SLAC-PUB-205*
July 1966

The SLAC High-Energy Spectrometer Data Processing System*

Richard M. Brown
Mary Anne Fisherkeller†
Anthony E. Gromme
John V. Levy

Stanford Linear Accelerator Center, Stanford, California

Abstract. One of the experimental stations for the Stanford two-mile Linear Accelerator is equipped with three large magnetic spectrometers for scattered electron analysis. To maximize data collection rates for these, an SDS 9300 computer with a sizable I/O multiplexing capacity has been programmed for fast real-time data acquisition and on-line analysis. The system is structured to permit the experimenter to specify his experimental configurations and program requirements in terms of strings of subprograms which can be triggered either by external interrupts, teletype commands, or programmed clocks. The subprograms are symbolically identified and may be either resident or loaded at execution time. A control language for teletype dialogue permits flexibility in the description of the desired configuration and mid-experiment editing of the strings of analysis procedures. The paper describes the data-gathering configuration, the program structure, and the control language of the system.

*Work Supported by the U. S. Atomic Energy Commission.

†Permanent Address: Argonne National Laboratory, Argonne, Illinois.

(Submitted to Computer Issue, IEEE Proceedings)

INTRODUCTION

Experimental End Station A of the Stanford two-mile Linear Accelerator is equipped with three large magnetic spectrometers (1.6-,8-, and 20-GeV) for use in electron scattering experiments. In such experiments data event rates can vary from the maximum pulse rate of the accelerator -- 360 pps -- down to rates measured in hours between interesting events. The spectrometer equipments are massive magnetic-optics and particle detection arrays which can be rotated with high angular precision around a scattering target and which generate information on the angle of scattering, the momentum of the scattered particle, and tentative particle identification. The equipmental configuration including the data collection and analysis system is intended as an experimental resource for use by resident and visiting physicists.

The complexity of the apparatus, the wide variety in experiments, and the high data rates possible argued the development of a computer-based data acquisition and analysis system which could also accommodate programs for the monitoring of the physical environment and the checking of the fast electronic circuits of the spectrometers. A number of experimenters have reported the development of systems for such purposes.[1-5] To a great extent these are directed toward a closed class of experiments requiring only parameter changes between experiments, or else they demand considerable programming ability on the experimenter's part, in particular a knowledge of machine language. Scientific installations such as SLAC, faced with a broad spectrum of users and experiments, need data-handling systems of greater flexibility which minimize demands on the programming ability and time of the experimenters.

This paper describes one approach to the problem -- the SPECTRE system written for a Scientific Data Systems 9300 computer in the spectrometer facility. Principal emphasis will be placed on the data-gathering configuration, the structure of the SPECTRE system within which the various programs are embedded, and the control language used to specify the desired program structure and operation.

OBJECTIVES

The SPECTRE system was developed to control at any one time a single major experiment with possibly a very limited number of concurrent data recording operations for separate experiments. The following, in order of priority, were the functions required:

1. A flexible, event-triggered system for data acquisition, filtering, and recording.
2. An environmental data-logging system (magnet currents, etc.).
3. An on-line analysis and display system to speed up the detection of equipment failures and to aid in the set up and mid-course monitoring of experiments.
4. A set of equipment-checking and diagnostic procedures to be operated concurrently with experimental data taking.

The following considerations influenced the design of the system structure and parameters:

1. The data-gathering procedures are of the essence in the experiments envisioned; they will be unique to each experiment and thus must be easily specifiable by physicists relatively inexperienced in computer

programming. The first implies that some programming must be done by the individual user in setting up his experiment; the second that as a consequence Fortran, as the most common language, must be accommodated.

2. Most computer tasks during an experiment consist of short bursts of program execution triggered by an external signal (e.g., the detection of a successful event in the spectrometer), or by an action of the physicist (e.g., the desire to see a display of a specific set of data), or by the passage of time (e.g., the periodic monitoring of magnet currents, etc.). This suggests the organization of program units into strings of subroutines which can be triggered by external interrupts, commands typed on a teletype, or by an internally programmable clock.

3. Data gathering requires the sampling of approximately 500 bits of information at a maximum rate of once every 2.8 ms. Thus data can flow from equipment to recording magnetic tape as fast as 1.8×10^5 bits per second. This rate forces the data acquisition program to be interrupt driven on the highest priority level and requires reserving a separate output channel for exclusive use of the data-logging tapes. Pulse-to-pulse buffering of the raw data to smooth out statistical fluctuations would be only incrementally valuable.

The need for immediate response to an event trigger also implies that considerable attention must be paid to program sequences of lower priority which disable interrupts for any periods of time (system I/O

control can be a particularly bad offender here). A third consequence of the potentially high data rate is the result that no guarantee can be given that all raw data logged on the output data tape will also be available for digestion and analysis in the lower priority data analysis programs; instead such programs must operate on a sample of the raw data as it passes through the system.

EQUIPMENT CONFIGURATION

Figure 1 illustrates schematically the various elements in the SPECTRE system. Principal components include a 2048-position two-way multiplexer for accessing digital and analog variables of the spectrometer, a 32K SDS 9300 computer with conventional peripheral gear including a two-million character disc for program library storage, a CRT display with light pen, and teletypes to implement a physicist-computer communication system for experiment control and analysis. In addition there is a two-way data link to an SDS 925 computer at the Beam Switchyard which controls certain beam parameters and supplies data on beam characteristics -- pulse amplitude, energy, etc.

Via the I/O multiplexer the computer can ascertain and control the currents in all the spectrometer magnets, monitor their angular position, and sample the stored digital values of pulse height analyzers, coincidence circuits, and counter signals generated in the successful passage of a particle through the spectrometer. All of the connections to the spectrometer electronics are mechanically pluggable to provide the flexibility that the experimental program requires. This creates an uncertainty that

a desired configuration actually exists, but it is one which physicists are acclimated to and prefer.

In addition to the outputs from the fast electronics for data gathering, there are also inputs to most of the critical circuits which permit the injection of test signals for equipment checking between accelerator pulses.

LOGICAL STRUCTURE OF THE SPECTRE SYSTEM

Figure 2 illustrates the structure of the SPECTRE system viewed from the point of view of a physicist user. It consists of strings of subroutines attached to various triggers, which may be external priority interrupts, specific teletype commands, or programmable interval timers. Each of the strings (called SCRIPTS) is defined by the user in terms of a list of subroutine names. The triggering source for each SCRIPT is declared either as the number of the appropriate interrupt or as a unique command name to be assigned to the SCRIPT. Any named SCRIPT may be called by name on the teletype or be scheduled for automatically clocked repetition. During the course of an experiment the user may edit the strings, inserting, deleting, or replacing subroutines to modify program operation.

The subroutines in the strings are standard relocatable binary segments generated by the machine-language assembler or the Fortran compiler, and are stored on the disc until loading is required. Subroutines are generally considered resident, i.e., are loaded upon first being declared in a string. Any subroutines whose execution is to proceed in background may alternatively be designated as load-upon-execute; such subroutines will be loaded each time they are needed. All system subroutines are reentrant in form so they can be shared among several SCRIPTS of different priority.

User-written subroutines, which are also added to the library, need not be reentrant, but the responsibility for their correct use is left to the user. Subroutines, of course, call upon resident system routines for I/O operations and arithmetic functions. In addition system routines have been provided to simplify the common use of the cathode ray tube display.

The system can be seen to be a form of time sharing, except that all time-shared components belong to one user and the scheduling of execution is determined by his allocation of the priority interrupts. Most of the 32 interrupts are of course dedicated to specific system functions (display, light pen, data link, etc.); however, 12 of these are at the user's disposal for his own data gathering and analysis SCRIPTS.

SPECTRE CONTROL LANGUAGE

A set of control statements is available for the user to describe the desired program SCRIPT configurations and to edit and modify these during the experiments. Control statements may either be typed in on the on-line teletype or read from punched cards. Each statement occupies a single line and consists of a string of words or numbers separated by commas. The first element in the statement is the name of a command, either one of the basic commands of the system or one created by the experimenter to label a SCRIPT of his own definition. The remaining elements in the statement are treated like parameters in a Fortran CALL statement. For example, the following command is used to declare a string of three subroutines and to have it loaded and connected to interrupt 045:

SCRIPT, 045, SUBROUTINE 1, SUBROUTINE 2, SUBROUTINE 3

When interrupt 045 occurs, SUBROUTINE 1, SUBROUTINE 2, SUBROUTINE 3, will be executed in that order. A list of the basic system commands is given in Fig. 3. The fact that the user is able to define commands of his own, consisting of SCRIPTS to be executed, permits very flexible expansion of the command list to meet the user's needs. Another useful command, EXECUTE, calls for the immediate loading and execution of any of the programs stored in the library.

The command STATUS provides a compressed but understandable image of the programmed structure. It produces on punched cards a series of command statements sufficient to reconstruct the current SCRIPTS and their connections. This is not only valuable for saving the state of an interrupted experiment, but also provides a means for the structure of one experiment to be used in another with only simple modifications.

It is also possible for any subroutine to call the system executive, transmitting a command message. This permits modification of the program structure as a function of experimental parameters. For example, a program for setting up magnet currents, a time-consuming procedure, may be loaded in as a time interval driven SCRIPT for execution at periods of seconds or minutes. When the desired magnet current is reached, the program can then call the SPECTRE executive and cause deletion of itself.

THE SPECTRE EXECUTIVE SYSTEM

The SPECTRE executive system consists of a command language interpreter, a resident loader and symbol table, a linked list system for stringing together SCRIPTS and attaching them to the appropriate driver, and an error

recovery procedure for disconnecting strings which produce detectable program errors.

Command interpretation consists of a search through the list of acceptable system commands followed by a search through the dynamic file of user-defined commands. All executive action occurs in the background, when no interrupts are active, so as to permit correct SCRIPT editing. Command errors are appropriately indicated on the teletype.

SCRIPT information is stored in sets of linked lists which describe the subroutine elements of the SCRIPT and their entry parameters. Execution of a SCRIPT is controlled by the executive, which traces down its list, calling each subroutine in turn. When an interrupt occurs, a system routine saves the necessary parameters of the interrupted state in a pushdown list, and, after the SCRIPT has been serviced, restores the saved parameters and clears the interrupt. Thus the user is spared the responsibility and the difficulty of careful interrupt processing.

SPECTRE MEMORY MANAGEMENT

The resident loader and symbol table is expensive of memory but necessary for mid-experiment SCRIPT modifications. The editing of program strings creates memory allocation complications since deleted programs remain resident although inactive. Furthermore, external definitions in the deleted programs are not erased from the symbol table. When all memory space is exhausted, the user may request reloading of all his active program strings from the library. This compacts all of the subroutines and frees the unused memory as a single block.

Some SCRIPTS and data areas, however, must not be disturbed under reloading, for example, the data acquisition SCRIPTS. To solve this problem, SCRIPTS and memory allocated to them are divided into two categories: permanent and non-permanent. Permanent SCRIPTS are defined and loaded first, and then the permanent category is declared closed. All SCRIPTS defined after closure of the permanent area are subject to reloading and therefore relocation. Under normal circumstances, a standard set of data acquisition SCRIPTS will be assigned permanent status; non-permanent SCRIPTS will include the variable analysis strings and those loaded for a single execution.

The remaining unused memory serves as a pushdown store for the variables saved by Fortran reentrant subroutines. The loader, which operates at the lowest priority level, must not only disarm the interrupt connected to any SCRIPT it is modifying, but also must disable all interrupts during the brief interval in which the bounds of available memory are being changed.

This memory management strategy is not ideal, but represents a working compromise between the need for fixed program structures for high-speed data acquisition and flexible structures for data analysis and experiment control.

CRT DISPLAY MANAGEMENT

A schematic of the SPECTRE display organization is given in Fig. 4. The display consists of an unbuffered 21-inch CRT with point, vector and character generation capabilities and a light pen. Adjacent to the CRT are six sets of 8-digit thumbwheels with associated on-off switches and indicator

lights. These are used to identify the desired displays and to introduce parameters (e.g., scale factors) into the user's display program.

The display system is designed to permit the on-line user to select up to six concurrent displays on the CRT. For this a file of active display requests is maintained by the system. To enter a request in this file, a user program calls a system file subroutine providing as parameters: 1) the set of data; 2) an identification code for the data; and 3) a user function subroutine which will be called to transform the data into displayable form. If light pen action is desired with the data of the request, an additional user subroutine is specified. Subsequent calls with the same request resulting from re-execution of the calling program cause its file entry to be flagged for refreshing of the display data.

The display driver is triggered by a 30-cps interrupt. At this time the identification codes on the external thumbwheels are interrogated to ascertain the current requests of the on-line user. If any of the identified data sets are flagged for refreshing or if any of the codes are changed, the user's subroutines are called to transform the data and refill the programmed display buffer (1000 points or characters). The CRT is then driven from this buffer. When refreshing is not necessary, the buffer is immediately displayed without change. If no entry is found in the file corresponding to one of the codes on the thumbwheels, an indicator light adjacent to the thumbwheel set is turned on to alert the user.

Upon light pen interrupt, the associated display point in the buffer is traced to its corresponding file entry. If no light pen action has been

requested, the interrupt is ignored; otherwise, the user's second subroutine is called and supplied with the coordinates of the point detected by the light pen and a pointer to its location within the buffer. The point is also intensified briefly before resuming display of the entire buffer.

The use of a display request file in conjunction with thumbwheel switches permits rapid and convenient switching of the data display for on-line analysis. In addition time and memory space are saved, since only the data currently selected for display will be converted and stored in the display buffer.

CRITIQUE AND CONCLUSIONS

The design of a multi-purpose on-line system such as the SPECTRE system requires a number of compromise decisions. To others designing similar systems some comments regarding specific properties of SPECTRE may be of value.

The decision to accommodate Fortran subroutines resulted in a significant memory expenditure for supporting library and system subroutines. The resident monitor and command language interpreter require about 6000 words. About 2500 words of memory must be allocated to the resident loader and symbol table as the cost of on-line subroutine loading and linking. In addition the use of a programmed display buffer and control brings the total resident system to about 10,000 words. Proposed modification of the disc operations should reduce this to 7000 words.

The entire system operates within an SDS Real-Time Monitor System which does not queue the I/O calls. Since many of the I/O operations deal

with slow devices such as typewriters and plotters, there can be considerable congestion in use of the single channel available; a multiplexed channel would be more appropriate.

No memory protection is provided, since the dynamic protection techniques required are unavailable on the 9300. The hazards of unprotected time sharing are only partially ameliorated by the fact that there is only one user at a time. Program bugs can be exceedingly baffling. Modern forms of memory protection would also be very helpful in the problems of memory management.

The need for corrections or modifications of analysis subroutines is often only apparent during an experiment. On-line compilation as part of a background mode would be very helpful, so that experiments could continue while new subroutines were created and added to the library. This is impossible with the present system.

The SPECTRE system has been in use since July 1966. Despite the weaknesses mentioned, it provides a flexible and powerful framework for fast data acquisition in conjunction with concurrent on-line data analysis and control. In particular it permits a physicist relatively inexperienced in programming to develop easily a program structure tailored to his unique needs. In the development of the SPECTRE system the helpful criticisms and contributions of Dr. Adam Boyarski are gratefully acknowledged.

REFERENCES

1. Richard G. Allas, "The NRL Cyclotron Data Acquisition and Processing System", IEEE Trans. Nucl. Sci. NS-12, pp. 527-529, June 1965.
2. G. Krueger, "The Introduction of Integrated Computers into Nuclear Physical Experiments", Atomwirtshaft 10, pp. 110-118, March 1964.
3. J. Leiss, "The NBS On-Line System - Designed for Flexibility", Nucleonics 22, pp. 57-62, December 1964.
4. J. F Whalen, J. W Meadows, and R. N. Larsen, "On-Line Operation in Nuclear Physics Experiments", Rev. Sci. Instr. 35, pp. 682-690, June 1964.
5. See also: a) Proc. of EANDC Conference on Automatic Acquisition and Reduction of Nuclear Data, Karlsruhe, July 1964.

   b) Purdue Conference on Instrumentation for High Energy Physics, IEEE Trans. Nucl. Sci., NS-12, August 1964.

   c) Proc. of Conference on the Use of Computers in Analysis of Experimental Data and the Control of Nuclear Facilities, Argonne National Laboratory, May 1966.

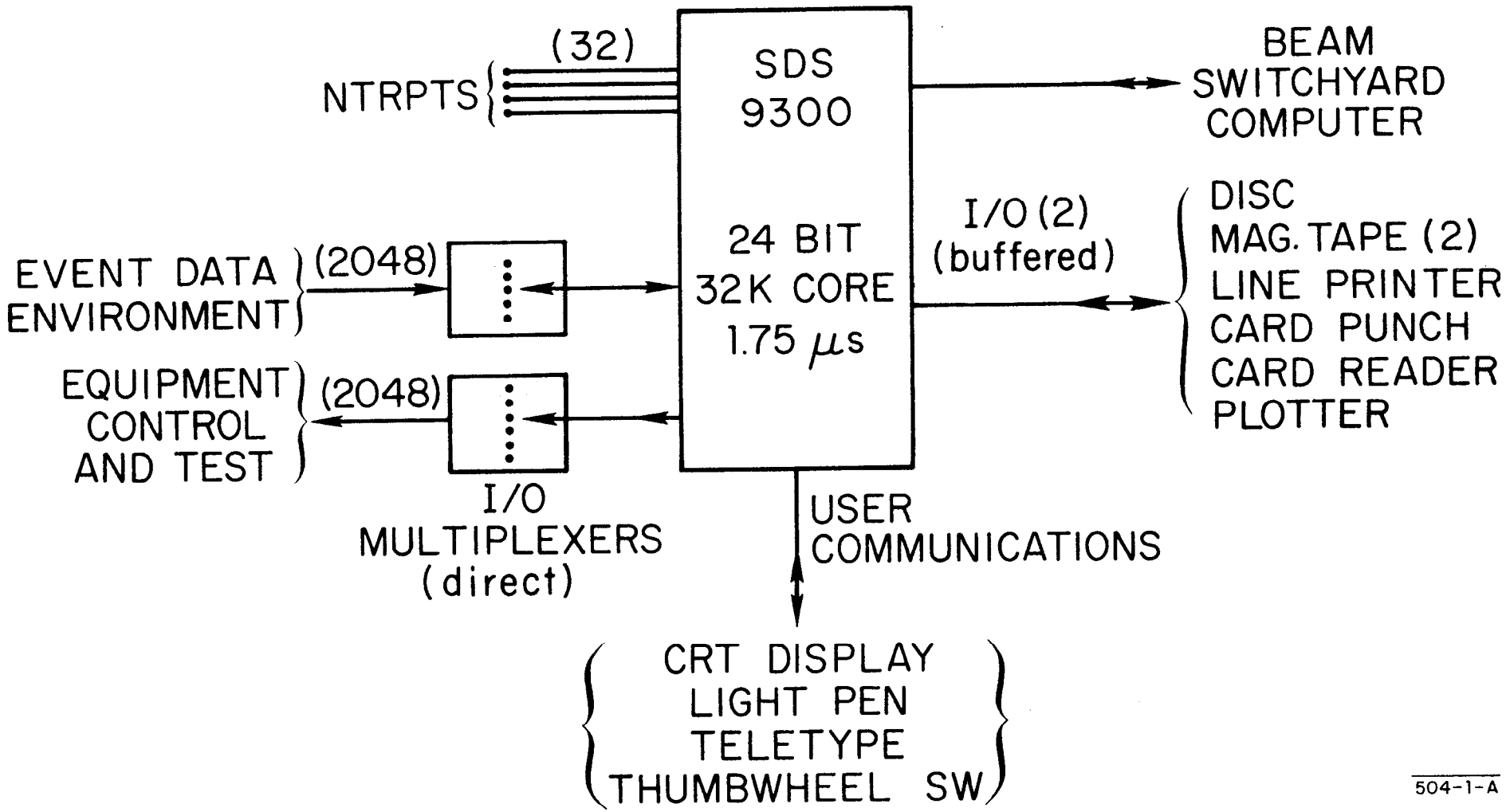
FIGURE CAPTIONS

Fig. 1. SPECTRE System Schematic.

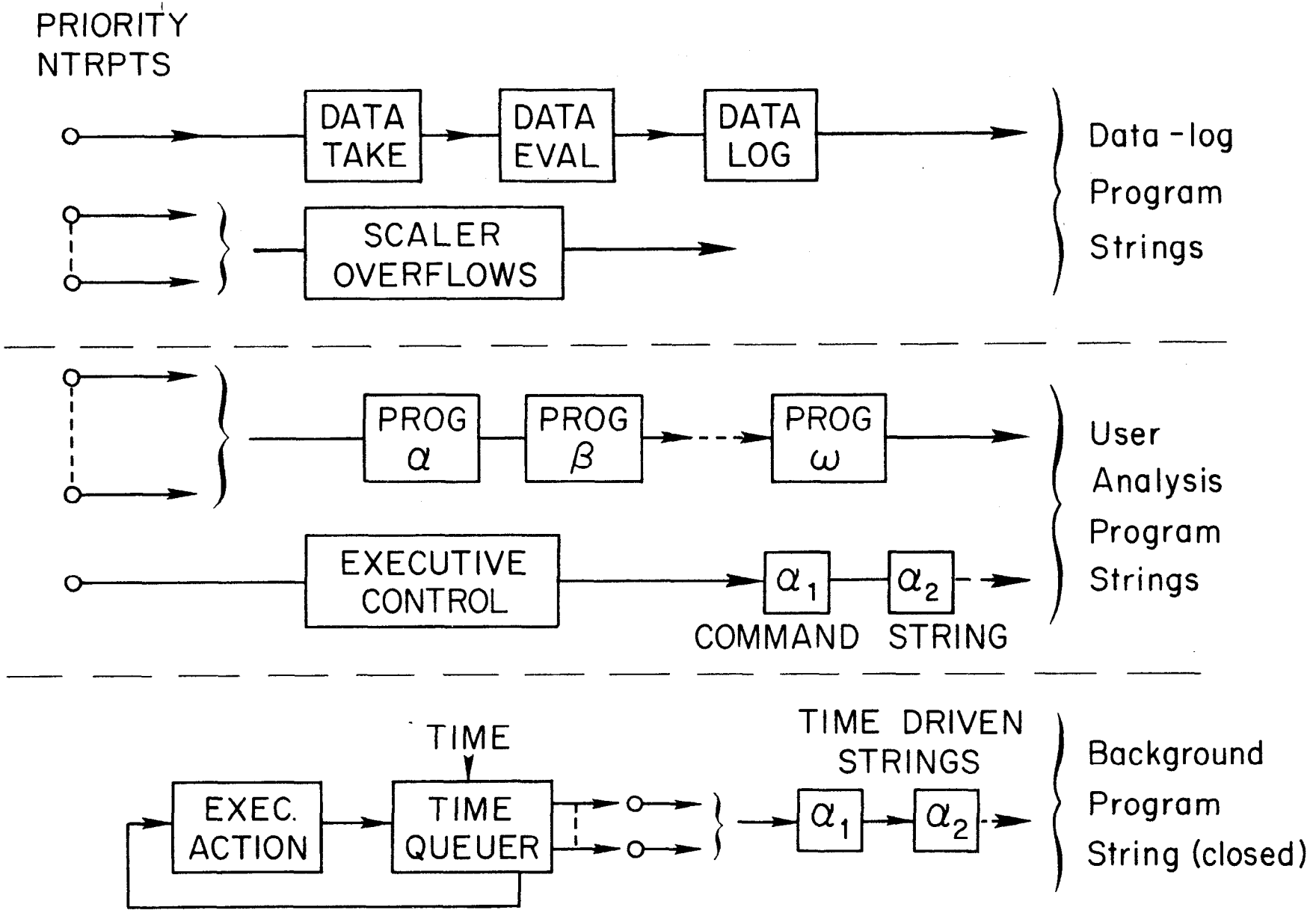
Fig. 2. SPECTRE program string structure as viewed by the user. Interrupt priorities increase toward the top.

Fig. 3. Basic SPECTRE system commands with parameter definitions enclosed by < > , descriptive information by ( ).

Fig. 4. Organization of the SPECTRE display.



SLAC SPECTROMETER COMPUTER CONFIGURATION



SPECTRE PROGRAM STRING STRUCTURE

520-1-A

BASIC SPECTRE CONTROL STATEMENTS

SCRIPT, <interrupt or command name>, <subr 1>, <subr 2>,....

EXECUTE, <subr 1>, <sub 2>,....

INSERT, <script identifier>, <preceding subr.>, <subr. string>

DELETE, <script identifier>, <subr. string>

REPEAT, <script identifier>, <start>, <repeat interval>, <stop>

RELOAD (load all non-permanent strings)

ARM, <interrupt>

DISARM, <interrupt>

CONNECT, <interrupt>, <program>

QUERY, <variable name>

SET, <variable name>, <desired value>

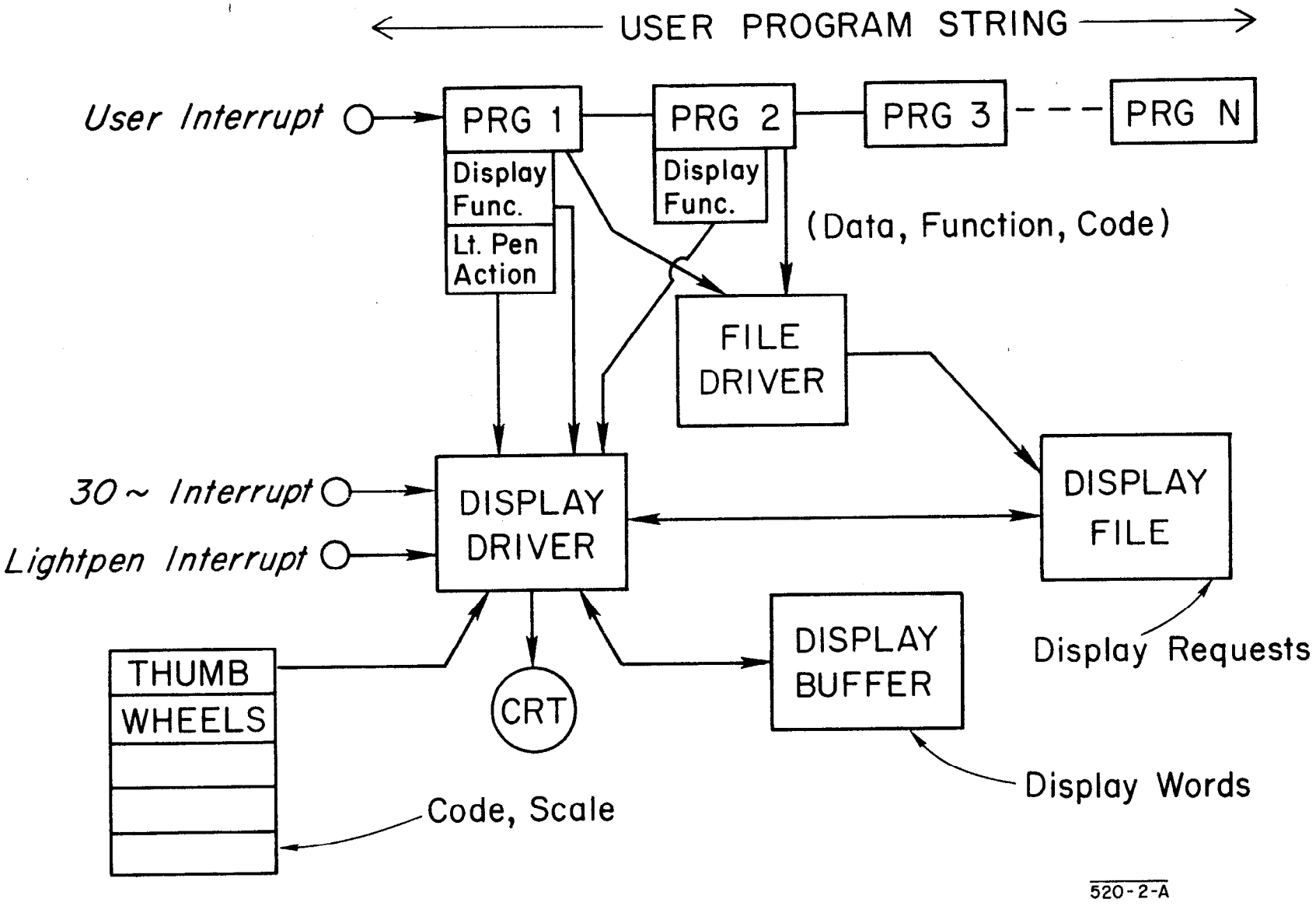
CONTROL, <device for subsequent executive control>

STATUS (punch out control statements)

CLOSE (permanent script loading completed)

OPEN (resume permanent script loading)

504-2-A



SPECTRE DISPLAY SCHEMATIC