the computer control system of the Stanford Linear Accelerator is described. Included is a discussion of the distribution of functions throughout the system, the local controller features, and currently implemented features of the touch panel portion of the system. The functional use of our "triplex" of PDP11-34 computers sharing common memory is described. Also included is a description of the use of "pseudopanel" tables as data tables for closed loop control functions.

System Configuration

The present accelerator control system has evolved slowly for more than ten years. Over the past two years we have come to have a system which is basically a star network. At the center are three PDP11-34's sharing common memory. Fifteen arms radiate outward connecting to various mini/micro computers. A few local processors are connected using parallel links, but most links are 9600 baud asynchronous. Eight arms consist of a PDP8 heading a secondary partyline of twelve microprocessors. The PDP8's and microprocessors are all local controllers containing monitoring and control loops as described below. All together the system includes three PDP11-34's, ten PDP8's, 31 Intel 8080 systems, 66 Motorola 6800 systems, and one PDP9. This summer we plan to install Intel 8086 multibus systems to replace the PDP9 and one of the 6800 systems.

PDP11 Triplex

The PDP11 triplex, under the RSX-11M† operating system (V3.2), provides touch panel software as the operator interface, specialized functions such as magnet conditioning, link interfacing, etc. Common memory contains status arrays, digitized analog arrays, buffering for displays and inter-cpu communications, and touch panel tables.

In general, one PDP11 provides the network interfacing, task initiation via touch panel buttons, panel and global data selection, and effects controls which originate as touch panel pushes. Another PDP11 provides the display system interfacing; manages all normal touch panel displays of analogs, status, and messages; and provides a few special displays. The third PDP11 currently provides "higher order" functions such as magnet conditioning, logging of data, special analysis, etc. It also serves as the program development computer and as a spare for the other PDP11's. Each PDP11 has 80k of local memory and 48 k of common memory. Each has an RK05 removable disk drive. The program development computer additionally has a second RK05 drive, an RL01 drive, a tape drive, and a Versatec line printer.

The operating system is single user and the executive is unmodified. However, we have modified the SAVE task to prevent memory clearing and the INSTALL task to provide a NOLOAD feature which inhibits actually loading a common partition image into memory. Both features were required for booting the systems in a common memory environment. Also, we use a driver subroutine to send unsolicited data to a task and/or request or resume a task. Until recently, we have not had hardware interrupt capabilities between cpu's, so each has a system task to monitor various common memory buffers for activity. Task communication between cpu's and display data are handled by drivers which interface to common memory arrays rather than hardware interfaces. File transfers are done via a deposit and retrieve one-block-at-a-time scheme. Handshaking is generally done via common memory flags.

Touch Panel Software

Touch panel software consists of a collection of programs which run on two of the PDP11's to provide the operator interface. It provides the basic functions of panel selection; global selection; task initiation; push button controls; and displays of analog, status and messages. It uses panel and global tables (one for each of 16 terminals) and analog and status arrays in common memory. Touch panel tables actually consist of two separate binary tables. One contains the static text, graphics, and button coordinates and is used only by the panel select program to initialize the display of a newly selected panel on a terminal. The other is the table which is kept in common memory (one for each terminal) for reference by all online functions.

In addition to these two tables, the panel compiler permits one to design a pseudopanel to describe a set of elements to be used by the closed loop set programs. This mechanism uses the compiler to generate control blocks for elements so that control subroutines used for touch panels can also be used by other programs to retrieve data from common arrays and to effect controls. These control blocks form part of an online control data base which is separate from the touch panel tables.

The source data bases containing parameters for all elements in the system, all touch panel sources, and the panel compiler (a PL1 program) are located on the SLAC computer center system. We compile panel tables and other secondary data bases and download binary tables to the PDP11's for online use.

Global (Terminal) Data

In addition to each of the 16 separate panel tables in common memory, there is also a separate table of global data for each terminal. This is data that does not change as different panels are selected on a given terminal. Presently we have the following global selects available: elements for control and analog display; accelerator sector and beamline for indexing control, analog, and status displays by sector and beamline; rate selects for dynamically modifying slew rates, DAC increments; setpoint and slew rates for DAC controls; and field values and time increments for bit controls. Planned, but not implemented, is the ability to select whether the digital value displayed for an element is its current value or another of its parameters such as its upper/lower DAC limits, tolerance maximum/minimum, slew rate, etc.

Local Controllers

As presently implemented all local functions are driven by local tables whose parameters are set by the PDP11's. For program diagnostics or special table changes outside normal control functions, local controller memory can remotely be read or written from the PDP11's by a read/write function. Also, there are

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

† Digital Equipment Corporation trademark

(Contributed to the 1981 Particle Accelerator Conference, Washington, D.C., March 11-13, 1981.)

typically four other basic functions: DAC control with slewing, bit control with variable field width and on time, status or binary state monitoring, and analog digitizing and monitoring. There is no operating system; all basic functions are table driven foreground or background loops.

Some table parameters are predefined at program assembly or load time, while others are dynamic, originating from touch panel data or special online programs. Generally, local control is effected by a loop response to table entries rather than to the direct reception of a network control packet. The tables provide the interface between the net and the local control. Likewise, monitor loops flag changes in tables which in turn are monitored by a network monitor loop which sends off changes as rapidly as possible.

DAC Control

The DAC control mechanism can be used for controlling any digital value whether DAC or memory location. The local controller has a slewing loop which has a repeat rate of ten times per second. For each DAC, or digital value, to be controlled there is a table entry for each of the following items: the current value, the requested value, the upper and lower DAC limits, the integer and fractional portions of the slew rate, and a counter for fractional slew rates. Should the limits be invariant and the same for all elements in a given controller, single constants rather than tables are used to save memory.

The local slew loop monitors the slew rate (integer plus fraction) and if nonzero compares the requested value with the current value. If different, the slew value is added to (subtracted from) the current value at each loop execution until the current value reaches the requested value. The slew rate table entries for that element are then set to zero and no further change takes place.

Asynchronously, control packets may arrive. These carry data which the receiving routine sets in the tables. Currently implemented are three packet types: stop, increase/decrease by delta (the packet provides a signed delta to be added to the current value to yield a new requested value), and setpoint (which is the new requested value). Newly calculated requested values are subject to the DAC limits. Although we have not as yet implemented it, DAC limits could be determined by the central computers and remotely set as required. To provide "instantaneous" changes, the slew rate is set to the maximum DAC range and to provide a stop it is set to zero.

In addition to being explicitly or globally specified, slew rates and delta values can be further modified dynamically by specifying global rates. MEDIUM leaves specified rates unchanged, SLOW forces a one DAC-unit/second rate, and FAST forces four times the given rate. Additionally, all rates can be single step (one change per button push) or continuous (while one has his finger on the button).

As yet, we have not implemented physical control knobs apart from the touch panel control surfaces. Should we wish to, we could readily implement them using touch panel coordinates which are off the screen. We have 16x by 16y control bits but use only 10x by 13y. Knob activity could be interpreted as a button push of an off-the-screen button, with slew rate or increments determined from the knob rotation by whatever algorithm is desired.

Bit Control

The Bit Control Mechanism is used to control the binary state of bits in a field from one to eight bits wide. Bits can be pulsed on for a given time (dt) or set to a given state. This mechanism is used for motor, program, on/off, open/close, etc. control. As with the DAC control there is a local control loop (1/60 second period) which reacts to table entries for each element or bit field. These table entries are: the current field data, field mask, on time count (dt), and mapping data as required to map field data to memory or I/O address. The local timing loop monitors the dt table and if an entry is positive, decrements the count and reacts as follows: a 1-to-0 transition clears the field in the mapped address and sets dt = -1. Other transitions set the field in the mapped address. Effectively, an initial dt greater than one produces a pulsed output, an initial dt equal to one will clear the field, and an initial dt equal to zero will set the field producing a dc output. Control packets may arrive asynchronously. They carry the time (dt), the field mask, and the new field data. Data arriving is inserted into the appropriate tables by a packet receiver routine.

Continuous actions can be programmed in two ways. First the push can send a packet to set a field to one state while the release could send a second state. Or a push could send a given state continuously while your finger is on the button producing a local retriggerable one-shot effect.

Status Monitoring

The local monitor loop period is about 1/10 second. Current states are compared to the state of the last scan on a per byte basis and are subject to a filter mask. Changes are saved in a fourth table. Although we have not as yet implemented it, the filter mask could be dynamically changed via a control packet from the central computers. A network monitoring loop continually cycles through the changed list, transmitting the current state and changed bits of any bytes flagged as changed. In the central computers, status bytes are updated in common memory and individual bit changes processed for status and message displays. Individual status bits (elements) can be displayed on touch panels as 1/0, boxes, or bars. A few are routed to special display routines.

The message display program manages a scrolling text display of status messages together with their time of occurrence and general geographical location. Data base parameters determine how the display is to react to a state change of a given element. There is a parent-child masking scheme whereby secondary fault messages occurring with a primary fault are not displayed. Elements can be entirely masked, state changes can be appended to the bottom of the display, and if desired, any previously shown fault messages erased. Message displays can be put on any panel with the number and length of the text lines and characters sizes specified.

Analog Monitoring

In general, digitized analog data is acquired and placed into a current value table in memory by a routine which manages an A-D converter. A local loop (period 1/5 second) monitors three states of an analog's value: hardware errors (e.g., overrange), tolerance limits, and changes of value for the purposes of sending unsolicited current data to keep the central computer data up-to-date. There are six tables with an entry for each element monitored. These are: the current value, the previous scan value (for comparison to the current), a delta for determining if the value has changed enough to send to the central computers, maximum and minimum values for determining out-of-tolerance conditions, and a status word for flagging changes and saving current error states. A changed flag bit can be set via a control packet from the central computers, providing a solicited analog read capability. The local controller network monitoring routine scans the changed table and sends to the

central computers the current digitized value and the status of any analog flagged as changed.

Upon receipt of the AM (analog monitoring) packet, the PDP11 interrupt routine saves the new value in common memory. Asynchronously, an analog display program regularly monitors all currently selected touch panel tables and updates the panel displays from data in common memory. Values associated with an increase/decrease type of button are updated about five times per second while the button is being pushed.

Presently, analog values can be displayed digitally as an n.m. format (n = total digits, m = number of decimal digits) using a $y = a + bx$ algorithm for units conversion, an eight digit decimal equivalent of the raw binary, a verticle or horizontal bar, or a vertically moving asterisk. Polynomial expansions and displays are done by special programs outside the touch panel routines. For the digital displays character size can be specified. For the bar displays, bar scaling can be specified.

Summary

With our present distribution of functions, the local controllers can have high duty cycle control and monitor loops to provide rapid responses for large numbers of elements. Tables used to drive the local loops not only provide a simple interface to the network aspects of control but are an effective clutch preventing overloading, pile up delays, and overrun effects. Display routines, in a separate cpu from the control routines, do not affect control response, even if the display overhead becomes quite large at times. Also, high bursts of control activity are readily absorbed. The deposition of analog values directly into common memory by the link drivers and the unsolicited sending of other packets received by drivers directly to tasks reduces context switching permitting the system to run smoothly.

Reference

K. Crook and R. Johnson, "A Touch Panel System for Control Applications," Digital Computer Applications to Process Control, 5th IFAC/IFIP Conference, The Hague, June 14-17, 1977.