SLAC-PUB-4885
March 1989
(A)

STANFORD LINEAR COLLIDER HISTORY DATA FACILITY*

RALPH JOHNSON

Stanford Linear Accelerator Center
Stanford University, Stanford, California 94309

ABSTRACT

To effectively maintain the operation of the Stanford Linear Collider the variation over time of control system device parameters and feedback loops must be known. Device status must reflect unwanted variations as well as current states. This paper discusses the software facility which provides the collection and display of history data, and the determination of a behavioral status. The facility's operation, internal design, and user interface are described.

1. INTRODUCTION

The Stanford Linear Collider History Data Facility is used to save the history of control system parameter values and to monitor the "behavior" of those parameters. Any device which is defined in the control system database may have the history of its parameters or state and severity saved. Data are saved in ring buffers so that the latest data replaces the oldest. The facility consists of several processes and a number of related files are shown in Fig. 1. The definitions of devices to be saved, and the descriptions of how to save them and how to use them for error determinations, are contained in text source files. These files are separately compiled to produce data files which are then installed in a continuously-executing history process for data collection. Data saved in files can be retrieved and used by applications programs. A number of plots are in use on the operations displays.

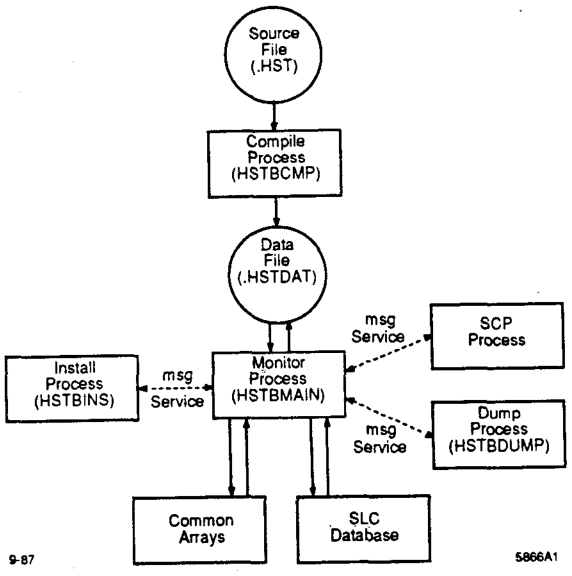

Fig. 1. History data system.

2. SOURCE FILE

The format of the lines in the text file is generally that of the DEC (Digital Equipment Corporation) CLI (Command Language Interpreter) format. Device definitions and save specifications are grouped into sections which produce corresponding sections in the data file. Different sections may save values at different intervals and have different numbers of datapoints. Likewise, different sections may specify different types of behavior checking. The general order within each section is to include a number of text lines to define a set of device parameters, followed by a specification of how often to save their values and how many values to save. For example:

```
DEFHSTB /DEV=(QUAD, LI02, 131, BACT)
DEFHSTB /DEV=(BEND, LI12, 552, BACT)
HSTBMON /INTERVAL=300 /POINTS=144
DEFHSTB /DEV=(QUAD, LI25, 900, BACT)
DEFHSTB /DEV=(QUAD, LI25, 901, BACT)
HSTBMON /INTERVAL=900 /POINTS=100.
```

The source file information is translated and written to a corresponding binary data file by a separate compiler process.

3. DATA FILE

A general diagram of the data file is shown in Fig. 2. This is a direct access unformatted file composed of a header, a queue of unit parameters to save, a database namedef area, a database index area, a database data area, a timestamp area, and the saved value area.

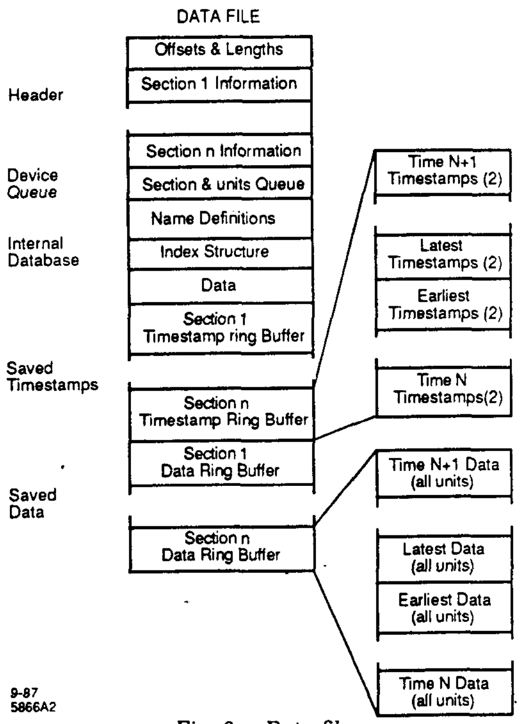

Fig. 2. Data file.

The file header contains pointers to the other areas of the file and contains section data which includes the number of points to save and the save intervals.

The device related queue structure contained in the data file is shown in Fig. 3. The queues are used to sequentially save device parameter values. There is a primary queue of section nodes. Each section node contains the next time to save data

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

for the parameters in its device queue and information needed to control the saving and processing of the data. These nodes also contain a pointer to a secondary queue of device list blocks which contain the specifications of all device parameters to be saved in that section. The unit list blocks contain the device name, subdevice name, area or micro where the device is located, a list of device units, the parameter name, and the name of the class of the device.

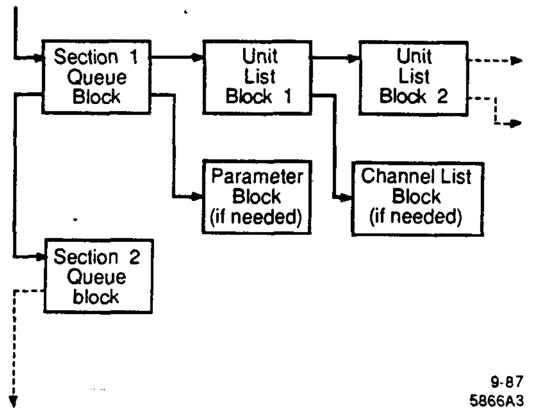

Fig. 3. Section and unit list queue.

The data area of the file contains a separate ring buffer of data points for each data section of the file. All data for all device units for a given point in time is contiguous in the order of the units queue specification. Thus each "data point" is a block whose length is the length of the format of the data times the number of parameter values in the section. It is these blocks which are the elements which are circularly saved, the oldest block overwritten by the newest.

The timestamp area of the file contains a separate ring buffer of pairs of timestamps. There is one pair of timestamps for each of the "data points" or value blocks in a file section. As with the data, these timestamp pairs are the elements which are circularly saved. The first timestamp of each pair is the time that the value of the first device parameter in the queue of parameters was saved; the second is the time the last device parameter value was saved. Therefor, a timespread is associated with each "data point". An application can use the first timestamp or the mean time as the time of a value, and can show the error in the time, if desired.

A database is contained within the data file for the purpose of direct access to data relating to a single given device parameter which is being saved. This provides a more rapid means of retieving the data than sequencially searching through the queue of devices and units. The database is composed of an index structure of three levels, as shown in Fig. 4. Each level is an array of structures. To retrieve values the device name, etc., are converted to numeric equivalents. Then the first level is searched to find the device/subdevice-area combination. The number of units for the combination and their location within the second level are used to search within the second level for the unit. The number of parameters for the unit and their location within the third level are used to search within the third level for the parameter. The number of words of data and their location within the value array are used to retrieve or save data which specifies the section number containing the device parameter, its location in the section ring buffer, and the current number of data points. This information provides direct access to the list of values (and associated timestamps) for a given device parameter specification.

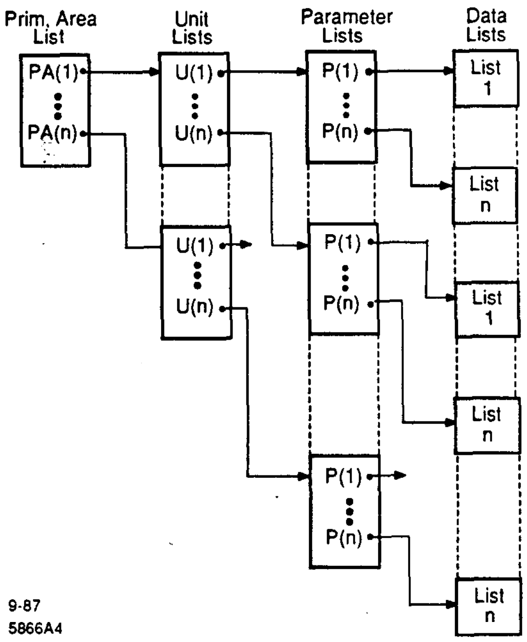

Fig. 4. Internal database Index structure.

4. DATA COLLECTION AND SAVING

The monitor process waits for either a message or the expiration of a timer. If there is a message, it will call the appropriate routine to perform data retrievals or file installations, resets, and removals. If the timer has expired, the update time will be checked for all sections of all files. After processing all files the timer is requeued using the minium of new section update times. For each section, the "data point" beginning timestamp is saved in the timestamp ring buffer, the data processing for all units in the section is performed, and then the ending timestamp for the "data point" is saved in the timestamp ring buffer. After data for each file section is saved, the main process loop is called to see if there are any waiting messages; this is to improve the response to user requests should there be a large number of units to be processed.

The processing of the unit list blocks consists of looping over all channels of all units in each list. The values are obtained from the control system database and saved in the "data point" value block in the data ring buffer. Point counts are updated in the internal database, and any behavioral checking is done.

During the initial pass of processing a newly-installed data file, the routines which get values from the internal or control system database use the supplied device, area, parameter, and channel names to obtain and locally save pointers to the needed data. The translation of names to low level pointers is very time consuming. Since subsequent file processing is done in the same sequential order, the saved pointers can be used, eliminating much unneeded overhead.

5. STANDARD DEVIATION CALCULATION AND USE

For most of the parameters saved, we maintain the standard deviation from the mean value for the sets of saved values. After saving each new value, a new deviation value is calculated. If the deviation is greater than the parameter's deviation tolerance, a flag bit is set for the device in the control system database. Other processes can use the flag to warn operators, and the state of the device's "behavior" can be included in device displays.

The algorithm used to calculate the deviation is one where the deviation is exponentially damped with a 1/e period of a set number of points. It weights the most recent data and provides a deviation which is meaningful for determining device "behavior". Operational diagnostics require that the deviation is responsive to recent values. However, device maintenance and design is better served by having longer term histories. By using a heavily weighted algorithm, longer histories can be kept but only the effects of the recent data points will effect the current deviation value. This algorithm is used rather than calculating the deviation from the actual set of values because older values are not readily available when a new value is saved. To use the list of current values would require too much cpu time since the system is designed to minimize the cpu time needed to save values rather than to retrieve them.

6. DATA RETRIEVAL

A function is available for inclusion in a process to retrieve a set of saved parameter values for a given device. The function requires device, subdevice, area or micro, parameter, and channel names; the unit number; and the history file name . It must also be given the address of a standard data retrieval structure where the data are to be deposited. This function sends a message containing the above information to the history process. The history process uses the data file's internal database to retrieve the requested parameter values and associated timestamps from the file and return them in a reply message.

7. DATA PRESENTATION

Plots of history data for operations use are viewed on the operations console display screens; a typical one is shown in Fig. 5. They are labeled with the device and parameter name. The horizontal axis is labeled in hours. Various other data are also displayed such as the data maximum and minimum values, the time of the last data point, the number of intervals, and the standard deviation value. The plots are generated as graphic elements of a general purpose display facility. Their layout is defined in a source file of the same type of syntax as the history source file discussed above. They are selected by buttons on the console touchpanels where an operator can also change the vertical scaling and the time interval of the data plotted.

8. UTILITIES

A utility process can be used to install a data file to be monitored. File installation consists of opening and loading the specified file into memory, and setting a flag so that it is processed immediately. The utility can also be used to reset a file so that the point count is set to zero and the data are effectively erased. The utility can also be used to remove a file from being installed or to list all files which are currently installed.

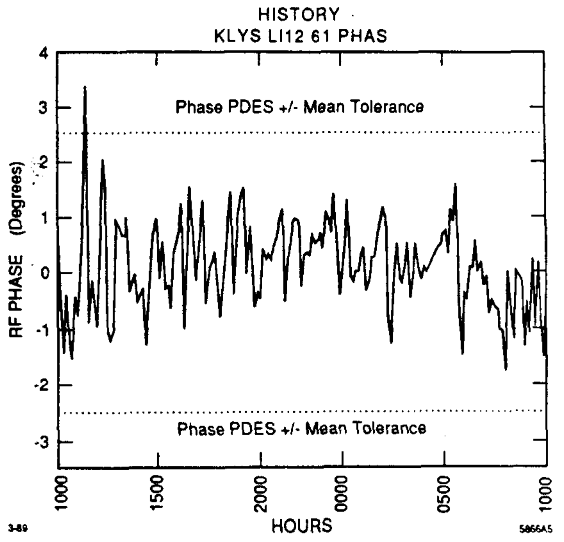

Fig. 5. History plot example.

To obtain a list of the values which have been saved for a given device parameter, a dump process prompts for the file name and the device specification. The output is written to a logical name, which can be assigned as the user desires. The output contains the values, the timestamps for each value, the number of values, the file and device specification, the data format, length, and interval, and the time of the first and last data points.

9. ARCHIVING AND LONG TERM HISTORIES

We are currently saving all magnet output values and the difference between the actual and desired values, all klystron parameters, most temperatures, some timing values, and many miscellaneous parameters. Most data is saved each six minutes, and the number of points saved provides a one day history. This uses about 5% of the CPU time. We have not yet implemented a facility to archive and retrieve data for periods of days or months. We plan to begin taking one-week histories as soon as possible and are discussing methods of saving data sets once a day or week.

ACKNOWLEDGEMENTS

Mike Glaviano and Linda Hendrickson contributed to the plot-scaling software, and Tom Himel contributed the standard deviation calculation algorithm.