DIGITAL RECORDING AND ANALYSIS OF SIMULATOR OUTPUTS

MR. E. A. ROBIN
International Business Machines Corporation
Federal Systems Division

Simulator computer programs should be designed to capitalize on the latest training innovations in data recording and analysis. Related equipment should be configured for operational training and instructor guidance and include four main considerations in the design of data recording and analysis programs.

1. The recording function should be integrated into the on-line data collection system to preserve program system integrity of the operational simulator programs.

2. A data collection executive and modular on-line recording routine can effectively use computer memory and execution time resources.

3. The post-run analysis program can provide human factors engineers the media for operator analysis without the usual delays of a data reduction facility. The actual simulator digital computer can be programmed to give the instructor, human factors engineers, and other pertinent personnel a set of reduced training data reports or analyses immediately after a simulation session. The ability to quickly reduce data gives all key personnel a unique selection of data reduction programs necessary to evaluate the simulation run just completed. Human factors engineers can enter their judgments in selecting the data reduction programs best suited to the situation.

4. The dynamic program selection of post-run analysis programs directly after a simulation run gives meaningful results immediately. In contrast, normal data processing procedures process all the data from daily simulation sessions in a batch process manner, usually generating a large amount of paper. Using this daily procedure a more substantial problem results when the instructor wishes to find a specific situation for man/machine evaluation.

Two systems—Federal Aviation Agency Computer Driven Simulation Environment (CDSE) and the TACFIRE Training Support System (TSS)—will emphasize data recording and data evaluation. The first system has been implemented and actual results will be given. The second system is designed and design considerations will be discussed.

SOFTWARE INTEGRITY

Realism in a digital training or simulation environment can be obtained best by using the actual Operating System, application programs, and realistic system inputs. Application programs should be used intact in the overall software set. A straightforward method to insure system integrity is to add the data recording programs to the software system as separate program modules linked to the operation programs in a non-interfering manner.

This approach to data recording software allows complete checkout and debugging of the operational programs without altering the operational program coding. Several problems in this method can arise from inadequate memory space to accommodate these data recording modules. A programming method often used to overcome this problem is to treat the simulation or training operational program as a multi-programming environment.

The data recording programs can be stored in an auxiliary memory device, e. g., a disk file, magnetic drum, or auxiliary core memory. The data recording program module can be called into the main storage memory (Figure 47) as needed. Normally the data recording module is called into core at a fixed cyclic rate based on a submultiple of the application program sampling rate. To preserve system integrity, this scheme must be evaluated against the data recording resolution requirements, the write/read time to overlay the

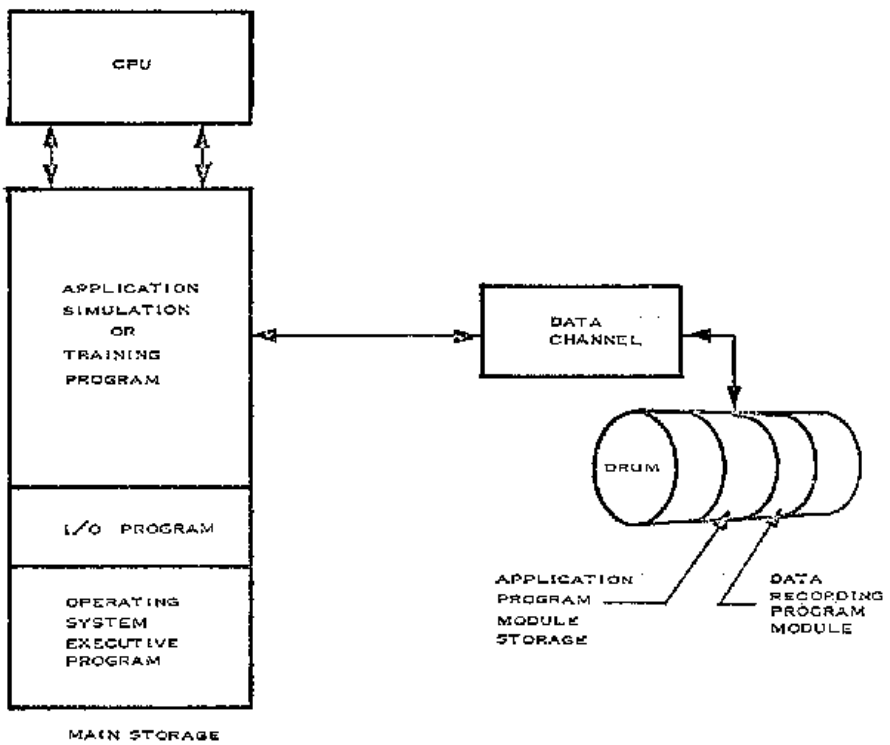


Figure 47. The Use of Drum Memory For Data Recording

application program, and the complexity of data recording tasks. The advantages are:

1. Core space is conserved. The application program module and the data recording program module can be checked out as separate units. Then the input/output (I/O) program to overlay can be added to integrate and debug the programs.

2. If the cycle time is exceeded on some cycles, the data recording overlay need not be loaded into core and executed during that cycle. As a consequence data is lost; however, the sample rate will remain in synchronism.

3. If execution time permits, a complex set of logic for data recording can be used because adequate core space is available for implementation. This allows on-line screening of data during the execution of the recording program.

Sharing the memory between two or more programs in a time dependence manner is called multi-programming. The degree of multi-programming varies considerably from one type of system to another, but always it adds complication to program writing and particularly to program testing.

If adequate core storage is available, the data recording program module can reside in core memory along with the applications program. This eliminates the need for multi-programming. The overhead running time needed to "swap" the application simulation program module and the data recording module is then saved.

A simple scheme exists for linking the data recording program module with the application program module that preserves software integrity. The link the application programmer uses to transfer control from his routine to the data recording program module can be a MACRO-instruction. This line of code, after the application program is compiled, generates numerous instructions which give the required linkage to data recording program module. In early checkout of the application programs, the data recording MACRO's are expanded into no-instruction operations (NOP); i. e., they do not transfer to the data recording program modules. When the time comes to integrate the data recording program modules into the system, only the MACRO needs redefining.

For example, the MACRO "LOG" may be defined as:

Label	Operation	Operand
LOG	MACRO	BUFFERI, X, Y, Z, T

It may expand into six instructions when compiled.

Location	Operation	Operand
i	SLJ	LOG
i + 1	NOP	BUFFERI
i + 2	NOP	X
i + 3	NOP	Y
i + 4	NOP	Z
i + 5	NOP	T

The first instruction at location i transfers program control to the logging routine (LOG) in the data recording program module. The locations i + 1, i + 2, i + 3, i + 4, and i + 5 contain the operands such as BUFFERI, address of the data recording buffer area, and S, Y, Z and T—the variables to be recorded. Then the data recording function is integrated throughout the application program without destroying software integrity. This MACRO standardizes

all recording functions (Figure 48). This is only one method of approach. Another plan is periodically sampling the data for recording in one program each time through program cycle and recording all data one time per cycle.

A third alternative is to record data by triggering an event, e. g., an input or output interrupt. When an interrupt occurs, a sequence of computer actions is necessary. Some of these will be programmed and some will be automatic, depending on the sophistication of the hardware. The sequence of events in a system with only one level of interrupt is:

1. Further interrupts will be inhibited or prevented from occurring until the input/output interrupt is serviced.

2. The location where the application program was interrupted will be stored so a return can be made to that point when the data recording routine is over.

3. Certain indicator registers must be preserved, as the contents or setting of these will be needed again by the interrupted application program.

4. Required action must be determined and control transfered to the data recording program module. Cases may result where only the request for data recording is queued and the actual data recording will be done later.

5. Program control now is returned to where the applications program was interrupted. The indicators, etc., must be restored.

In either method be careful to preserve system integrity. In the third case, the problem gets a little more difficult as data recording timing becomes a critical factor in servicing the interruption.

A method used successfully to minimize this service time is to stack only the request for data recording during the interrupt service period. On completion of servicing interrupts, the processing of data recording requests is executed. This can be accomplished during the sample cycle as time permits.

DATA RECORDING PROGRAM MODULARITY

The tasks of digital recording and analysis can be divided into three parts: initialization, operational data recording, and data analysis. Initialization is necessary to write the proper set-up and identify information on data recording media (disc file or magnetic tape). This identification data is needed for historical purposes and the set-up parameters usually are needed by the data analysis and reduction programs.

Often each application program routine has its own initialization requirements. For example, programs are written separately by six different programmers. However, initialization can be accomplished by a separate data recording initialization routine that simultaneously starts all six programs, insuring proper system data recording initialization.

The operational data recording program module can be organized as a data recording executive and modular on-line data recording routines. In this manner new on-line data recording routines can be added easily to the data recording program module.

Some data may be stored by programs other than those in the data recording program module. This data is computed by the application program and stored in a special area of computer memory. At the end of the training or simulation session this data is recorded on the recording media (disc or tape) by a separate "clean up" data recording routine. The "clean up" need not reside in the core memory. It also should write all partially filed data tables on the recording media to insure a complete set of recorded data.

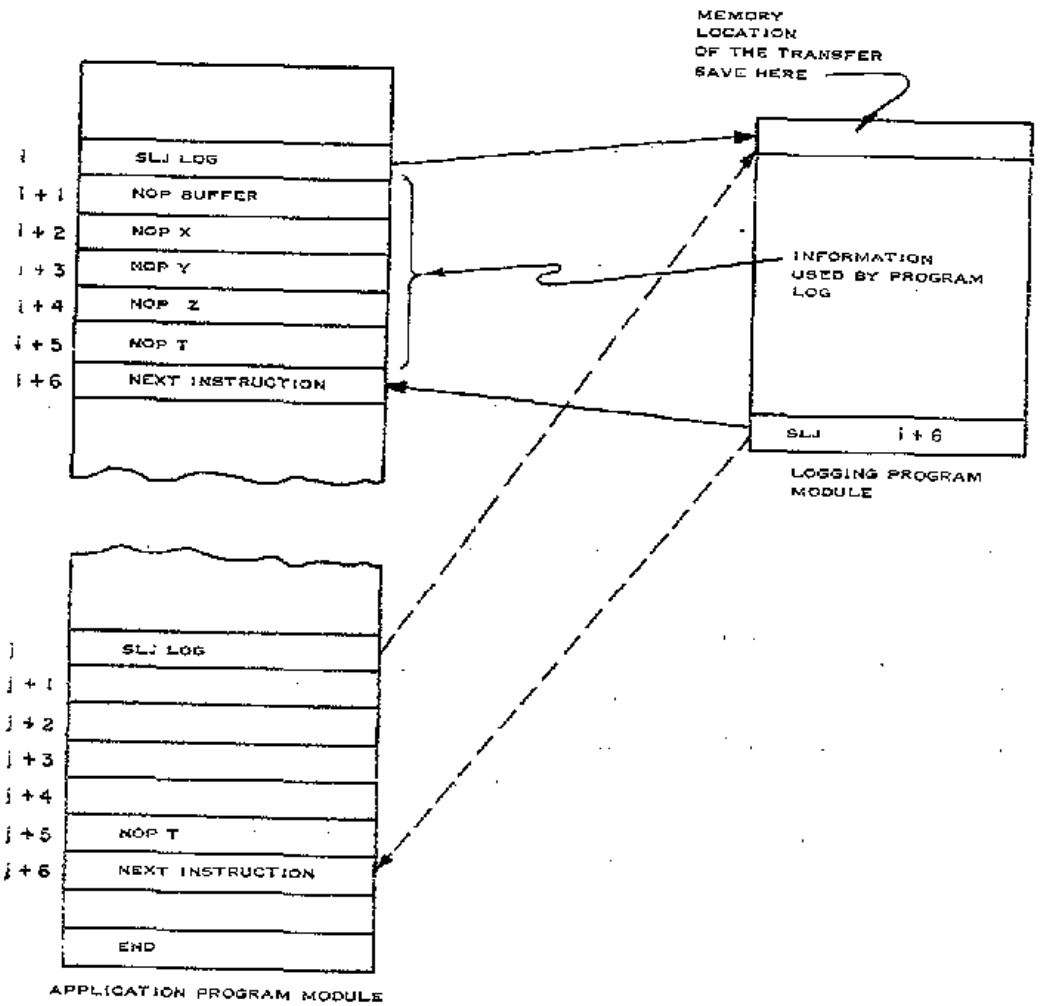


Figure 48. In-Core Transfer to Program Module—Log For Data Recording

Data can be double buffered in the core memory as a measure to increase data recording speed. This scheme provides two buffer areas in core memory for the data reduction routines to store the required data before outputting it to the recording media. The recording routine may be storing data in buffer area # 1 while the data from buffer area # 2 is being outputted to magnetic tape (or disc). Then the process is reversed, i. e., data is stored in area # 2 while area # 1 is being outputted.

Storing all data in common output buffer areas is another method to speed up logging. Each data point can be tagged with a logging identify code for processing by the analysis programs. This eliminates the need for complex logic to sort the data into separate data buffering areas and separate outputting routines.

POST-RUN DATA REDUCTION AND ANALYSIS

The analysis programs, which can be designed to run directly after the training or simulation session, can also be designed as an off-line executive and a set of modular analysis routines. Then the operational application programs do not have to reside in core memory--the entire analysis software can occupy core memory. Various complex data reductions also can be implemented as timing is not restrained by the applications software.

The analysis and reduction routines should be run directly after the simulation or training session. The availability of all the data on a quick-look basis allows the principal investigations to evaluate a session while qualitative evaluation is fresh in their minds. Frequently this data has a direct bearing on the setting of parameters for the next run. Discussion of training performance can be held with trainees immediately after the session if the data is available. This type of evaluation is more effective than an evaluation of the data when it is processed in a routine over-night manner.

Normal procedures are to process all the data as a routine overnight data processing job, so a huge amount of paper is generated. This often creates a problem as the instructor or evaluation personnel must "wade" through masses of paper to find specific data for man/machine evaluation.

The ability of the instructor, human factors engineers, and other pertinent personnel to dynamically select those analyses or reduction routines necessary to give only the required data is important. These methods can be implemented as follows:

1. A selection data card can be used at the end of operation phase of training or simulation to select the appropriate data analysis and reduction routines all at once.

2. The data can be presented to the personnel in real-time and allow them to select additional routines as needed in a sequential manner.

COMPUTER DRIVEN SIMULATION ENVIRONMENT

The CDSE provided real-time simulation studies for an Air Traffic Control Terminal Area Research Project. The objective of these simulations was to evaluate the general use of digital computer techniques as an aid to air traffic management processes under advanced terminal area airplane sequencing and control concepts. CDSE application was thoroughly tested and is considered to be a significant advancement in the art of aircraft simulation.[1] The computer's rate in digital recording, while performing the necessary simulation function and preserving program system integrity, contributed to the project's success. Dynamic selection of data reduction and evaluation programs greatly aided human factors analysis of simulation runs.

A digital computer was used in real-time simulations to:

1. Study computer aided systems.
2. Generate aircraft targets.
3. Simulate subsystems such as radars, trackers, or beacon transponders.
4. Allow flexible programmable situation displays.
5. Allow rapid changes from problem to problem.
6. Collect data and analyze while performing the above tasks.

The IBM 7090 computer at the computation center was equipped with two channels, a direct data connector, and a real-time clock. The direct data connection provided high-speed data transfer to the Display Buffer System and from the Keyboard Data Entry System (Figure 49).

The chosen displays included the Charactron which can display alphabetic and numeric information, special symbols (for aircraft fixes) as well as lines. Since the display format was under program control, flexibility existed on these displays for any desired situation (Figure 50). A set of input keyboards was required to allow air traffic controllers and simulator pilots to communicate with the computer. Six keyboards were used for simulator-pilot input functions and three for air traffic controller positions (Figure 51).

Four voice communication channels are provided between the controller and the pilots. Each channel was monitored via the IBM 7090 sense lines in order to record data for computing the distribution of message lengths and communication delays.

A typical simulation consisted of three main program modules arranged in a chain of three links. The purpose of Link 1 was to read some 94 parameters into the computer, select input traffic and initialize the data recording routines. After completion of Link 1, the real-time operational simulation program, Link 2, was called. The initialization program (Link 1) ran for less than one minute compared to two hours or more for a simulation run (Link 2).

After the simulation run, the final data was recorded onto magnetic tape. Link 3 was called and read into the memory over Link 2. Link 3 consisted of some 52 different analysis programs to process the data collected during Link 2 and stored on magnetic tape.

Figure 52 shows a functional diagram of the CDSE. The 7090 program consisted of four functional areas: simulated aircraft targets, data acquisition (e. g., radar), operational control systems, and data recording. Some of the parameters read into the computer in Link 1 are shown in the figure. For example, the estimated aircraft characteristics, geometry of the area simulated, and the forecase wind were used in the operational control system logic. The external displays and keyboard provided command and control hardware to integrate the air traffic controllers into the operational control system. The data recording programs were able to monitor communication line hardware, simulated aircraft keyboard workload, and control system performance.

The on-line data recording program module consisted of seven subroutines (Figure 53). The data recording executive initialized all on-line data recording subroutines at the beginning of the run. The executive also was periodically tested during the run to determine if each of the other data collection subroutines should be called. The executive was called once each second.

The communication data subrountine wrote either one of two blocks of computer memory containing communication data. Two storage areas were used to allow communications data

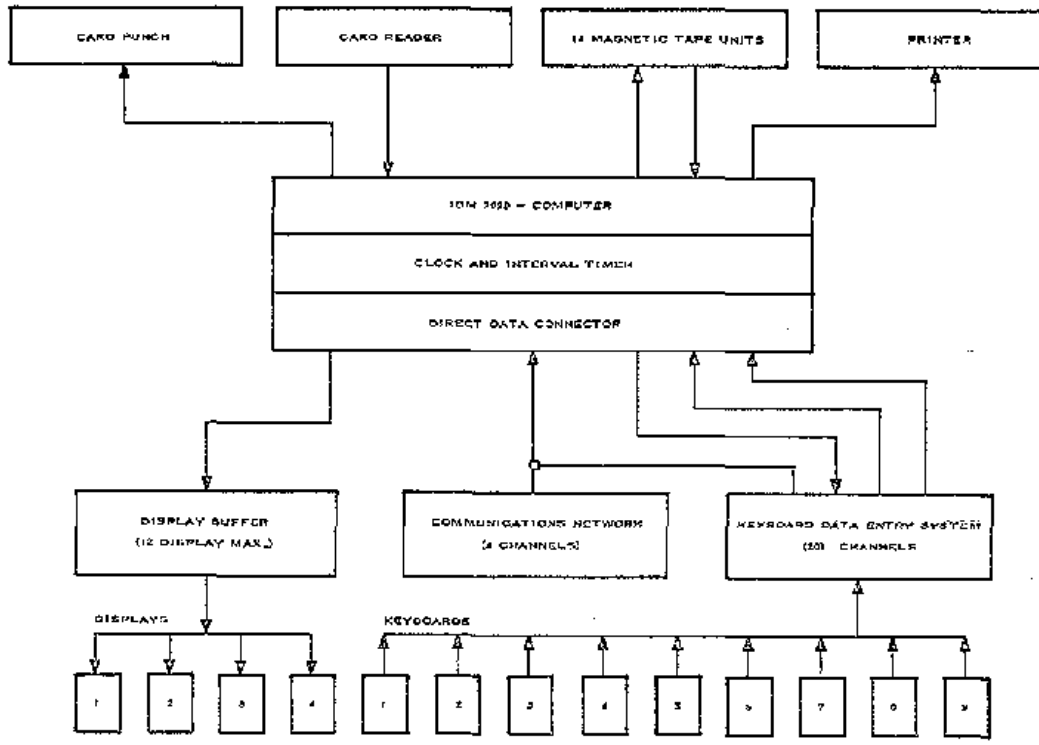


Figure 49. IBM 7090 Computer Equipment Configuration

Figure 50. Approach Controller's Display Control

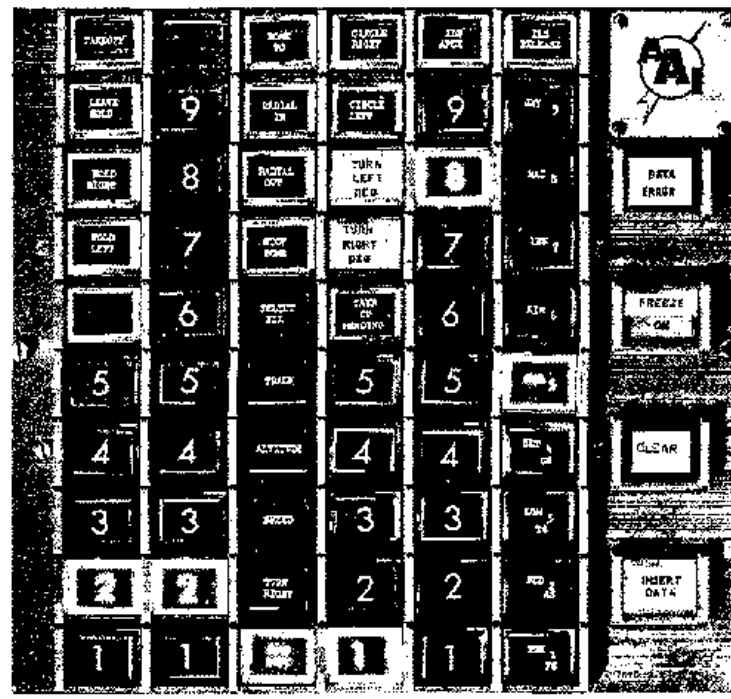


Figure 51. Simulator Pilot Keyboard

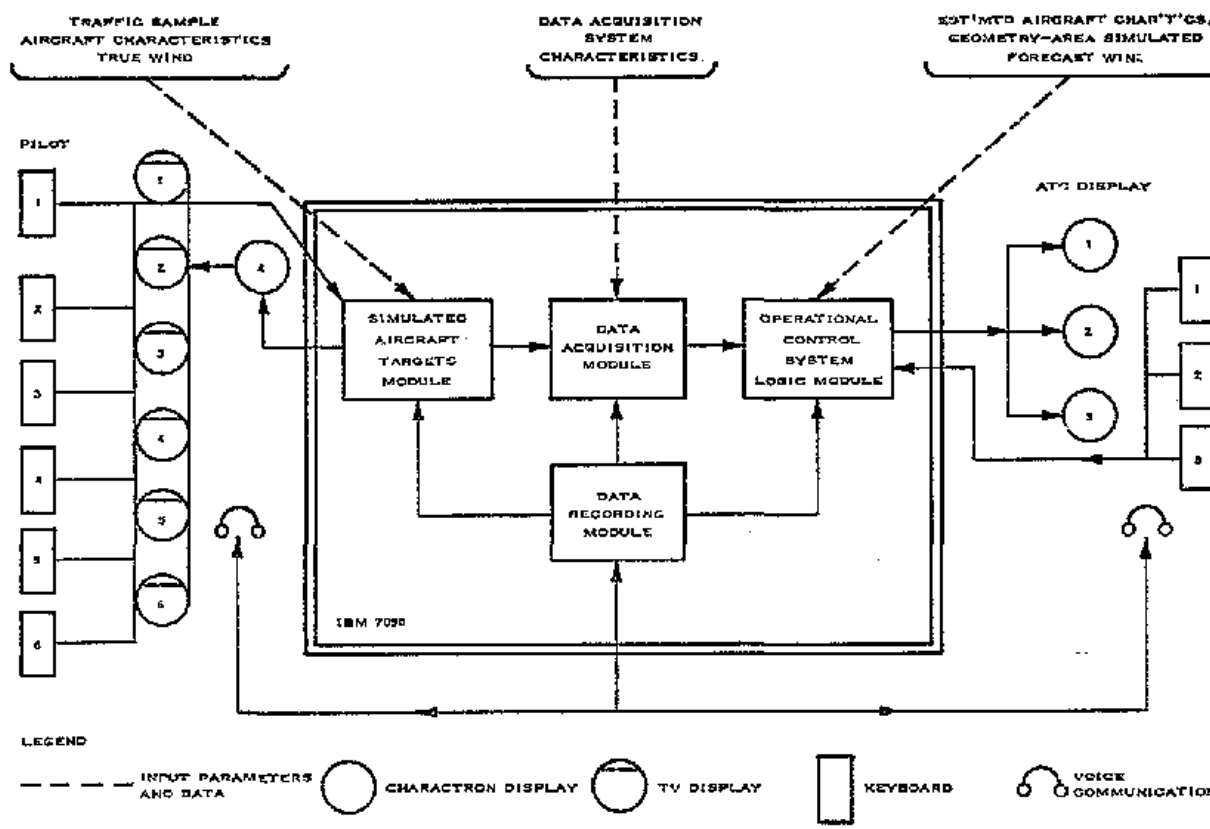


Figure 52. Functional Diagram of the Computer-Driven Simulation Environment (CDSE)

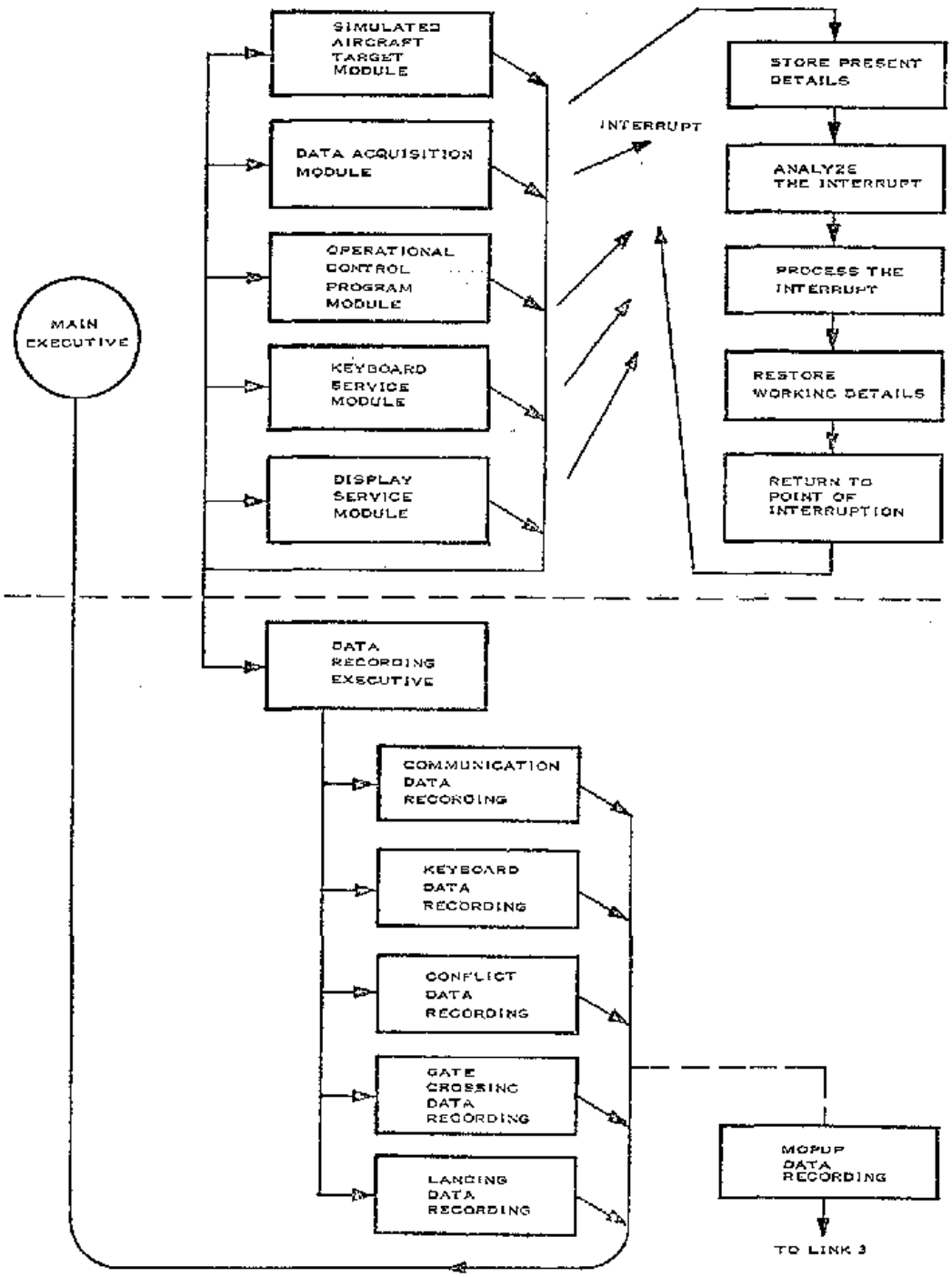


Figure 53. Relationship of Data Recording Programs to Other Program Functions

to be collected in one area while the other is being written on magnetic tape. Each time one of the areas was filled, a switch was set by the subroutine which indicates to the executive that a block of data can be written on tape.

The keyboard data recording subroutine stored the number keyboard messages of each of the ten keyboards periodically during the preceding minute. The tape writing and buffering scheme was similar to communication data subroutines.

The conflict recording subroutine checked each ten seconds to determine if a conflict existed between any pair of aircraft. A conflict situation occurred between a pair of aircraft whenever both the minimum vertical and horizontal separation criteria were violated. Data was double buffered and written on tape as the buffers were filled.

The gate and landing subroutine detected when aircraft cross the Instrument Landing System (ILS) gate. Pertinent data was recorded in memory for each aircraft which crosses the gate and lands. This data was recorded in memory during the simulation. On completion of the simulation run, this data was written on tape by the "mop-up" data recording routine. In addition, this routine wrote all the partially filled data buffer areas on magnetic tape to complete the data set for the run.

The off-line analysis program consisted of a sequence of 26 program switches which were used to select which particular subroutines were to be used in an analysis run. After passing through a sequence of these 26 program switches, the option was available to loop through these switches once again with different value settings for each of the switches. At each switch point three options were available, that is to call subroutine A, call subroutine B, or go on to the next switch point. Possible values any switch may have were (-), (0), and (+). A zero value for a switch caused the executive program to pass to the next switch point. A negative value causes subroutine A to be called while a plus value causes control to pass to subroutine B.

Figure 54 shows the organization of this chain of logic. Each block represents a subroutine of the Analysis Processor executive. Therefore, 26 possible subroutine options could be used with this program module. However, not all of these blocks were defined. Therefore, there was the ability to add other subroutines to the Analysis Processor to handle other types of data reduction and analysis not evident before. Some of these subroutines in the blocks use other subroutines to do certain statistical functions. These supplementary subroutines were of a statistical type and were written in FORTRAN compiler language which was understood.

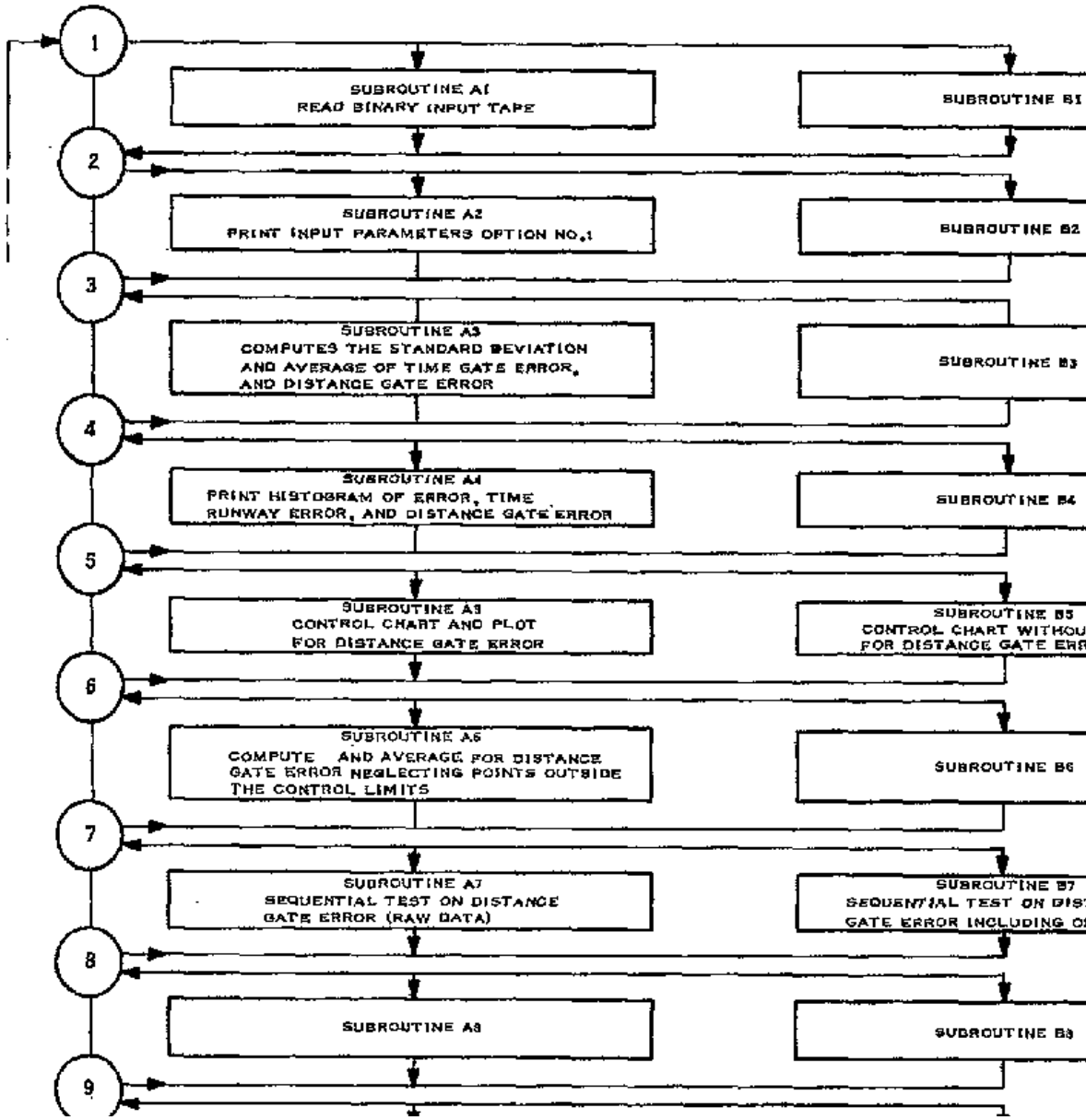
Therefore, the only input to the analysis control program was a sequence of 26 one digit integers having the values of either -1, 0, +1 and the on-line data tapes. These values plus a value for the number of times to loop through the Analysis Processor were the only inputs to the executive program proper. These parameters determine which switch points are selected; thus, selecting the various subroutines for the processing job and determining how many times the analysis loop should be traversed to analyze the data.

The analysis program module ran directly after each simulation session. This data was available to all personnel 15 to 30 minutes after the simulation. Figure 55 shows a typical output of subroutine A25 including supplementary plotting subroutine used to print the histogram.

TACFIRE TRAINING SUPPORT SYSTEM (TSS)

The design objective of the TSS is to provide TACFIRE operating personnel with a simulated real-time environment for field training and evaluation purposes. The TSS recording and analysis programs were designed to give TACFIRE operating personnel a simulated real-time environment for Army field training evaluations. TACFIRE is a military system

BLANK PAGE


1
SUBROUTINE A1
READ BINARY INPUT TAPE
SUBROUTINE B1
2
SUBROUTINE A2
PRINT INPUT PARAMETERS OPTION NO.1
SUBROUTINE B2
3
SUBROUTINE A3
COMPUTES THE STANDARD DEVIATION
AND AVERAGE OF TIME GATE ERROR,
AND DISTANCE GATE ERROR
SUBROUTINE B3
4
SUBROUTINE A4
PRINT HISTOGRAM OF ERROR, TIME
RUNWAY ERROR, AND DISTANCE GATE ERROR
SUBROUTINE B4
5
SUBROUTINE A5
CONTROL CHART AND PLOT
FOR DISTANCE GATE ERROR
SUBROUTINE B5
CONTROL CHART WITHOU
FOR DISTANCE GATE ERR
6
SUBROUTINE A6
COMPUTE AND AVERAGE FOR DISTANCE
GATE ERROR NEGLECTING POINTS OUTSIDE
THE CONTROL LIMITS
SUBROUTINE B6
7
SUBROUTINE A7
SEQUENTIAL TEST ON DISTANCE
GATE ERROR (RAW DATA)
SUBROUTINE B7
SEQUENTIAL TEST ON DIS
GATE ERROR INCLUDING O
8
SUBROUTINE A8
SUBROUTINE B8
9

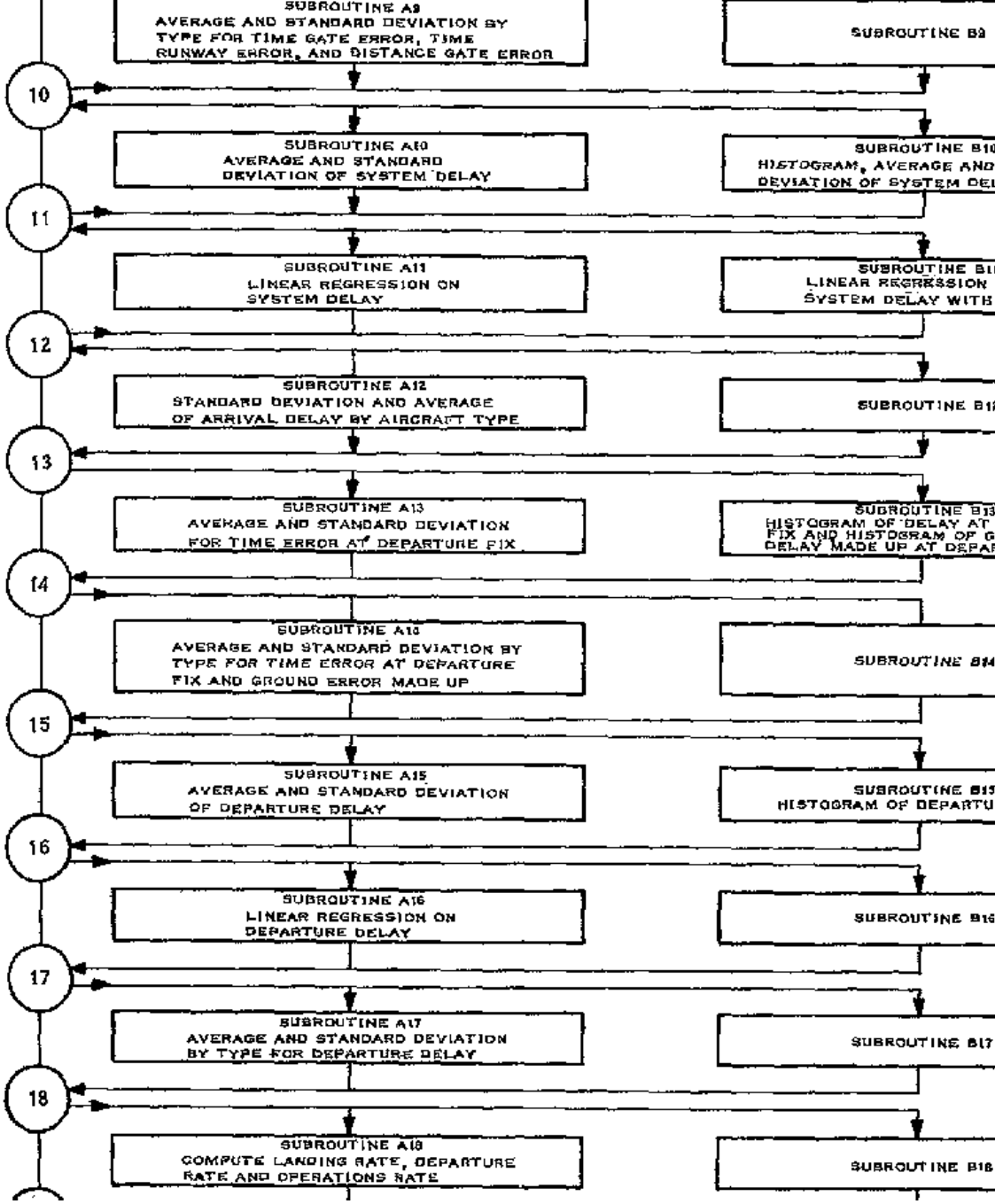

SUBROUTINE A9
AVERAGE AND STANDARD DEVIATION BY TYPE FOR TIME GATE ERROR, TIME RUNWAY ERROR, AND DISTANCE GATE ERROR
SUBROUTINE B9
10
SUBROUTINE A10
AVERAGE AND STANDARD DEVIATION OF SYSTEM DELAY
SUBROUTINE B10
HISTOGRAM, AVERAGE AND
DEVIATION OF SYSTEM DEL
11
SUBROUTINE A11
LINEAR REGRESSION ON SYSTEM DELAY
SUBROUTINE B1
LINEAR REGRESSION
SYSTEM DELAY WITH
12
SUBROUTINE A12
STANDARD DEVIATION AND AVERAGE OF ARRIVAL DELAY BY AIRCRAFT TYPE
SUBROUTINE B12
13
SUBROUTINE A13
AVERAGE AND STANDARD DEVIATION FOR TIME ERROR AT DEPARTURE FIX
SUBROUTINE B13
HISTOGRAM OF DELAY AT
FIX AND HISTOGRAM OF G
DELAY MADE UP AT DEPAR
14
SUBROUTINE A14
AVERAGE AND STANDARD DEVIATION BY TYPE FOR TIME ERROR AT DEPARTURE FIX AND GROUND ERROR MADE UP
SUBROUTINE B14
15
SUBROUTINE A15
AVERAGE AND STANDARD DEVIATION OF DEPARTURE DELAY
SUBROUTINE B15
HISTOGRAM OF DEPARTU
16
SUBROUTINE A16
LINEAR REGRESSION ON DEPARTURE DELAY
SUBROUTINE B16
17
SUBROUTINE A17
AVERAGE AND STANDARD DEVIATION BY TYPE FOR DEPARTURE DELAY
SUBROUTINE B17
18
SUBROUTINE A18
COMPUTE LANDING RATE, DEPARTURE RATE AND OPERATIONS RATE
SUBROUTINE B18

Figure 54. Off-Line System Control Real Time Analysis Processor

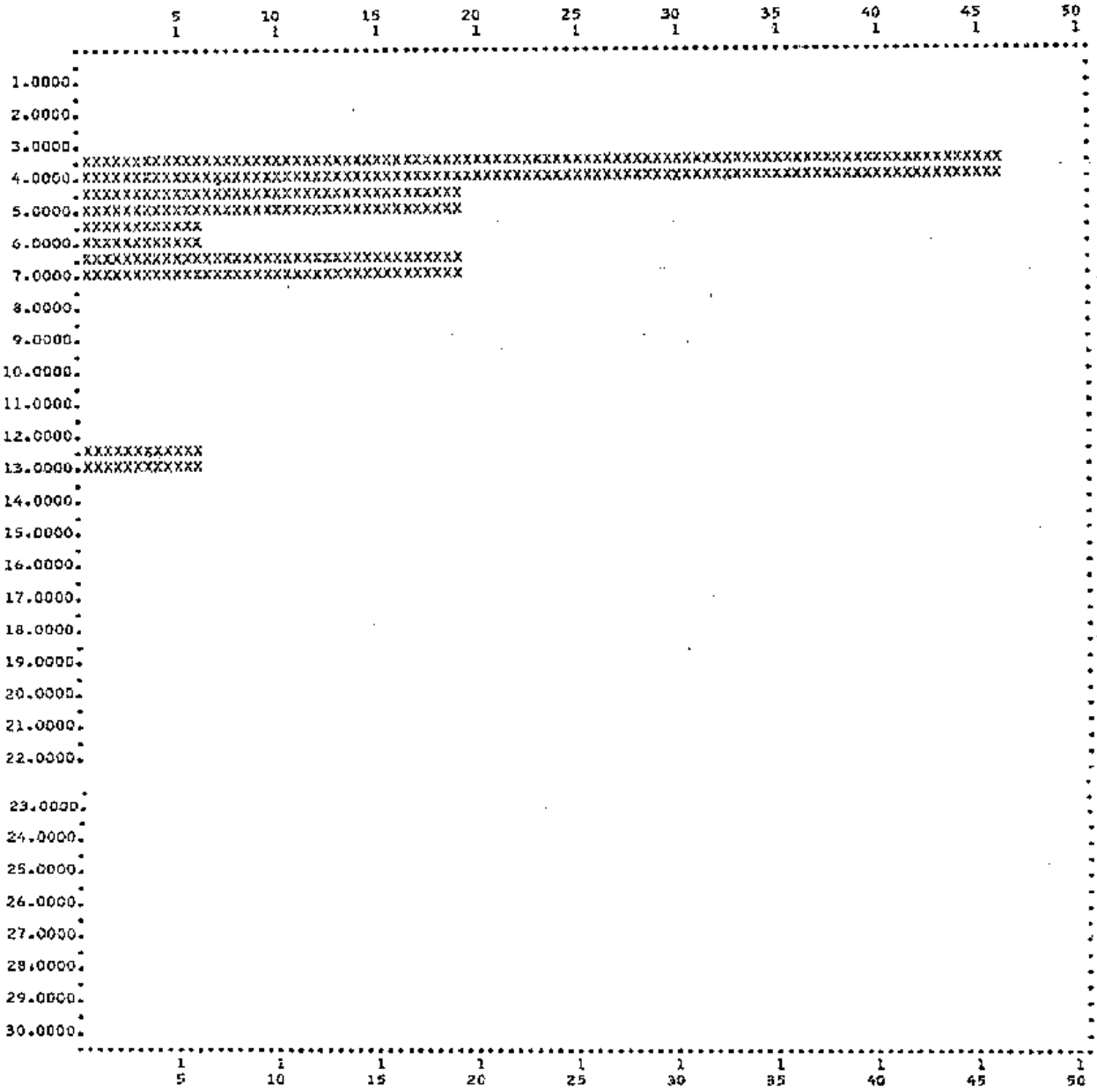


Figure 55. Outputs of Subroutine A25

of highly reliable operating equipment configured in tactically mobile vehicles. Each system contains two digital computers. The major design consideration of the TSS is preservation of operating system integrity. Digital recording and analysis programs were designed under this constraint.

Supervisory control over the simulated environment is designed so a training instructor may monitor, supervise, and direct dynamic closed-loop control over training of field personnel. Concepts of computer-aided instruction (CAI) will be used in interfacing with the TACFIRE operating system so the instructor may exercise maximum control over each training mission.

Measures of personnel performance will be obtained for controlled mixes of message traffic subject to timely instructor evaluation. While a training mission may run four hours, accumulated mission time can represent any amount of simulated real-time operation. The instructor, by interfacing CAI routines to the TACFIRE Operating System, will control the timing of message arrival, departure and interactions. He may replay selected messages or introduce new message combinations through manipulation of the system pseudo-real-time clocks. He may alter existing message format or context for fault insertion into the system. Through inclusion of the above facilities, the instructor is given a quantitative tool which can measure trainee response to simulated real-time operating conditions.

The training operating system will require three additional pieces of equipment to make the TACFIRE training system operational:

1. An additional disc storage unit for field training input data and data recording.
2. An instructor's display console.
3. Two 8,000-word core memory modules.

The added core memory will be used to incorporate the TSS operating system software (i.e., CAI and simulated input control services) so the TACFIRE operating system software will remain intact (Figure 56).

The training operating system is structured to operate in two function modes of operation:

1. Team training state for a TACFIRE vehicle.
2. Operator training state for display console.

Within each state, TSS operation will encompass three separate phases (Figure 57):

1. Pre-training initiation.
2. Training-run operation.
3. Post-training analysis.

The TSS software supporting this operation will be divided among two different computing systems. Input message traffic and simulated display output traffic will be generated on an off-line data processing system which will generate data to be stored in the training disc storage device. Disc cartridges then may be shipped to the TSS training library as representative traffic samples to run on the TSS hardware.

TSS software initializing the TSS installation will operate on the TSS tactical processor hardware and use this prepared disc cartridge. On instructor command, the TSS Operating System will commence the training-run execution. The instructor shall select the team or

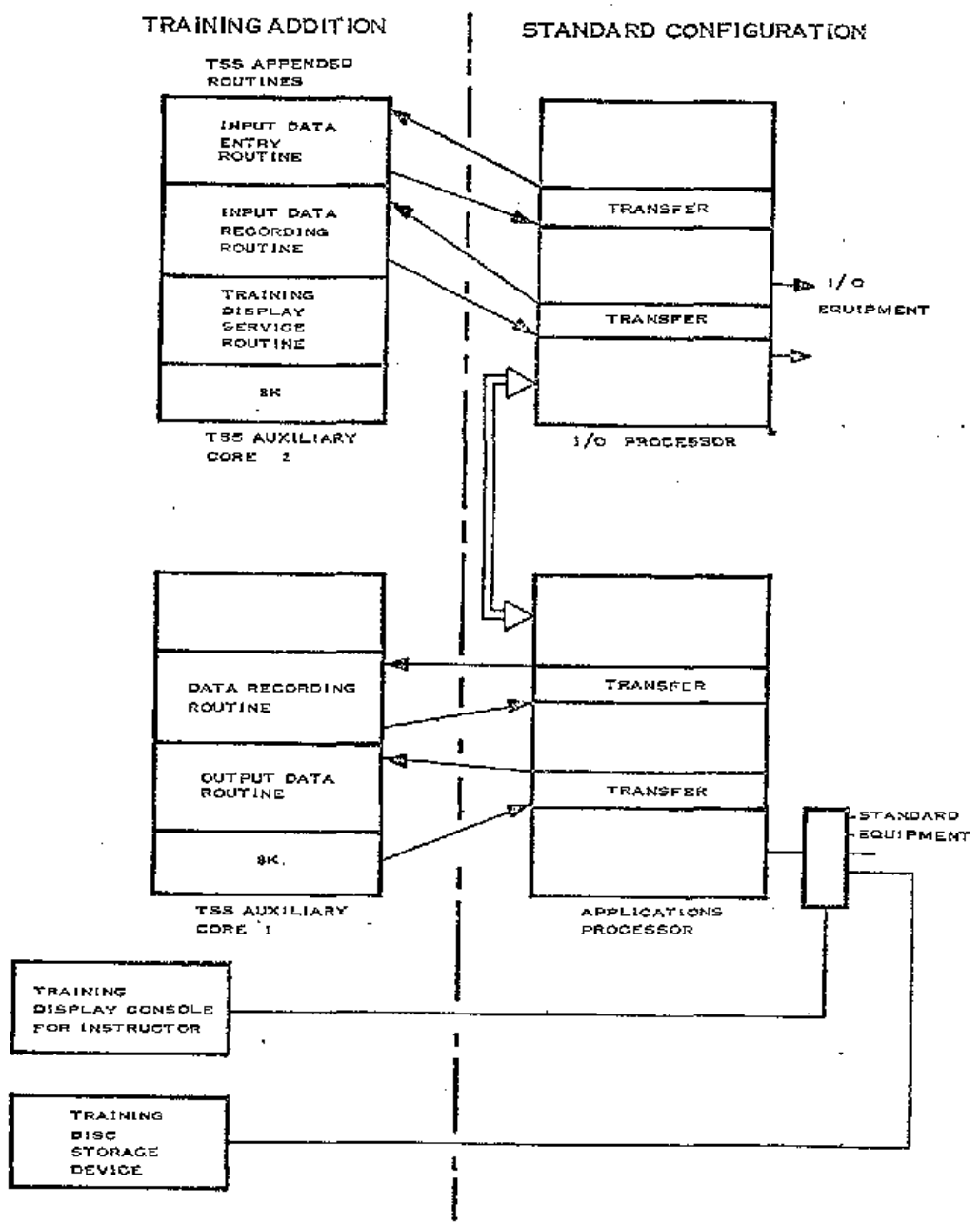


Figure 56. Tacfire Training System

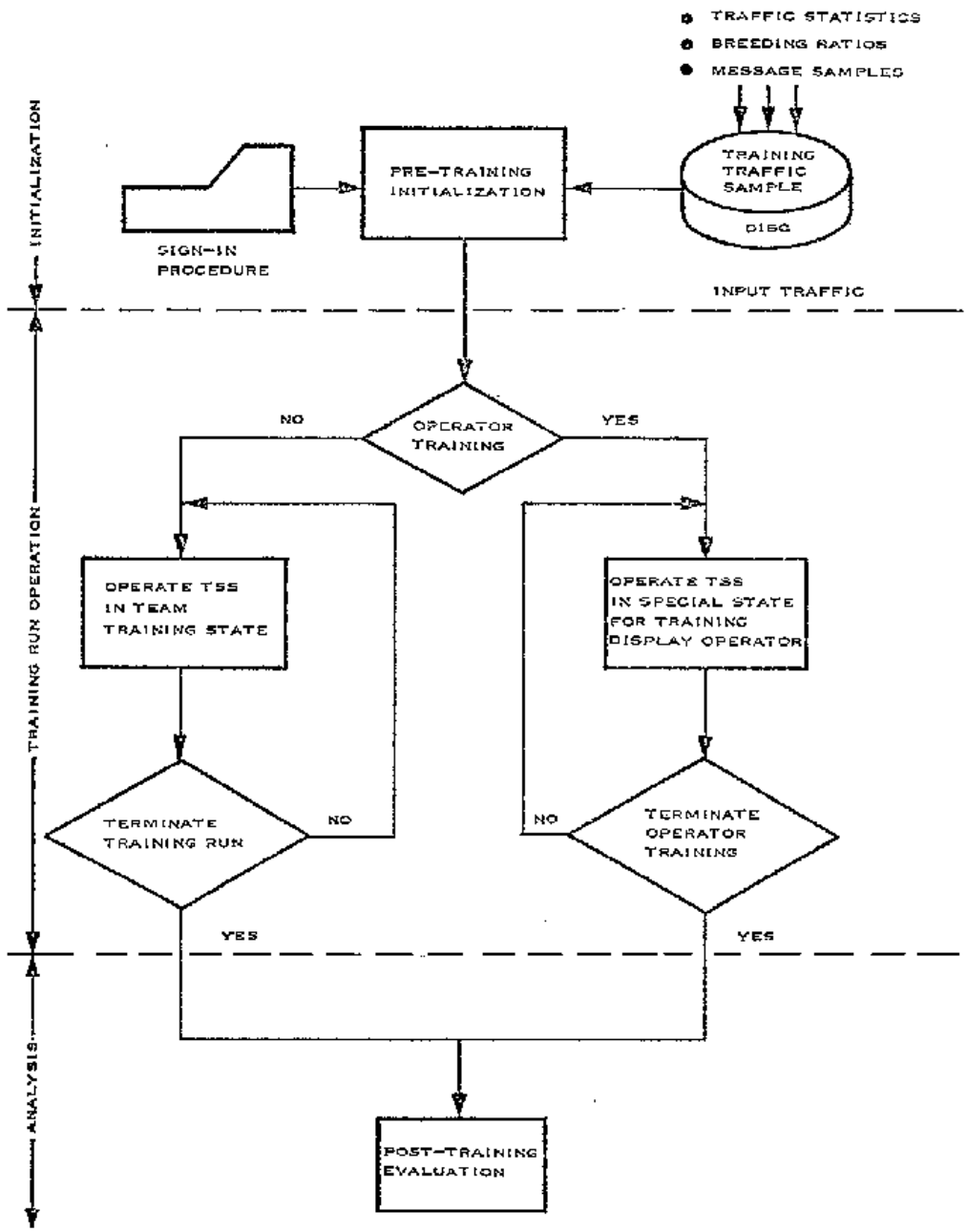


Figure 57. Modes of TSS Operation

operator training state. Data will be recorded on this same disc cartridge for post-run analysis.

Finally, after completing a training mission, the instructor can terminate the training run and enter the data reduction and evaluation analysis phase.

Software design will provide the necessary simulation support routines required for implementation of the fieldable TSS Operating System. Software routines will include the following functional programs:

1. Load Generation—This service routine will generate the simulated message traffic for use on the tactical processor.

2. Simulated Input Control—Includes the various routines used to effect the routing of simulated message traffice within the TSS TACFIRE Operating System. These routines will include:

A. I/O handling.

B. Time management.

C. Simulated I/O routing.

D. Logging.

E. Data recording.

3. CAI/Instructor Console Control—The CAI interface to the simulated input control services and the TACFIRE Operating System is effected from this service routine.

4. Post-Training Evaluator—The post-training evaluator service will enable the instructor to selectively evaluate and analyze all data recorded during the training mission. At least 20 different evaluator routines may be invoked to post process message traffic response information collected during each training run. Post-training evaluation will include the following recap services:

A. Print set-up and sign-in parameters.

B. Print input and output messages.

C. Analyze and plot profiles of mission runs.

D. Analyze and plot input and output.

E. Compute and plot human reaction to output messages.

F. Correlate input and output messages by source and device.

G. Plot scatter diagram of input vs output messages.

H. Perform two-way analysis of variance of run statistics.

The two systems described have been designed with data recording and analysis considerations as an integral part of the software. Considerable thought should be given to what should be recorded and what analysis methods will be used. Many statistical routines already may be available for the simulation computer, thus reducing some of the analysis programming costs. Sufficient data should be recorded so additional analysis tasks can be later added to

the post-run reduction and analysis program. The usefulness of data after a simulation or training session is inversely proportional to the time it is available for evaluation by the experimentors. This gives further reason to run the analysis of the data directly on completion of the simulation as an integral part of the simulation run. Control of the analysis program should be given to the experimentors so they may select easily the minimum set of analysis and data reduction programs for that specific simulation or training run.

Reference:

1. "A Computer Driven Simulation Environment for Air Traffic Control Studies" Proceedings Fall Joint Computer Conference, 1963.