UCRL-16481

University of California

Ernest O. Lawrence Radiation Laboratory

TRIST--A TRILEVEL MULTIPROGRAMMING EXECUTIVE COORDINATING REAL-TIME PROCESSING OF DATA FROM AN ASYNCHRONOUS ON-LINE DEVICE WITH DATA ANALYSIS AND BACKGROUND PROGRAMS

UCRL 16481

For 1966 Spring Joint Computer Conference
Boston, April

UNIVERSITY OF CALIFORNIA

Lawrence Radiation Laboratory
Berkeley, California

AEC Contract No. W-7405-eng-48

TRIST--A TRILEVEL MULTIPROGRAMMING EXECUTIVE COORDINATING REAL-TIME PROCESSING OF DATA FROM AN ASYNCHRONOUS ON-LINE DEVICE WITH DATA ANALYSIS AND BACKGROUND PROGRAMS

Carol Osborne, Robert Larson, Thomas Oliver,
Joan Pinchak, and Howard S. White

October 26, 1965

TRIST--A TRILEVEL MULTIPROGRAMMING EXECUTIVE COORDINATING REAL TIME PROCESSING OF DATA FROM AN ASYNCHRONOUS ON LINE DEVICE WITH DATA ANALYSIS AND BACKGROUND PROGRAMS

Carol Osborne, Robert Larson, Thomas Oliver
Joan Pinchak, and Howard S. White

Lawrence Radiation Laboratory
University of California
Berkeley, California

October 26, 1965

Abstract

TRIST is a trilevel multiprogramming executive designed both to coordinate computer operations with a real-time asynchronous device attached on-line to an IBM 7094 II and to make the computer time between real-time demands available for other uses. Although designed for these specific purposes, many of the ideas incorporated in the program are basic to any multiprogramming problem, and several of the special features included in the memory-protect routine, priority handling of subroutine use, loop and halt detection, and quality control are of particular interest.

Introduction

The basic problem in the program design was to utilize the computer time between real-time demands from an on-line film-scanning and digitizing device (Flying Spot Digitizer--FSD; see Appendix A)[1,2] while meeting the data-transmission demands of that device. The digitizing device should not be stopped or delayed, therefore there is a time restriction of 500 microseconds after a demand within which CPU (central processing unit) control must be given up for real-time processing. Computer time is available because real-time processing of the data transmitted from the FSD can be completed before the next transmission demand occurs.

Machine Configuration

The TRIST executive is written for an IBM 7094 II with two 7302 core storage modules (65k), and six data channels: one with a 1301 disk file; one with an LRL-built interval timer which interrupts every 137 milliseconds; one with the on-line device; and three tape-unit channels, one of which has a printer, punch, and reader as well as tape units. See Fig. 1.

Core Storage Structure

TRIST logic occupies 20_8k locations of Core A, and the input-output routines plus Core A-Core B communication linkages occupy 5_8k locations of Core B. The real-time program occupies Core A from 20_8k up, and Core B is available from 5_8k up for other programs. See Fig. 2

Priority

The priority requirements are unique in that the real-time program has the highest priority at all times. In addition to keeping the FSD busy, the executive must handle the priorities of batches of data and

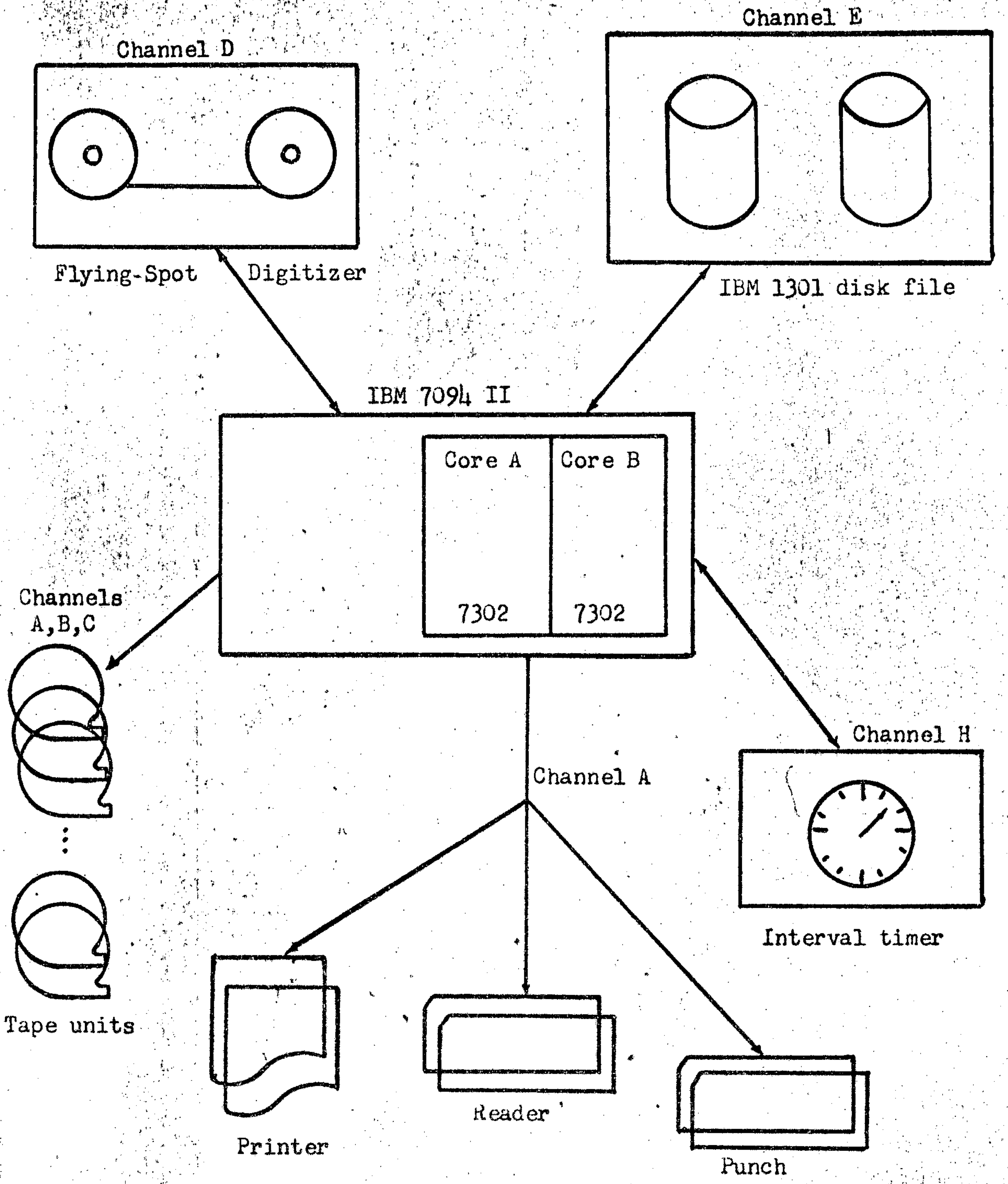

Fig. 1. Machine configuration.

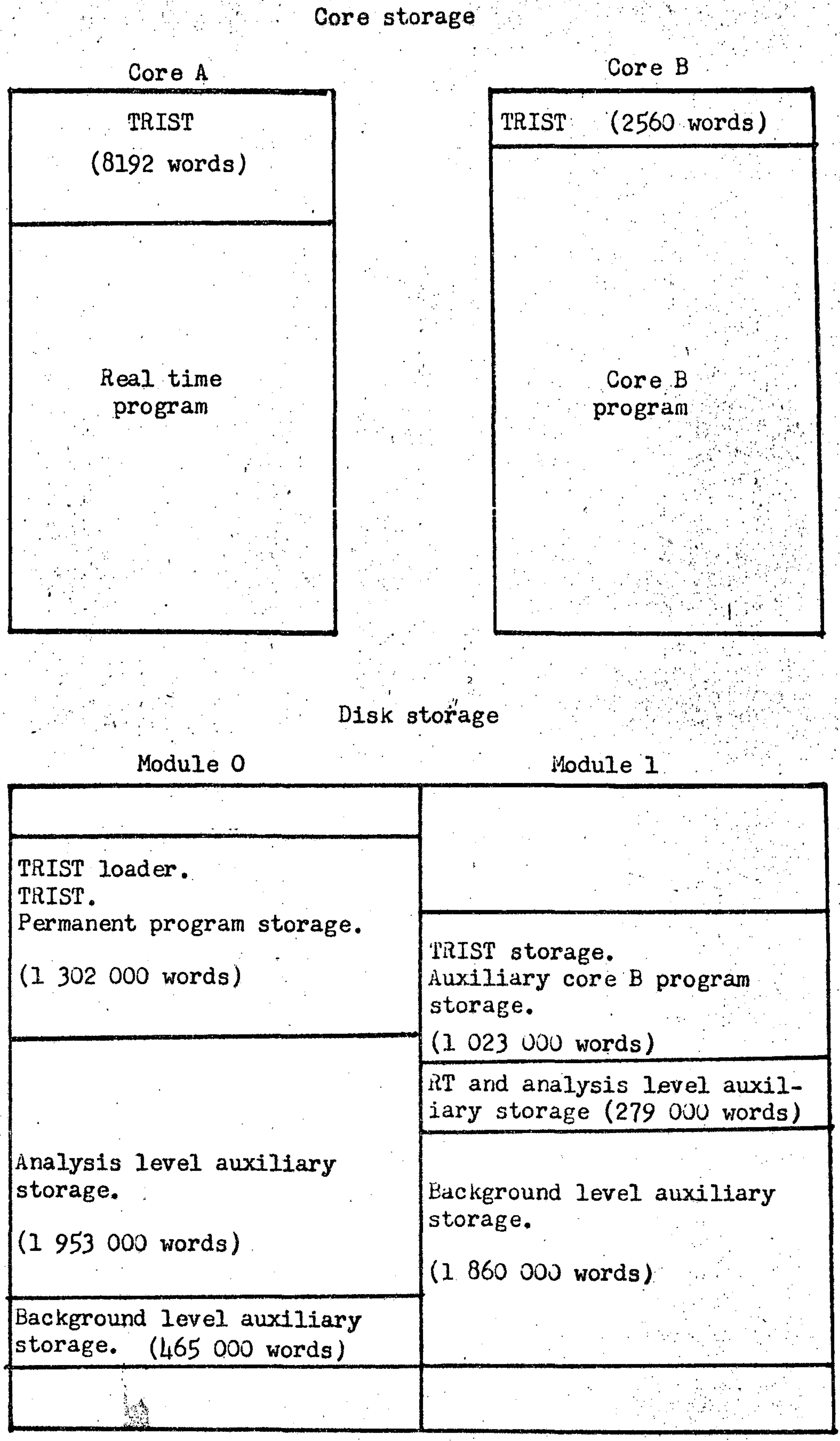
Core storage
Core A
TRIST
(8192 words)
Real time program
Core B
TRIST (2560 words)
Core B program
Disk storage
Module 0
TRIST loader.
TRIST.
Permanent program storage.
(1 302 000 words)
Analysis level auxiliary storage.
(1 953 000 words)
Background level auxiliary storage. (465 000 words)
Module 1
TRIST storage.
Auxiliary core B program storage.
(1 023 000 words)
RT and analysis level auxiliary storage (279 000 words)
Background level auxiliary storage.
(1 860 000 words)

Fig. 2. Core and disk storage.

of program levels. Three "levels" of programs are recognized: the real-time (RT) program, which processes data from the FSD; the production programs, which analyze these data; and background programs, which may be debug or other production programs. The second-level (analysis) programs are dependant upon the real-time program for data, however, the other two levels can be run together or, if necessary, separately, meaning that the third-level (background) programs can continue if the FSD is down.

Second highest priority is given to the data-analysis programs. There can be a maximum of 12 analysis programs, therefore batches of data are cycled through a string of these programs. Lowest priority is given to the third-level background programs, thus the executive must know the status of data and the status and order of the string of analysis programs while also keeping track of the current background program. This priority handling of data and levels must be done with the added complication of meeting the demands from the FSD whenever they occur.

Control is maintained via a set of four entry blocks (the real-time, non-real-time, executive, and core-B blocks) in which machine conditions and identification information are saved. There is only one core-B program at any one time, but it can be any one of the string of analysis programs or the current background program, depending upon the priority of data. Therefore, when a program is stored in auxiliary storage, the entry block is stored with it and, in turn, read back in when the program is recalled.

When a real-time demand occurs, the status and machine conditions of the current non-real-time (NRT) program (which may be either the executive itself or the core-B program) are saved in the NRT entry block, and the RT program is entered via its entry block. Then, when the RT program gives control back to the executive, the data status and priorities are tested and a NRT program will be entered via an entry block. For instance, if a third-level background program was running when the RT demand occurred and there are data ready for processing when the RT program gives up, the NRT block (which was third-level information) is moved to the core-B block and the core-B block and program are moved to auxiliary storage. The first of the string of analysis-level programs is then recalled from storage and its entry block is moved to the NRT block, which is then entered. A summary flow chart is seen in Fig. 3.

Quality Control

One of the problems connected with a high-volume-data-processing system is quality control. The analysis-level programs run under TRIST make it possible to have the analyzed results of a night's FSD run within a very short time after the end of the run. This feature will become even more important within the next few months, when the RT program will be doing pattern recognition on the input data.[3] It will be necessary to have analysis of the bubble chamber events as soon as possible after a run, to check both the digitizer and the program results.

Memory-Protect Features

In addition to its obvious function of protecting the executive from user errors, the memory-protect package is used for subroutine sequencing, i.e., to protect the library subroutines from multiple calls. This feature allows all the library subroutines to remain in memory as part of the executive, thus saving a large amount of both memory space and input-output (I/O) time. Otherwise, all library subroutines needed by

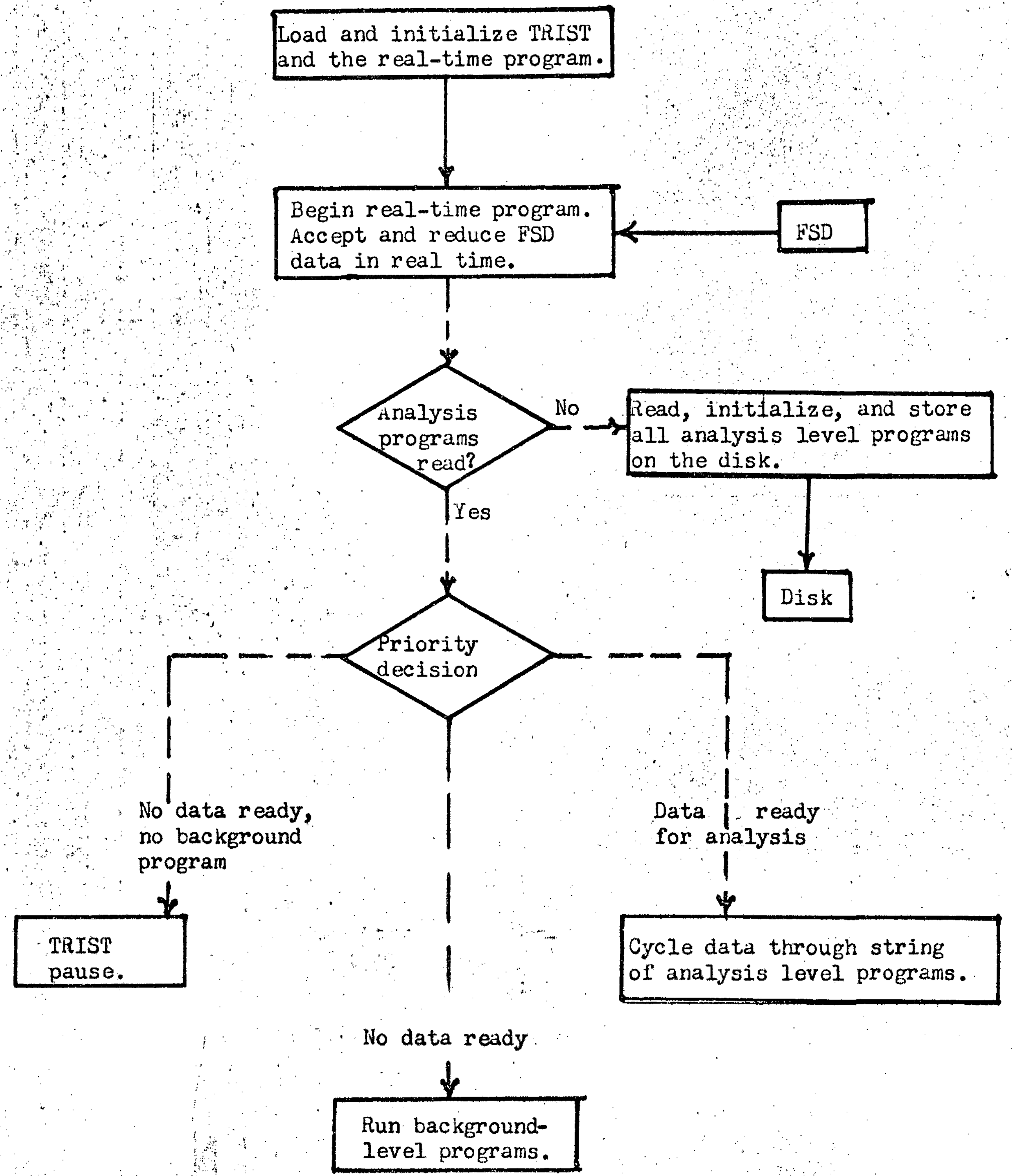
Load and initialize TRIST and the real-time program.
Begin real-time program. Accept and reduce FSD data in real time.
FSD
Analysis programs read?
No
Read, initialize, and store all analysis level programs on the disk.
Disk
Yes
Priority decision
No data ready, no background program
TRIST pause.
Data ready for analysis
Cycle data through string of analysis level programs.
No data ready
Run background-level programs.

Fig. 3. Summary flow chart.

a particular program would have to be read in with that program.

Setting the memory-protect limits causes an interrupt (trap) to occur whenever a location within the specified limits is referenced, and control then goes to a memory-protect routine. Since the library subroutines are within the protected region, control goes to the memory-protect routine whenever a library subroutine is called. Multiple calls to a single subroutine can occur if, for instance, a background program was in the middle of a library subroutine when the real-time program was given control and then the RT program subsequently calls that subroutine. The memory-protect routine allows the prior use of the subroutine to complete, saves the machine conditions of the subroutine exit in the proper entry block, and then allows the current subroutine call. The proper entry block is always the NRT block, because control has gone only into the RT program from the current NRT program, therefore the executive has not had a chance to juggle the entry blocks for priority demands (this happens when control goes from the RT program to the executive). The RT program never requests a library subroutine during its actual RT operation, therefore all conflicts of subroutine calls are solved and there is no interference with RT demands.

Time Accounting

There is a problem in keeping track of the time per program, because the executive shifts control among the programs as required by the priorities of the FSD and of batches of data. This problem is solved via a set of clock blocks, which are stored and recalled with the individual programs as are the entry blocks. Each time a program is entered or interrupted, the time is read and used time is calculated and saved. The executive then produces a final run summary for itself and for each program, which includes priority level, a time-charge number, time (starting, ending, and used), run type (debug, production, and preproduction), and a completion code (normal, interrupted, restarted, etc). This run-summary information is also stored on the disk for use by a time-accounting program which produces weekly, monthly, and yearly time-accounting reports.

Halts and Loops

Since the executive is designed to keep the FSD operating and since there is a minimum time within which control must be given to the RT program after an FSD demand, it becomes obvious that something must be done to handle program halts and loops. (Halts would cause the FSD to stop, and loops may not be recognized by the operator for some time, since programs are being flipped in and out.) This problem is solved with an interval timer attached to one of the computer data channels. This timer causes an interrupt (trap) to occur every 137 milliseconds. When a timer interrupt occurs, control is sent to a timer service routine, which examines the current machine conditions for halts and loops. When a halt is detected in a debug program, the location of the halt is written, a core-memory dump is written, and the next program is called. When a halt is detected in a production program, the location of the halt is written and the timer routine attempts to set up a restart of the program.

Loops are detected by an I/O timer word which is part of each program's entry block. Since all programs are of a data-processing nature, a maximum permissible elapsed time (approximately four seconds) between I/O requests can be established. The I/O timer word is incremented at each timer trap and cleared when an I/O request is recognized. When the timer word exceeds the established maximum, a statement is printed on-line and the computer operator may take corrective action.

Operating

Setting up the operating procedures for the system posed the same type of problem as program loops and halts. Since halts for operator action (tape requests, etc.) would halt the FSD, operator action "pauses" are set up in which a request for action is printed on-line and a pause loop is entered. In this way, the RT demand interrupts and interval timer interrupts can continue while the operator is performing the requested action. The operator then must have a way to signal the executive that the action is completed. The sense switches are used for this purpose.* Operator action sense-switch signals are handled by the interval timer service routine. In addition to checking for halts and loops, this routine checks and remembers the status of the sense switches. Each priority level has a specific operator action sense switch with which the program can request operator action or with which the operator himself can initiate action (for loop handling or interrupting), thus three sense switches are used for operator action. For example, when a program requires operator action, it prints a request on-line and goes to a "pause" subroutine. The operator then performs the requested action and puts the specified sense switch down. The timer service routine will recognize that the switch is down the next time it is in the specified priority level for that switch and will print an acknowledgment statement. The operator then puts the sense switch up and the program continues. In practice, this amounts to the operator's essentially flipping the sense switch down and up--only rarely is there a delayed reaction.

*The console keys are used as sense switches by each program--ten are provided for each level--and the executive has "sense switch" testing subroutines to test these in the same way as a sense switch is tested.

The operator may wish to initiate action to force an interrupt or a restart in a program. In this case, he puts down the sense switch for the specified level which will cause an on-line statement and then a pause in that specified level only. At this point, the operator may put the sense switch up or may indicate a transfer by entering a command in the keys and then putting the sense switch up. (He signals that the keys are to be used as a command rather than as sense switches by using the fourth sense switch. The fifth and sixth sense switches are used for interrupting first- and third-level processing.) The timer service routine checks to see that the transfer is to a "legal" area for that level. Important communication links exist between the memory-protect routine in its feature of multiple subroutine call protection and the timer service routine when it allows direct operator action. Depending upon the status of the memory-protect routine, the timer routine may not immediately execute a desired transfer, but may set the location to which the memory-protect routine will transfer. It should be noted that operator transfers are done only as an emergency procedure: to end a third-level program, to attempt a restart if a production program is in trouble, etc.

Sense Lights

The sense lights give the operator an indication of the level that is currently in control. There is one sense light for each of the three priority levels and one for a pause routine; all off indicates executive control. If, for instance, the sense lights are flashing between 1 and 4, there are no background programs--only the RT program is running. The upper half of Fig. 4 shows the TRIST interrupts and the lower half shows TRIST control.

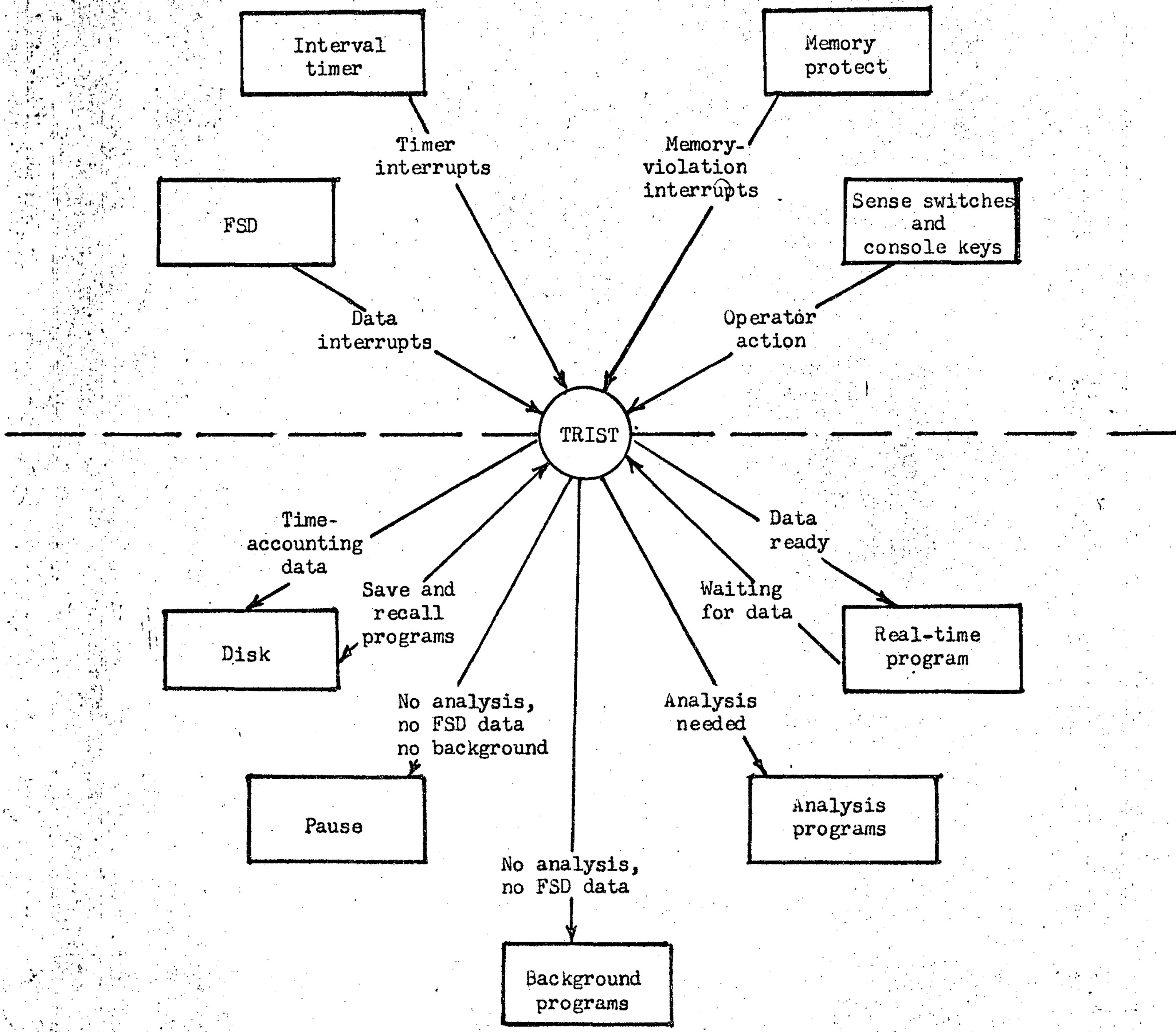

Fig. 4. TRIST interrupts and control.

Program Decks and Skipping

The programs to be run are set up as a deck of cards consisting of a program card specifying program and level, a time-charge card specifying charge number and run type, modifications, if any, and an execute card. At the beginning of the run, the RT program deck is read and the FSD is started, the analysis programs are initialized and moved to auxilary storage, and then the third-level program decks are read and stored on a disk area. Third-level programs are then run in order from this disk area. The operator can skip backward or forward in the list of these programs via a specified console "skip" key. When the skip key is put down, identification of the next program in the program list is printed on-line with its serial number indicating its relative position in the program list (the fifth program to be run has serial number 5). The operator may select that or any other program in the list by indicating its serial number or he may terminate all third-level programs by selecting serial number zero.

Additional Decks

When all third-level programs have been run or if no third-level programs were originally read, a deck can be read via a "more cards" console key. This key initiates the reading of third-level program decks onto the disk from which they are run. This means, for instance, that a RT run can be started and successive third-level runs can be initiated during the continuing RT run. If desired, any particular third-level list of runs can be terminated and a higher-priority list of decks can be initiated without interrupting the other levels.

Debugging

With three interrupts (FSD, interval timer, and memory protect) going at once, timing became a large factor in the debugging of the executive. It was difficult, in fact almost impossible, to get a bug to repeat because of the changes in the timing of these interrupts. The interval timer interrupt could be operated with a manual switch, which saved a great deal of confusion. Otherwise, once a bug occurred, one might be several routines (and bugs) away from the original problem by the time a halt occurred. Also, it was necessary to add special features to take care of the problem of simultaneous interrupts.

Conclusion

Approximately 17 man-months of effort have gone into the development of the TRIST executive; however, in one month, the computer time saved by the multiprogramming operation equalled the development cost. The executive has been in operation since February of 1965 and successfully coordinated computer operation with the on-line, asynchronous FSD, and simultaneously makes available nearly 60% of the CPU time for other use.

The large computing capacity of the 7094 is required during real-time processing, but the large amount of time otherwise available needed to be used efficiently; therefore, the TRIST executive was developed.

Appendix A

The on-line device is a Flying Spot Digitizer (FSD),[1,2] which is used to digitize pictures of nuclear interactions occurring in bubble chambers. The FSD scans across the film, digitizing coordinates of images along the scan line. These coordinates are then saved in a buffer (block of storage) and transmitted to the computer when a buffer is filled. Scan lines containing about 40 words of data are produced every 2 milliseconds. Once the scan of a picture is begun, the FSD operates asynchronously, and nearly 100 thousand words of information may flow into the computer during the several seconds required to measure a large picture. Since the computer storage available for these data is only a few thousand words, the data must be analyzed and reduced in real-time while they are being transmitted.

The real-time program reads data into alternate buffers, processing the data in one buffer while the other buffer is being filled. There are two logical delay periods in the real-time program. One occurs when the program has processed one buffer of data and is waiting for the other buffer to be filled and the other, longer delay occurs when the FSD is advancing the film to the next picture. The lengths of the delay periods vary with the size of the bubble chamber image on the film, but an average of 60% of the CPU time during an FSD run is available.

References

1. P. V. C. Hough and B. W. Powell, in Proceedings of an International Conference on Instrumentation for High Energy Physics, Berkeley, 1960 (Interscience Publishers, N. Y., 1961), page 242-245.

2. H. W. White, et al., Preliminary Operating Experience with Hough-Powell Device Programs, Nucl. Instr. Methods, Vol. 20 (1963) pp. 393-400.

3. Dennis Hall, Joan Pinchak, H. S. White, The DAPR Production System. A Paper delivered at the Conference on Programming for Flying Spot Devices, Columbia University, October 1965.