NASA CONTRACTOR REPORT

NASA CR-1867

NASA CR-1867

SPACEBORNE COMPUTER EXECUTIVE ROUTINE FUNCTIONAL DESIGN SPECIFICATION

Volume I. Functional Design of a Flight Computer Executive Program for the Reusable Shuttle

by R. T. Curran

Prepared by
COMPUTER SCIENCES CORPORATION
FIELD SERVICES DIVISION, AEROSPACE SYSTEMS CENTER
Huntsville, Ala. 35802
for George C. Marshall Space Flight Center

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION • WASHINGTON, D. C. • OCTOBER 1971

TECHNICAL REPORT STANDARD TITLE PAGE

1. REPORT NO.	2. GOVERNMENT ACCESSION NO.	3. RECIPIENT'S CATALOG NO.
NASA CR-1867		

4. TITLE AND SUBTITLE	5. REPORT DATE
Spaceborne Computer Executive Routine Functional Design Specification, Volume I. Functional Design of a Flight Computer Executive Program for the Reusable Shuttle	October 1971
	6. PERFORMING ORGANIZATION CODE
7. AUTHOR(S) R. T. Curran	8. PERFORMING ORGANIZATION REPORT #
9. PERFORMING ORGANIZATION NAME AND ADDRESS Computer Sciences Corporation, Field Services Division, Aerospace Systems Center, 8300 South Whitesburg Drive, Huntsville, Alabama 35802	10. WORK UNIT NO.
	11. CONTRACT OR GRANT NO. NAS8-24930
12. SPONSORING AGENCY NAME AND ADDRESS National Aeronautics and Space Administration, Washington, D. C. 20546	13. TYPE OF REPORT & PERIOD COVERED Contractor Final Report
	14. SPONSORING AGENCY CODE

15. SUPPLEMENTARY NOTES

16. ABSTRACT

This report presents a flight computer functional executive design for the Reusable Shuttle. The design is given in the form of functional flowcharts and prose description. Techniques utilized in the regulation of process flow to accomplish activation, resource allocation, suspension, termination, and error masking based on process primitives are considered. Preliminary estimates of main storage utilization by the Executive are furnished. Conclusions and recommendations for timely, effective software-hardware integration in the Reusable Shuttle avionics system are proposed.

This document is Volume I of a three-volume report entitled "Spaceborne Computer Executive Routine Functional Design Specification." The other volumes are:

Volume II: Executive Design for Space Station/Base

Volume III: Executive Routine Primitives and Process Control

17. KEY WORDS	18. DISTRIBUTION STATEMENT
Reusable Shuttle Executive Program Avionics Real-Time Monitor Onboard Computer Data Management Systems Information Management Systems	Unclassified - Unlimited

19. SECURITY CLASSIF. (of this report)	20. SECURITY CLASSIF. (of this page)	21. NO. OF PAGES	22. PRICE
Unclassified	Unclassified	177	$3.00

*For sale by the National Technical Information Service, Springfield, Virginia 22151

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (Continued)

LIST OF TABLES

DEFINITION OF SYMBOLS

Symbol	Definition
Active List	List of processes that are time-dependent. Each process will be transferred to the Ready List when the Execution Delta T is decremented to zero by the Clock Update routine.
CPU	Central Processing Unit
DBM	Data Bus Monitor
Execution Delta T	Part of process data in the Active List, representing incremental time until execution (countdown).
FIFO	First In, First Out
I/O	Input/Output
IOCU	Input/Output Control Unit
K	1000
KB	Keyboard
LRU	Line Replacement Unit
MDAC	McDonnell-Douglas Astronautics Company
PCB	Process Control Block
PD	Powered Down
Process Control Block	List of information required by the Executive in order to maintain priority performance of processes. Data includes register storage, process priority level, beginning location for execution and allied information.
Process ID	Identifier part of process data in either Active or Ready Lists. Data contains priority assignment and pointer to the Process Control Block.
Ready List	List of processes that have been waked and are to be executed by the Process Dispatcher on a priority basis.

DEFINITION OF SYMBOLS (Continued)

Symbol	Definition
RVC	Running Voting Controlling
$RV\bar{C}$	Running Voting Not Controlling
SCU	System Control Unit
ST	Self-Test
Standby	Power on but not executing code
SVC	Supervisor Call
TLIC	Timeline Interpreter/Controller

FOREWORD

The work reported herein was administered in the Systems Research Branch, Computer Systems Division, Computation Laboratory, MSFC, with Bobby C. Hodges assigned as Contracting Officer's Representative. The writer would like to thank Mr. Hodges for his helpful suggestions on the content and emphasis of the material covered in this report.

SUMMARY

This report comprises the generic functional design for the Space Shuttle flight computer system executive. The executive provides the basis for controlling all process flow within the flight computer system.

The executive design concept is based upon an analysis of known program requirements, similar space projects, and experience in the design of comparable computer software. Assumptions are made and parameters are established to identify critical decisions and the impact of those decisions on the computer system and the executive.

The generic functional design is oriented specifically to provide a basis for subsequent modeling and simulation. The executive, as defined, may be considered a baseline programming system to complement the Space Shuttle Phase B Hardware Definition.

The functional design identifies crucial elements of the executive system as:

- Process Activation
- Device Allocation
- Main Storage Allocation
- Central Processing Unit Allocation
- Process Suspension and Termination

The functional executive design specifies a control structure for scheduling the Space Shuttle application processes according to a wide range of execution priorities and timing requirements. Necessity for modification of system applications processes during two-week turnaround period was a primary factor in each design criterion.

Verification of hardware and software design requirements as delineated by this report should proceed in parallel with subsequent simulation and evaluation to minimize development costs and reduce implementation cycle.

An extensible version of the FORTRAN language with carry-through capability is recommended as a baseline implementation language for the flight Executive and the application programs.

Detailed man-machine interface requirements must be further developed with emphasis on critical mission phases. Further study is essential on all phases of the Avionics system to:

- Determine the efficacy of the Executive in meeting Space Shuttle requirements,
- Develop detailed software requirements,
- Verify integrated hardware/software interfaces,
- Identify hardware/software tradeoffs in system configuration, and
- Ascertain actual computer loading parameters.

SECTION I. INTRODUCTION

This report is submitted in compliance with requirements of NASA Contract Number NAS8-24930 for a final report on the Functional Design of a Flight Computer Executive Program for the Reusable Shuttle.

The problem of evaluating radically different approaches to the design of a complex computer executive system has, by no means, a straightforward solution. The generic functional design of the Space Shuttle flight computer executive, outlined in this report, is oriented specifically to provide a basis for subsequent modeling and evaluation by simulation. In addition, the flight computer executive system, as defined herein, can be considered a baseline programming system to complement the Phase B hardware definition. The executive also establishes reference for the specification and design of applications processes, such as Navigation and Guidance, helps in estimating main storage requirements, and sharpens requirements for hardware/software trade studies.

The functional design identifies crucial elements of the executive system. The operation and interaction of these elements are discussed utilizing prose description and flow diagrams. The discussion is presented from the point of view of the regulation of the flow of control through the processes within the system, assumed to be in a steady-state operation. New process activation, brought about by start signals (possibly interrupts) or based on critical time dependency, is analyzed. Regulation of the demand on system resources by an executing process is scrutinized. Completion of the process task implies that the executive functions associated with rescheduling or termination must be invoked. In contrast to the successful run to completion of a process, faults or error conditions interrupt the application process to cause execution of failure-masking processes. A failure is masked by reconfiguring the system. This reconfiguration is effected so that either the failed device is replaced by an intact Line Replacement Unit, or a graceful degradation strategy is adopted. In the graceful degradation mode, the device is removed from the active system, but no replacement is available.

The functional executive design specifies a control structure for scheduling the Space Shuttle application processes according to a wide range of execution priorities and timing requirements.

The report comprises the following:

- Design Overview/Organization
- Process Flow Regulation

- Failure Response Tactics
- Configuration Control
- Inter-Subsystem Information Flow
- Executive Structures
- Preliminary Estimates of Main Storage Utilization

Procedure Descriptions, functional attribute descriptions, and flowcharts are provided in the appendices.

SECTION II. DESIGN OVERVIEW/ORGANIZATION

The computer configuration used as a design basis for the executive system is the centralized configuration proposed by McDonnell-Douglas Astronautics Company (MDAC) (1). This configuration consists of four computers with associated Input/Output Bus Control Units (IOCU), two mass memories, two maintenance recorders, and a redundant System Control Unit (SCU). (See figure 1.) Each computer communicates with any of four data buses through its IOCU. The SCU communicates with the computers via hardwires rather than the data buses, and is responsible for computer fault isolation and reconfiguration. Communication with the mass storage devices and maintenance recorders is via the data buses.

In addition to the central computers discussed above there are display processors and engine computers (two computers/engine).

The central computer executive, however, is chosen for elucidation as representing the most complex programming system present. In addition to process flow regulation within the central computer system, the central executive is responsible for scheduling processes in the display processors and in the engine computers. (See figure 2.)

In contrast, the display processor executives are special-purpose in nature and can be a subset of the central computer executive (depending, however, on hardware/software trade study results).

The SCU parallels the engine computers in that it has responsibility for a very particular assignment; i.e., the masking of central computer faults by reconfiguration. The SCU apparently (depending, however, on current trade study results) will be highly hardware-implemented and require merely a skeleton software executive. Procedures to accomplish reconfiguration by either hardware or software implemented processes are discussed.

---

[1] McDonnell-Douglas Astronautics Company: Space Shuttle Phase B Systems Study, March 1971

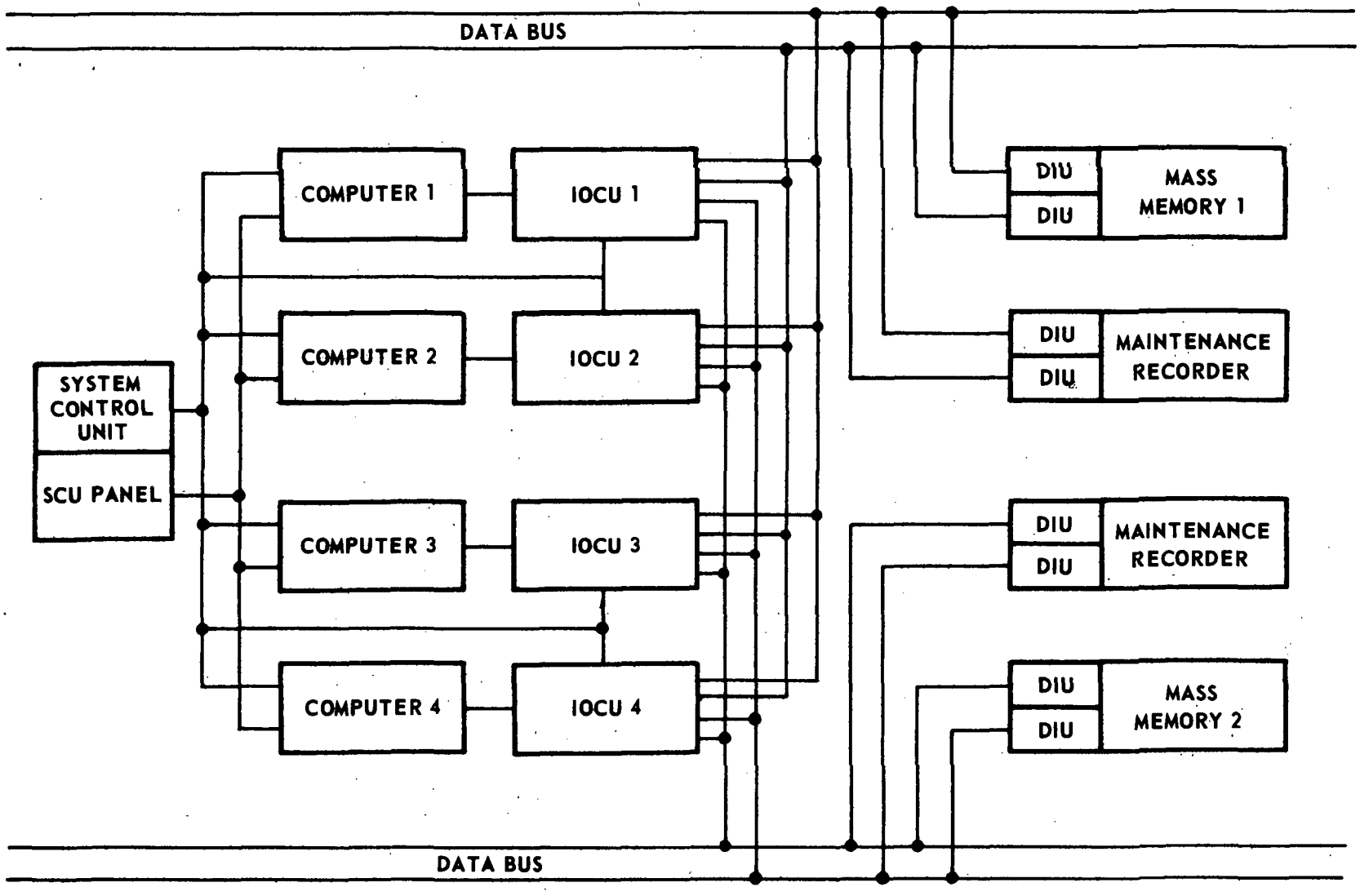


FIGURE 1. DMS BLOCK DIAGRAM FOR CENTRALIZED SYSTEM

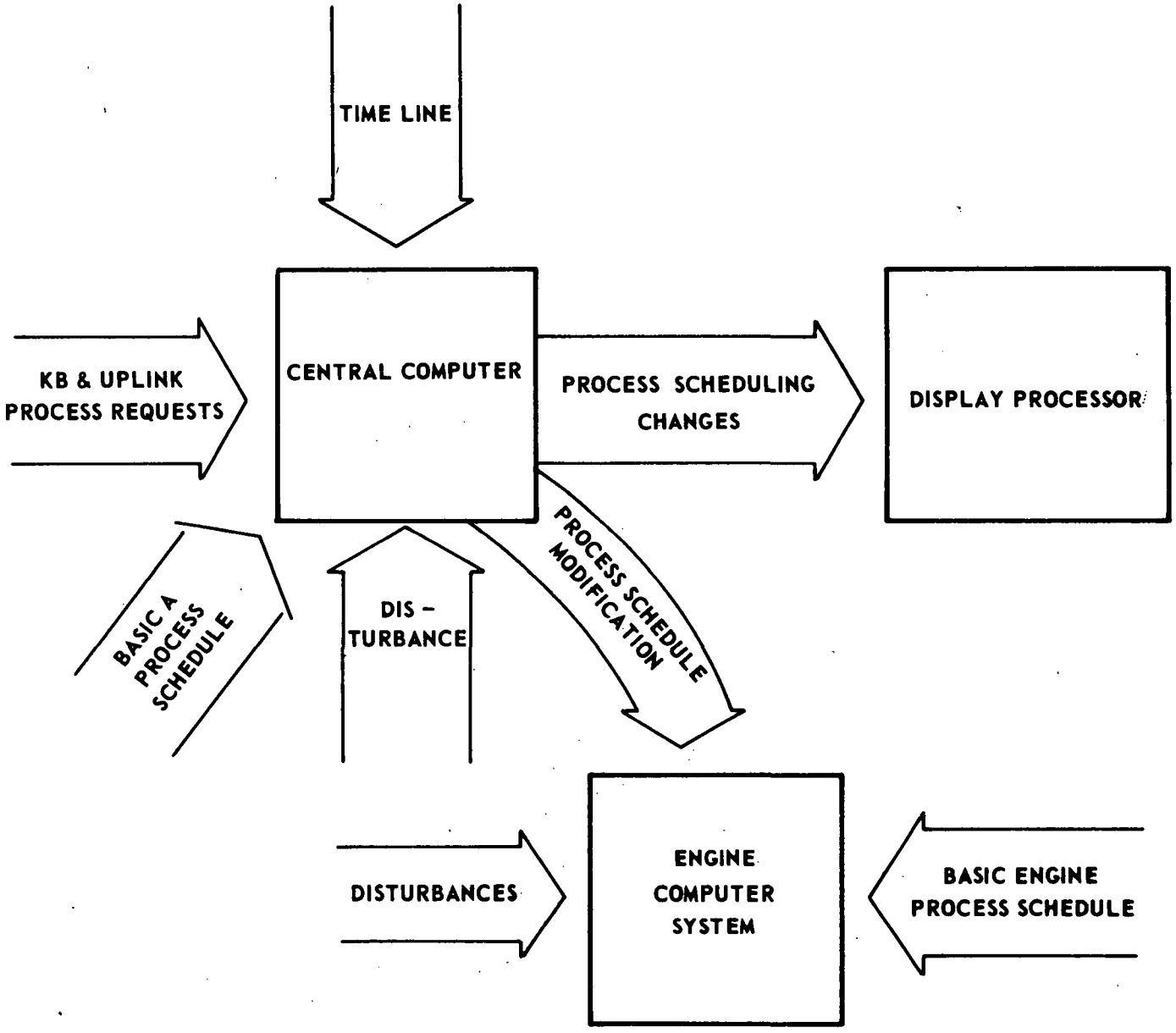


FIGURE 2. SYSTEM PROCESS SCHEDULING FLOW

From the preceding, our approach becomes apparent. The birth and death of new processes occurring in the central subsystem, assuming a steady-state operation, are analyzed. The executive operations required to cause a process to be executed upon command are cited. This process may conclude successfully or be disrupted by an error. If disrupted, executive procedures are detailed that record suitable concurrent information (for post-flight analysis) and mask the fault, usually by reconfiguration. The process is reentered at a later time, competing priorities permitting, until execution is complete. When execution is complete, termination procedures are invoked to return resources and perform executive list maintenance. The central computer executive is chosen for study since we believe that the engine computer, display processor and System Control Unit executives will be a subset of the central computer executive.

The central computer executive is designed as a set of cooperating process modules. (See figure 3.) The modular organization is broken down into six major classifications:

- Kernel,
- Input Output,
- Timeline,
- Error Masking,
- Displays, and
- Status Input.

Within these major categories the constituent process modules are delineated. Process modules may be removed from cooperation with the Executive Kernel or other modules added with minimal disruption of the executive functions.

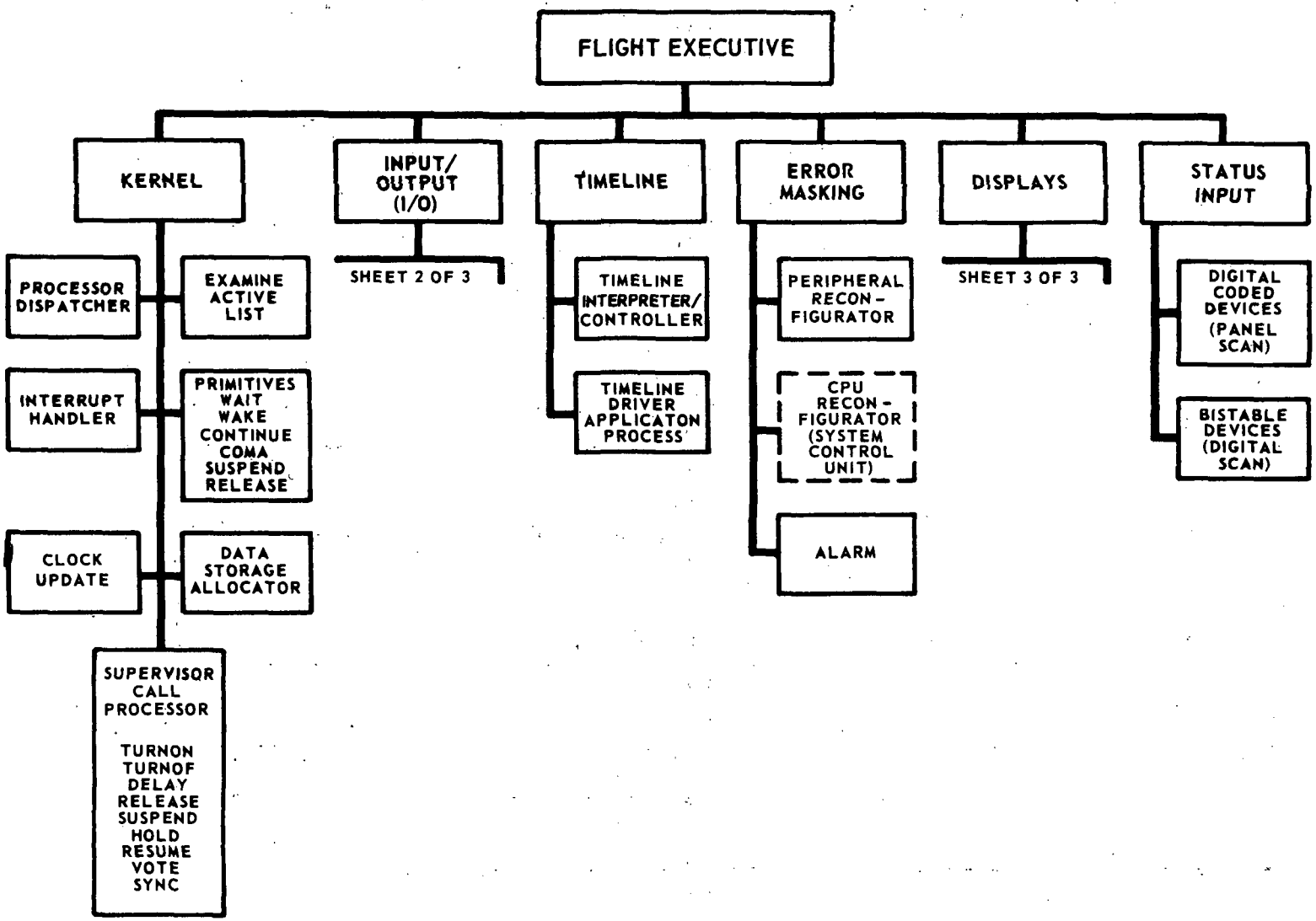


FIGURE 3. CENTRAL COMPUTER EXECUTIVE ORGANIZATION DIAGRAM (SHEET 1 OF 3)

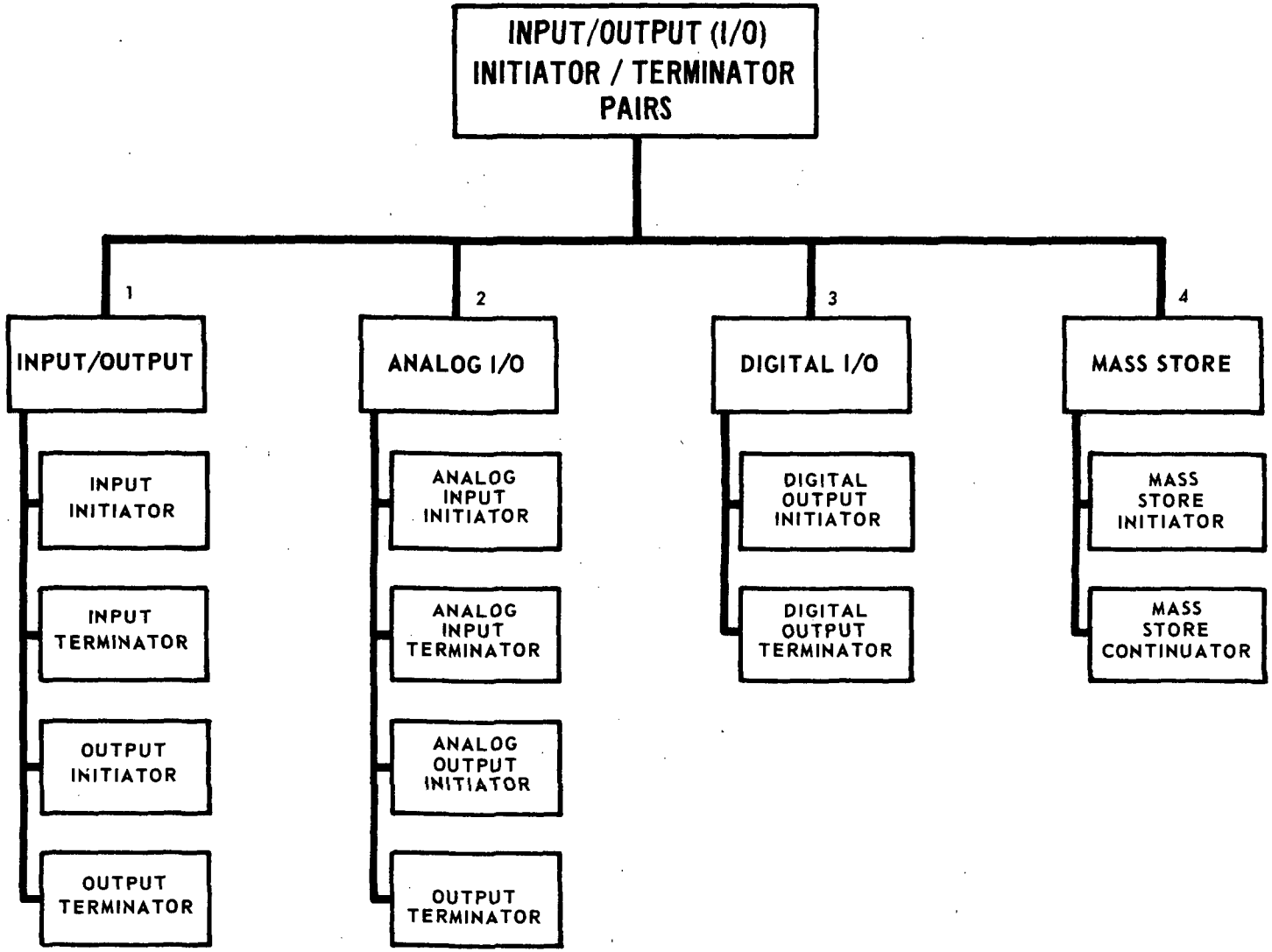


FIGURE 3 CENTRAL COMPUTER EXECUTIVE ORGANIZATION DIAGRAM (SHEET 2 OF 3)

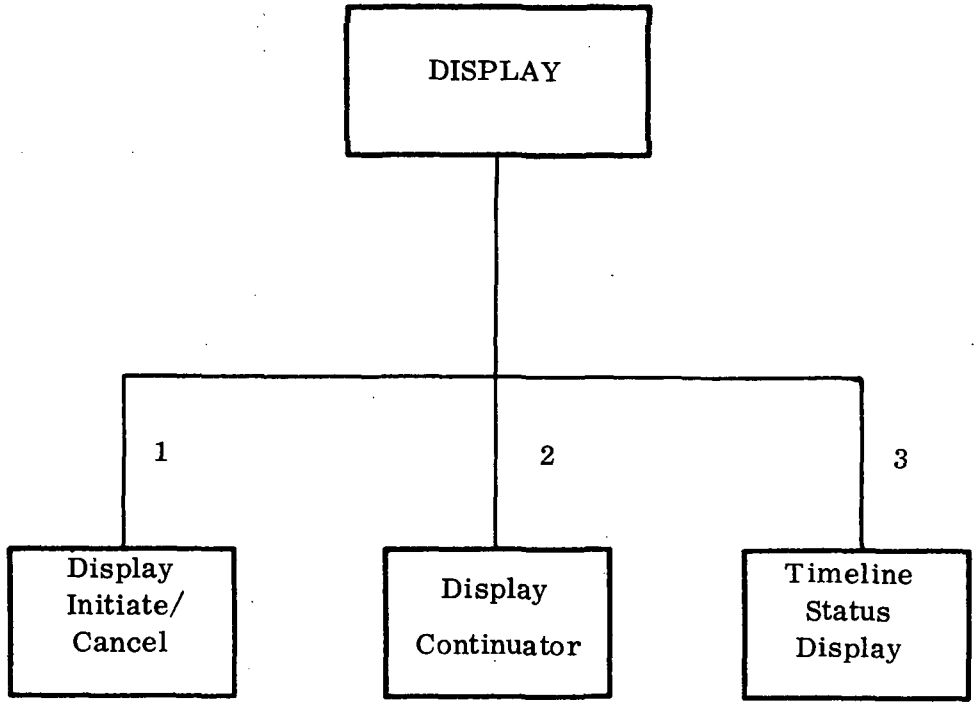


FIGURE 3. CENTRAL COMPUTER EXECUTIVE ORGANIZATION DIAGRAM (SHEET3 OF 3)

Page Intentionally Left Blank

SECTION III. PROCESS FLOW REGULATION

Regulation of process flow is the dominant purpose of an executive. In the direction of this flow, certain conflicts must be resolved prior to process activation. Such resolution is generally made by choosing the highest priority contender, although more sophisticated algorithms are sometimes warranted. During process execution, other conflicting demands for resources arise - usually for main storage or for peripheral devices. Such conflicts are regulated as mentioned above. Finally, terminating processes must relinquish resources to permit allocation to lower priority processes whose execution has been held up due to lack of these resources.

A. Activation

To illustrate the principles involved in activation of a process, refer to figure 4. This figure reflects the states that a process can enter as a result of a Supervisor Call (SVC) or as a result of preemption by a higher priority process. State transitions are governed by a collection of programs called the Executive Kernel, comprised of the following algorithmic functions:

- Process Dispatcher,
- Examine the Active List (EXAM),
- Interrupt Handler,
- WAKE, WAIT, SUSPEND, RELEASE, CONTINUE, COMA primitives (possibly hardware implemented),
- Clock Update,
- Data Storage Allocator, and
- Supervisor Call processors TURNON, TURNOF, EXIT, DELAY, RELEAS, SUSPEN, HOLD, RESUME, VOTE, SYNC.

Control of the state transitions is illustrated in figure 5, Executive Logic Flow. When Executive control is in EXAM, the system is said to be dormant. In this case, control will loop on a check to determine whether or not time-dependent processes have become critical. Critical processes are waked (activated) and control then passes to the Process Dispatcher for subsequent Arithmetic Logic Unit (ALU) allocation and process execution. If no time-dependent processes require activation, possibly hardware diagnostics can be executed. Otherwise, the looping continues until an interrupt is detected. Interrupt response processes are scheduled using the WAKE primitive, with the exception of the Clock Interrupt. Here, control passes

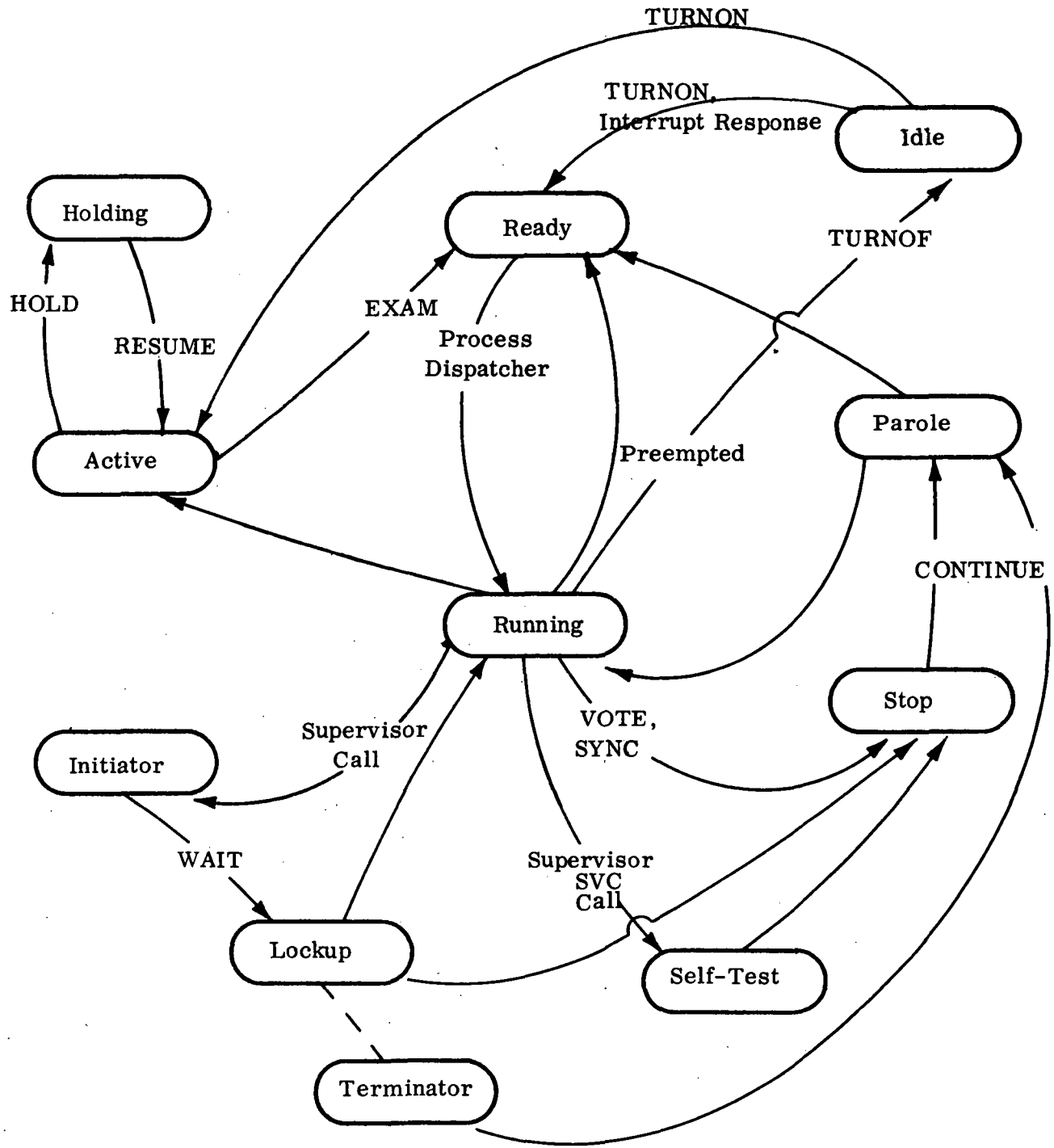


FIGURE 4. PROCESS STATE DIAGRAM

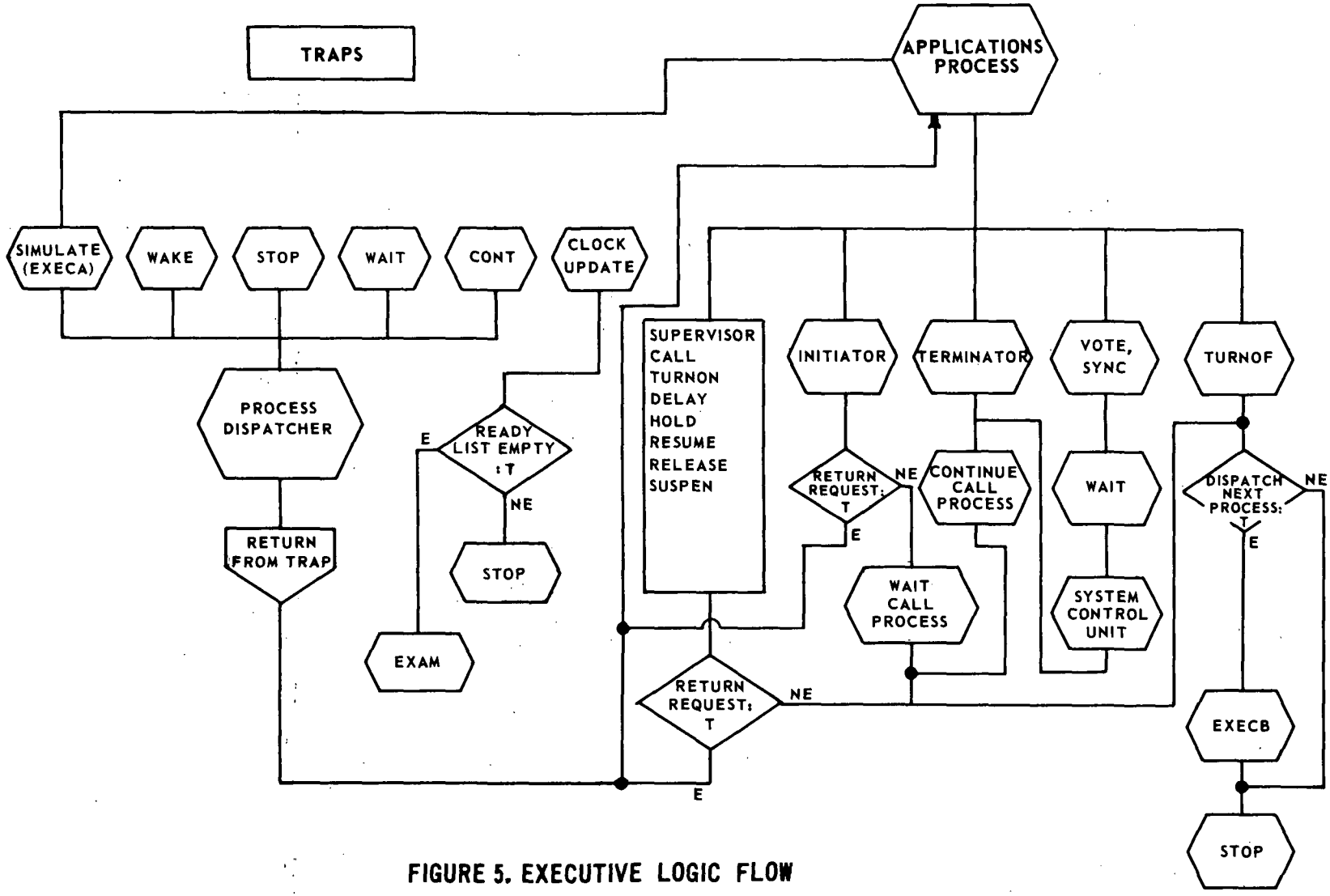


FIGURE 5. EXECUTIVE LOGIC FLOW

to the Clock Update routine and then can pass back to EXAM to ascertain whether or not the passage of time has caused a process to become critical. If this, in fact, proves to be so, the process is waked and possibly becomes the executing process. Application processes may communicate with other processes and hardware via Supervisor Calls only. As the figure shows, Supervisor Calls to Initiators can return to the executing process. This facility is provided to enable external communication without relinquishing control. Control can be given up temporarily by an option in a call to an Initiator. In this case, the Terminator associated with this Initiator can return control at the completion of the operation. Control is relinquished through the use of the Supervisor Calls EXIT and TURNOF.

A process may be suspended and released by the Executive when it is in any of the states shown in figure 4.

Process activation can be considered from the point of view of the executive system or from that of the process. From the executive system point of view, the following techniques must be provided to ensure timely process activation:

- The Executive Kernel maintains two lists of Process Control Words sorted by priority level with the highest priority items chosen before the lowest. The lists, known as the Ready List and the Active List, contain process identification and priority level data used by the Process Dispatcher to resolve priority conflicts and to allow scheduling of time-dependent processes. In addition, each entry has a pointer to the Process Control Block for the associated process.
- Software clocks are maintained for several purposes. Of chief interest here is the need to decrement the time interval until next execution of a periodic process. The time interval, Execution Delta T, is kept in the Active List as part of the process state and identification data.
- If the clock updating routine is executed, the Active List is then searched for processes that have become time critical and are candidates for Ready List insertion. Ready List candidates are inserted in the Ready List according to process priority level.
- Data buffer areas are provided when requested. Associated with this provision is the necessity to maintain an inventory of available main storage and provide dynamic relocation when contiguous areas are assembled by the "garbage collecting" function.

- Process terminations must trigger the executive routine that searches the Ready List for processes that have been waked.
- If the executive is not otherwise busy, it loops around a test to detect time critical processes.
- Executive call sequences provide an interface between the Data Management System and application processes. An application process must communicate with an external process or peripheral device via the executive.
- Validity checks are conducted on all executive call sequences.

With the preceding executive structure, processes can be activated as follows:

- An interrupt arrival will cause the process to be placed in the Ready List at the selected priority level. From this point, executive dispatching operation will cause the process to be entered.
- An executive call to activate a process, such as TURNON, will cause the process to be placed in the Ready List at the selected priority level unless the call sequence contains an Execution Delta T greater than zero. In this case, the process will be placed on the Active List. Once on the Active List, the clock update procedure will decrement the Execution Delta T until the process becomes critical, and then place the process entry in the Ready List as previously described.
- A process activate command may be detected during interpretation of uplink data, in keyboard input, in transmitted data from the central computer, or in the timeline sequence. The activate command is handled as discussed above after discovery and interpretation by the Input Terminator.

Of key importance to the preceding discussion is the overview of the scheduler operation shown in figure 6. This is an information flow diagram of the logic associated with the start signal, interrupt servicing procedures, and the Process Dispatcher convocation. The following techniques become apparent:

- Error conditions must be corrected prior to examination of other factors.
- Once error conditions have been corrected, abort requests can be answered. If the abort request is not confirmed, a special process is required to reinstate any procedure that has been preempted.

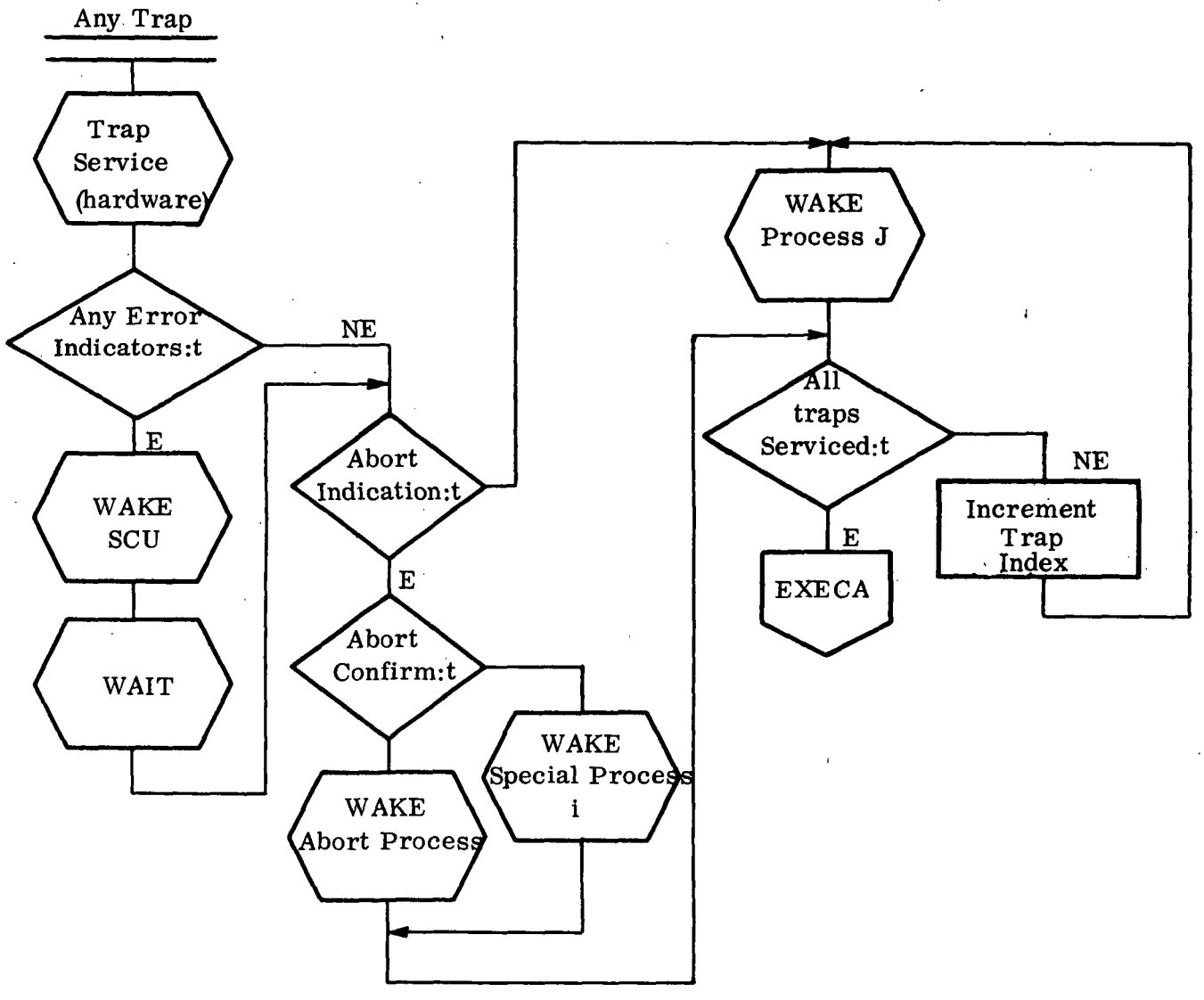


FIGURE 6. TRAP SERVICE OVERVIEW – PRELIMINARY

- If the preceding actions are not required, or if response is complete, other sources of action can be identified. Associated with each source is a process which is placed in the Ready List for execution at its priority level. Within a priority level, processes are stacked in order of arrival (FIFO).
- When all active, approved interrupt requests have been serviced, the Process Dispatcher is invoked at the entry EXECA. This entry causes the dispatcher to search the Ready List for a process demand, beginning at the highest priority level. The dispatcher will proceed to execute the first demanding process it finds for which there are sufficient available resources.

B. Device Allocation

Device allocation will be a factor in process execution when the executing process reaches a point at which it requires a peripheral device. Presently, allocation is to be effected according to process priority. It appears that investigation is required to determine if and when a current input/output (I/O) should be interrupted and the device reallocated to a higher priority demand. There is some basis for belief that allocation should be based on message priority. This is another area worth investigation during the simulation study.

Another class of allocation occurs in the event of a device failure. An alternate device, if available, must be selected and allocated to replace the failed peripheral. This allocation problem is discussed under CONFIGURATION CONTROL.

The allocation of peripheral devices will be the responsibility of the Initiator process. The exact number of Initiator processes, not now known, is dependent on specific avionics configuration. However, the need for the following Initiators has been recognized:

- Input Initiator
- Output Initiator (includes flight recorder and maintenance recorder)
- Mass Store Initiator

- Display Initiator
- Analog/Digital Output Initiator
- Timeline Display Initiator.

The main factor obscuring possible requirements for the Initiator process is that the Initiators are hardware dependent, and in addition, it may be possible to group basically common Initiators into a single process. This amalgamation, however, makes modification more difficult. These processes along with other critical executive processes are described in detail in the appendices.

C. Main Storage Allocation

Main storage allocation requirements are restricted to possible allocation of data areas. Storage allotted to process instructions is assumed capable of containing all concurrent processes. Mission phase dependent processes may be overlaid on preceding phases, giving better storage utilization and circumscribing requirements for storage allocation. However, it may be impractical to provide enough main storage for all concurrent data area requests. For this reason the Data Storage Allocator process is provided. Data Storage Allocator procedures are discussed in the appendices.

D. Central Processing Unit (CPU) Allocation

The allotment of CPU resources among many relatively slow and irregular peripheral device drivers can have a marked influence on executive design. To free the CPU for other activities and maximize device thruput, the conventional tactic has been to provide a device completion signal to the CPU. The receipt of this signal causes a device driver process to actuate the device to perform any pending operations. This technique allots to the device only the CPU time required to respond to the signal and actuate the device. The saving of CPU resources compared with programming the CPU to detect the completion signal is dramatic, since the devices operate irregularly and the exact response time for the completion signal cannot be predicted.

The allotment of CPU resources has also led to fragmentation of applications processes. In the multiprogramming real-time environment, it is desirable that no process utilize the CPU for "long periods" relative to the system response time requirements. Also, contributing to fragmentation has been the systems design principles of:

- Sharply focused planning
- Modular programming

- Functional segmentation
- Ease of checkout
- Ease of modification.

The executive system design outlined herein will allocate CPU resources by multiprogramming and provide the structure to permit flexible response to momentary system overloads during critical operations.

E. Process Suspension/Termination

Suspending or terminating a process has several ramifications:

- The process may be suspended by the Executive.
- A process may be placed in a hold condition.
- The process may relinquish control to reenter and possibly check for completion of an event.
- The process may surrender control to set a delay to cause process execution at some time in the future.
- The process may make an executive call for an I/O operation. The process is locked up during the I/O operation and reentered upon completion of the I/O.
- The process may be interrupted and preempted by a higher priority process such as an error handler process. Upon completion of higher priority processes, control is returned to the interrupted process and execution continues.
- The process may terminate. In this case, the process must be reinitialized in order to run again.

Executive services are provided to accomplish the preceding needs using the primitives WAIT, WAKE, SUSPEND, COMA, CONTINUE and RELEASE as components to construct the executive service routines. Although these primitives are discussed more thoroughly in (2), the salient points are presented below.

---

[2] Kennedy, Sr., J. R.: Spaceborne Computer Executive Routine Functional Design Specification - Volume III: Executive Routine Primitives and Process Control, NASA Contractor Report, Contract NAS8-24930, Huntsville, Alabama, March 1971.

Process suspension is accomplished by the executive call, SUSPEN, setting a suspend marker in the Process ID Description Word of either the Ready List or the Active List. The Process Dispatcher will not choose a process for execution if this bit is set. Resources allocated to the process may be optionally released.

Conversely, the process is released by resetting the suspend marker, via the executive call RELEASE. This allows the Process Dispatcher freedom to choose this process for execution at the appropriate time.

Similarly, a process operating on a relative time base, as described under TURNON in the appendix, may be placed in a hold condition using the executive call HOLD. The Process ID Description Word hold bit is set in the Active List. The Clock Update routine recognizes the hold condition and does not decrement the Execution Delta T. Processes that are executing continue to execute.

In contrast, the hold condition is released using the RESUME call sequence. The Resume routine resets the hold marker and allows decrementing of the Execution Delta T.

The EGRESS call may be used to relinquish control to the executive while remaining on the Ready List. Entry location upon return can be reset as required. In a similar manner, use of the DELAY call will cause process execution at a time in the future. The return is to the next sequential process location. This feature permits cyclic operation at a constant period of execution.

Input/Output operations can involve considerable time delays in terms of computer cycles. To utilize these CPU cycles more effectively, the I/O process provides optional echoes. That is, a branch back is provided after initialization of the I/O operation. In addition, the process is locked up until the operation is completed. At this time the process is paroled and can continue running. While the process is locked up (WAIT), the CPU is utilized in the execution of lower priority processes.

Preempting process execution using a trapping action is assumed to be accomplished using hardware and will not be analyzed further here. It is discussed in Volume III of this report.

Termination of a process is consummated by the executive call TURNOF. The process is removed from the Ready List. Resources are made available to lower priority processes. To be executed in the future, the executive call TURNON or DELAY must be effected.

SECTION IV. FAILURE RESPONSE TACTICS

Failure response implies failure detection and the existence of relationships associating failures with the required responses. Several methods of failure detection in response to expected sources of failures are provided. Executive call sequences are checked for validity to detect both hardware and software induced errors. Similarly, other active error detection procedures that are provided are: limit-checking critical variables; watchdog timers; hardware diagnostic programs (assumed to be executed in microcode); and, software examination of status registers to detect anomalies. Passive error detection procedures are triggered by fault conditions.

In contrast to the above method of classification, detection procedures can also be typed by hardware. Error detection schemes for the computers depend on fault triggering, stall alarms, diagnostics, and validity checks on executive call sequences. Error detection schemes for the peripheral devices, on the other hand, use watchdog timers, fault triggers, and status checking. In addition, distinction can be made by processes. Initiator processes perform call sequence validity checks. Terminator processes accomplish status checks. Additionally, terminator processes are charged with responsibility for separating transient errors from solid failures. Once a failure has been detected (by retries and counting), response to the fault becomes the critical item. One factor associated with the response to any failure is the recording of data concurrent with the failure to expedite postflight analysis. Pertinent failure data is to be transmitted to the maintenance recorder for future evaluation using an executive call to the I/O processes. Once the failure has been recorded, the fault is masked by reconfiguration, discussed more fully under CONFIGURATION CONTROL.

To the extent possibly, failure recovery programming must be independent of hardware configuration. Configuration dependency must be associated with descriptive tables of hardware status, select codes, redundancy requirements and allied information.

Page Intentionally Left Blank

SECTION V. CONFIGURATION CONTROL

Configuration control is achieved in different ways for computers as contrasted to peripheral devices. Reconfiguration of the computers is accomplished in conjunction with the System Control Unit (SCU). Utilization of both software and hardware procedures is possible.

Software computer reconfiguration assumes the SCU has access to computer status registers and possesses the means of transmitting signals to an arbitrary computer. The following computer status information must be available to the SCU:

- Computer state indication, such as Running Voting Controlling, Running Voting Not Controlling, Self-Test, etc.
- Shared Spare Available
- Failed by SCU
- Self-Test complete
- Sync request
- Vote Request
- Disagree
- Error Indicator
- Halt
- Switch state for SCU-CPU data paths (if switched).

In addition, the contents of the program counter for a computer must be available to the SCU. The advent of a computer fault is assumed to cause execution of the appropriate SCU response process.

The SCU is, additionally, provided with the capability to cause a computer to enter a variety of instruction sequences.

A. CPU Software Reconfiguration

Although the software process is detailed in the appendices, a baseline for the software approach to reconfiguration is described here.

The software process to effect reconfiguration is made up of several cooperating operations.

- Sync or Vote entries in which validity checks are performed on the requestor identity and status.
- A statistics gathering thread which analyzes the computer system accumulating data on the distribution of states among the computers via a push down stack.
- A thread that analyzes the gathered statistics to detect anomalies.
- Sequences to effect change in state of the computers.
- Watchdog timers to ensure that directives from the SCU are in fact executed within estimated time period.
- A routine to display system status via the SCU display panel.
- A routine to sense the status of switches on the SCU panel and convert to internal representation.
- Process modules to execute the SCU panel input commands.

As mentioned above, further discussion of computer configuration using software techniques is given in the appendices.

B. CPU Hardware Reconfiguration

The approach in designing a hardware controller to accomplish CPU reconfiguration has similarities and differences to the techniques described under software reconfiguration. As in the software reconfiguration procedures, input stimuli are required, accompanied by responding output sequences in the computer system. Reliability considerations coupled with economics have constrained the hardware controller to a configuration limited to few basic components in contrast to the software controller which can be implemented with varied basic techniques.

For the purposes of this discussion, the following ground rules apply:

- There will be three computers plus one shared spare computer.
- Each computer can be in one of four gross states: Running Voting Controlling (RVC), Running Voting Not Controlling (RV$\bar{C}$), Self-Test (ST), and Powered Down (PD).

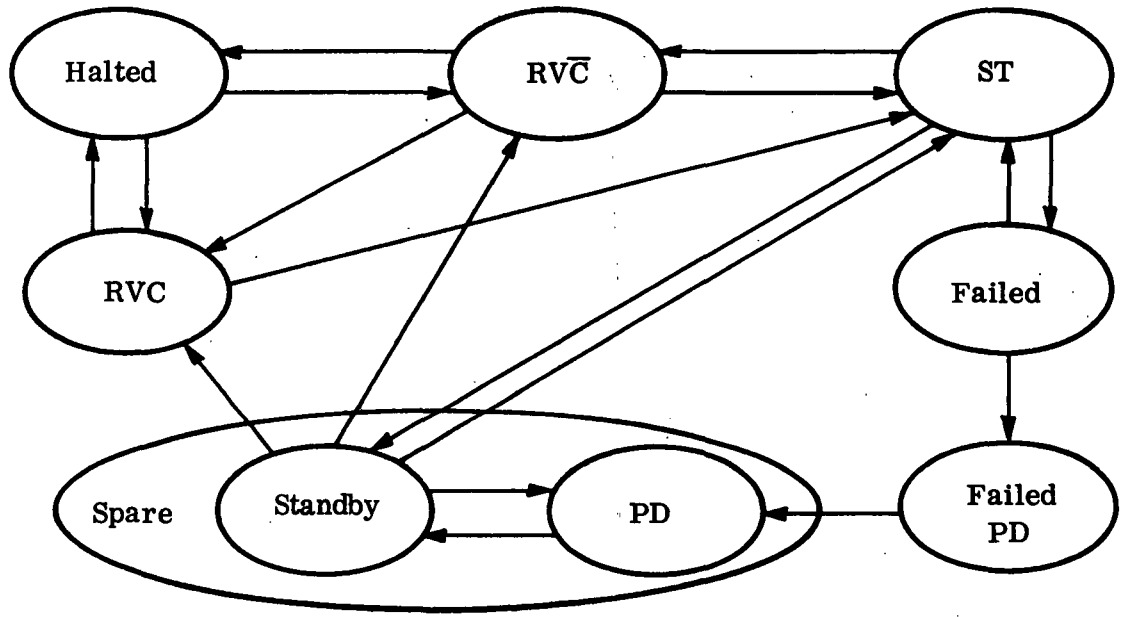
- The spare computer will be either PD or ST, possibly as a function of mission phase.
- Test and Set capability is implemented in the SCU, particularly, and is possibly implemented in the computers.
- Special procedures must be developed to separate transient failures from solid failures.
- Input stimuli and responding output sequences will be the same as discussed under CPU Software Reconfiguration.

With the preceding basis, consider the case of four computers, each capable of existing in one of four valid states. There are, then, 256 combinations of valid states possible. If five states are considered, the number of combinations of valid states possible is 625. The number of bit combinations present is a function of the bits required to represent a state. Three bits are required to represent 5 states. In addition, parity checking can be provided. It follows that many of the possible bit combinations are invalid, since a total of 8 states per computer will result from utilizing 3 bits. For four computers 4,096 states are possible, of which 625 are legal, assuming the 3-bit state representation. Possibly a more efficient method of state coding should be chosen.

Several classes of errors can occur:

- Within computers; that is, an invalid bit combination,
- Invalid combinations of computers; for example, more than one controlling computer, and
- Illegal transitions, which are transitions other than allowed in the state diagram (figure 7).

The controller must detect illegal states, invoke further means of analyzing the failure to determine the failed element, and convoke error masking sequences. If illegal states are not represented, the controller must analyze the voting response of the computers, upon demand, to detect voting anomalies which will lead to convocation of error masking sequences. In the analysis of voting, 16 disagree responses are possible for each computer state. If, for example, there are 128 valid combinations of states, then there are 2,048 combinations of computer states and disagrees (16 x 256), some of which are illegal and must be acted upon by the illegal state detector.



Failed: Halted	Sequenced to Fail state by SCU
PD	Powered Down
RVC	Running Voting Controlling
Standby	Powered but not Executing Code
SCU	System Control Unit
ST	Self-Test
Spare	Available as Replacement

FIGURE 7. CENTRAL COMPUTER STATE TRANSACTION DIAGRAM AS SEEN BY THE SYSTEM CONTROL UNIT

From the preceding discussion, several salient ideas arise:

- The design complexity increases rapidly with the number of states required to fully describe the computer status.
- Many of the possible states are illegal and can be treated using an illegal state detector, triggering possible pre-engineered sequences (3).
- The valid states must be analyzed by the designed logic to detect voting anomalies and invoke appropriate error masking sequences.
- Procedures for computer switching sequences must be defined. For example, if the RVC computer fails, the first RV$\overline{C}$ is promoted to RVC. The shared spare, if available, is set "not available" and sequenced to RV$\overline{C}$. The RVC computer is sequenced to ST. These procedures are triggered by specific input combinations on the disagree lines and status lines from the computers.
- After the scope of the design problem has been bounded by calculating the number of possible permutations, truth tables are formed, and functional equations determined; the logic requirements can be optimized using digital design techniques.
- Requirements for resolving voting "ties" must be determined.

C. Peripheral Reconfiguration

In contrast to computer reconfiguration, which is accomplished by specialized combinations of software-hardware, peripheral reconfiguration is handled as an executive subroutine. This subroutine reconfigures peripheral devices by manipulation of executive tables. Since all peripheral device requests are channeled through the executive, tables of values of internal address codes of

---

[3] Raytheon Company: Preliminary Technical Report Machine Implementation Study for Phase B Computer System, Sudbury, Mass., 23 October 1970.

devices are maintained by the executive. When a device is requested, the executive searches the table for the device address and uses it to perform the device selection. For every device, a table is constructed giving alternate devices. The responsibility of the reconfiguration process, crudely, is to select a preferred alternate in the event of device failure. The alternate device address replaces the failed device address in the device selection tables. Thus, the reconfiguration responsibility is hidden from the requesting process. As far as the requesting process knows, the message is input or output through the initial device. Reconfiguration, then, is of no concern to the applications process, unless no alternate is available. If execution of the peripheral operation is impossible, the executive must so notify the application process so that an alternate action can be taken.

In recapitulation, peripheral reconfiguration is achieved in the executive by manipulating internal executive tables. Successful reconfiguration can be hidden from an applications process, unless further study shows that notification to the applications process is desirable. As with other executive processes, a more detailed description of the peripheral reconfiguration process is presented in the appendices.

SECTION VI. INTER-SUBSYSTEM INFORMATION FLOW

Through wide changes in configuration and complement of the Data Management System, the stable consideration of a master subsystem controlling peripheral subsystems remains. Vital to this control concept is the composition and treatment of the components of flow (messages). To implement the information control concepts shown in figure 2, the following message-oriented approach is recommended as a baseline, although the processes required for inter-subsystem communication are somewhat hardware dependent and impossible to completely specify at this time.

- Communication with a peripheral device consists of call sequence validity check, buffer allocation, device manipulation, message transmission error checking, and program accounting associated with message termination. A significant part of the preceding is not hardware dependent at a high level of analysis. A significant portion of our programming system can then be specified without complete knowledge of hardware specifications. The hardware and software, however, ideally should be specified concurrently.
- Communication occurs between a central computer and peripheral devices with varying sentient and logical capabilities. Error checking efficacy will vary with the logical faculty of the peripheral subsystem. Where the peripheral device is another computer, the following techniques are propitious. Where the device possesses smaller capability, other hardware dependent procedures must be devised.
- A data bus controller will execute message transmission and notify the computer that the transfer is complete. Other capabilities may be added to the data bus controller later. Such capability could include line control, code conversion, buffering and queuing, error handling and recovery, and collation and analysis of statistics (4).

---

[4] Hebditch, D. L.: Programmable Control Units - A Way Forward in Data Communications Telecommunications, Dedham, Mass., November 1970.

- The data bus controller affords the chance to view information flow on a message basis. This follows from the fact that the actual word-oriented transfer operations are performed by the data bus controller.
- Every message will have a fixed length and format header. In addition, the message may have a body and will have a terminator.
- The fixed length header message includes message identification, source/destination information, error check information, time, message length (if required), action request class, and terminator code.
- Error checking by hardware (parity) is done on each word transmitted.
- Hardware and software checks are done on each message sent and received. These checks include status, longitudinal record check, format check. Further study may expose the need for additional message checks.
- Every message is given a number which must be sequential with the unique transmitting device. Messages received must pass the sequential check to prevent the undetected loss of a complete message due to hardware failure, system overload, line contention, or scheduling conflicts (5).
- Every message must be acknowledged within an arbitrary time. Acknowledgement means that the message has been received correctly. If the acknowledgement is not received, an error is assumed. The following method is used to detect the fault. A lock marker is placed (in memory) on each message transmitted. This prevents destruction of the message buffer. Acknowledgement causes the lock to be removed. A watchdog timer is set simultaneously with the lock. When the timer counts down, the lock is removed. This is an error condition and is treated as a hardware failure.

---

[5] Mills, D. L.: Topics in Computer Communication Systems Lecture Notes, University of Michigan, Ann Arbor, Michigan, May 1969.

- The preceding techniques for inter-subsystem communication can be logically divided into two classes of processes, Initiators and Terminators. These classes will suffice to handle broad classes of subsystems.

A typical example of an Initiator is the Output Initiator, which effects the succeeding procedures:

- Call sequence validity check,
- Parameter passing,
- Selection of the device control word from the Peripheral Control Tables,
- Completion of message formatting,
- Interface with the data bus controller to initiate output,
- Selection of appropriate return to calling process, and
- If there is no immediate return, invoking process suspension.

In contrast to the Initiator process presented above, the Input Terminator is an example of a typical Terminator. The following activities must occur:

- Reset of watchdog timer,
- Hardware error status check,
- Separation of transient from solid failures by requesting message retransmission. An arbitrary number of successive failures will qualify the fault as solid. Hardware diagnostics and reconfiguration processes can then be invoked. (Transient failures can be recorded although the possibility that the recording system can become saturated must be considered),
- With the hardware status successfully checked, analysis of various software message checks, such as longitudinal record check and message number, is performed. Failure to pass these checks results in treatment similar to hardware check failures,

- If the message passes the preceding error analysis, examination for action type is made. This analysis is comprised of the procedures that allow the master computer to transmit process execution information data to the slave computers. Action requests are: Acknowledge, Retransmit, No Data, Start of Message, Cancel, End of Message, Schedule, and Data,
- Determination of the action request identity by the Input Parser. This may lead to the several different process paths as shown in the appendix. An important consideration is the delineation of the different paths representing the actions: acknowledge, retransmit, no data, start of message, cancel, end of message, schedule, and data. A more precise presentation is given in the Appendix.

From the preceding, it can be seen that substantial parts of the concepts for subsystem control can be implemented prior to exact hardware definition. Subsystem communication processes are grouped in two broad classes, Initiators and Terminators, with hardware independent functions defined, while precise hardware design data is being determined.

SECTION VII. EXECUTIVE STRUCTURES

The executive structure provides the power and flexibility to implement arbitrary scheduling strategies. The executive can operate in a synchronous mode, permitting immediate response to faults, however. These error-triggered interrupts cause a momentary asynchronous mode of operation. The Supervisor Calls, SYNC and VOTE, give methods of reestablishing synchronization for a process. The executive can also operate in an asynchronous mode. If desired, it can operate in a mixture of modes using the SYNC or VOTE procedures to establish synchronization when required. Synchronization is required merely to ensure that voting procedures are performed on selected identical variables by all computers. Therefore synchronization can be viewed as affecting a particular process and furnishing Supervisor Calls to effect the isochronism of the computers.

In addition to providing synchronization to the degree required, the executive can be implemented to enhance a major-minor cycle method of scheduling. A proposed method of implementing the major-minor cycle scheduling divides available time into minor cycle slots. To provide 20 msec minor cycles, there are 50 slots. In each active slot, the minor cycle processes - plus 1 major cycle segment - are executed. Then, every second, the major cycle is completed with the minor cycle repeated 50 times. Empty slots, for example the even or odd numbered, are provided for expansion or for asynchronous event processing. Unfortunately, the presence of a hardware fault indicates that the information in the computer is suspect, and action must be taken immediately without waiting for even the next available slot. This is accomplished in the proposed executive by allowing a priority structure, with the error processes given a priority immediately below that of the Executive Kernel. This priority, coupled with fault-triggered trapping, assures timely error procedures to guarantee fault tolerant operation.

In addition to the above, the priority structure has other purposes; namely, to give an order of preferred execution of concurrent processes. Although this preferred execution can be accomplished by a rigid structural ordering of the applications processes, for reasons that will become clear, it is considered more desirable to effect this ordering using a priority level structure. Similarly, the priority structure can determine which processes may be delayed in the event of momentary system overload. Such overload might occur in rendezvous, docking or landing mission phases similar to that of the Apollo 11 landing (6). The system can undergo automatic graceful degradation

---

[6] Hamilton, M. H.: Letters, Datamation, Barrington, Illinois, 1 March 1971.

via the priority assignment, recovering when the overload condition disappears. Simulation, possibly non-deterministic, can determine the optimum priority assignment by enacting the overload situation, observing the Executive response, and altering the process priority assignments until a suitable compromise is reached. However, further study is required to know whether or not this priority level technique is sufficient or whether more sophisticated process selection algorithms are required for mission phases exhibiting relatively large computational demands, such as Landing.

In the basic cyclic mode of operation, conflicts must be resolved between the need to execute the minor (20 msec) operations at a constant 20 msec period and the requirement for computations that may exceed 20 msec. The solution to this conflict should ideally be dictated by the system requirements in the usage of the data gathered during the minor cycle. As such, it is now premature to impose a resolution to the conflict. However, two divergent methods arise as a result of the two different executive structures, the proposed Executive and the strictly (non-multiprogramming) Major-Minor cycle executive.

The Major-Minor cycle structure is rigid. As previously discussed, time slots are provided for each cycle (20 msec). Every 20 msec, the Minor cycle plus 2 percent of the Major cycle is executed. Therefore, execution times must be pre-calculated for every process and the system constructed very much in the way that a watchmaker builds a watch. This is to assure that the cyclic processes are actually executed during the intended periods. A system such as this can be awkward to modify. Difficulties in implementing the changes that are bound to occur over the life of the Shuttle, coupled with the deficient error handling procedures noted previously, make the Major-Minor cycle approach the less desirable.

In contrast to the rigid time-slot approach, the proposed Executive operates in a different way. A scheduling structure and priority level assignment are provided that allow periodic execution of processes based on an arbitrary unit of time, independent of a minor cycle. This structure is inherently more versatile. After initialization, periodic processes are executed by use of the Supervisor Call DELAY. The period of delay can be set to the desired amount.

The implication of the preceding discussion is that in the normal mode of operation, the system can be designed to operate in a cyclic mode identical to the Major-Minor cycle. If maintaining a 20 msec Minor cycle period is considered essential, then the Minor cycle processes are assigned higher priority than the Major cycle, random demands, or other cyclic processes. The lower priority processes will be preempted every 20 msec and the Minor cycle processes executed. Upon completion of the Minor cycle processes, the preempted process is reentered and runs to completion.

The occurrence of a random demand is treated using a different technique than the Major-Minor cycle. Here, the priority structure affords latitude in scheduling the response. Critical events (high priority) can be treated without intervening minor cycle applications process operations by waking the response process and executing it immediately through the Process Dispatcher. Less critical random demands can be deferred to be executed at the leisure of the Executive.

Modification of the priority assignments is accomplished by changing priority assignments in the Process Control Blocks. This is simpler and less subject to error than recompiling and rebalancing the system to effect differences in performance.

The proposed Executive is a system that gives the capability to implement the optimal Space Shuttle scheduling strategy, whether it prove to be Major-Minor cycle or some other technique elucidated by simulation.

Considering the preceding discussion, the Executive operational aspects are similar to the following for a particular mission phase: In the cold start initialization, the Timeline Interpreter/Controller (TLIC) is loaded along with the Executive programs into the central computers. The engine computers will require an Executive which can be transmitted by the central computer from the Mass Store as required.

After the executives with accompanying data have been loaded and the load operation verified, the TLIC can be initiated to begin analyzing timeline command sequences. Intra-phase events can be sensed, and phase termination procedures invoked.

For example, when the TLIC determines that the Prelaunch Checkout phase has been successfully completed, the TLIC will call in from the Mass Store the processes necessary to accomplish the Ascent phase. Each process will be accompanied by a Process Control Block (PCB) containing information required for executive control (2). Ready and Active lists will be initialized so that the Process Dispatcher can initiate phase execution.

Processes may be phase independent or phase dependent. If phase dependent processes are to be overlaid, then means to disable/enable storage protection, under program control, must be provided. This suggests that phase-dependent processes be segregated from phase-independent processes.

The TLIC will operate in a similar way for all mission phases, calling in new processes as required. The priority assignments will possibly be changed to assure that phase-dependent critical operations are executed within time constraints during certain mission phases such as landing. Contingency features to further assure that the time constraints can be met are possible within the Process Dispatcher for these critical mission phases. When the phase termination criteria are sensed by the TLIC through examination of the Current Status Tables, phase termination processes are waked. Upon completion of these processes, initiation of the next phase begins. This repetition is followed until all phases are completed and post-flight analysis is commenced. Current Status Tables contain system global variables which are periodically updated by data gathering and calculation routines.

SECTION VIII. PRELIMINARY MAIN STORAGE REQUIREMENTS ESTIMATE SUMMARY

While the executive functional design is aimed primarily at providing the basis for a simulation model, the analysis can also be used to generate preliminary estimates of executive main storage utilization. The flowcharts furnished in the Appendices are the foundation for these estimates. Program requirements are estimated at 12.5-13.5K 32-bit words. A breakdown of the components of the estimates is given in table 1. Data storage requirements are even more heavily hardware-dependent and, therefore, a method is proposed for calculating the storage utilization rather than a preliminary estimate. Such a calculation will be useful in hardware/software trade studies to estimate the impact on storage requirements of software implementation of functions such as analog input, and is shown in table 1. More detailed estimates are furnished in table 2.

Symbols appearing in table 1 are defined as follows:

A = Number of Processes
B = Number of Peripheral Devices
C = Number of Active Messages
D = Number of Current Results
E = Number of Analog Sensors
F = Number of Digital Sensors
G = Number of Active Processes
H = Number of Data Storage Blocks
I = Number of Analog Output Devices
J = Number of Digital Output Devices
** = For software driven analog/digital input
*** = Hardware manipulation only

TABLE 1. SUMMARY OF PRELIMINARY MAIN STORAGE REQUIREMENTS ESTIMATES

Process Modules	SCU	Storage
Executive Kernel		2010
Reconfiguration - Peripheral devices		1000
SCU	8000	
Data Bus Input/Output		2190
Analog Input/Output**		2650
Digital Input/Output**		370
Timeline		1320
Mass Store Input/Output		1200
Panel Scans		600
Display Control***		970
Alarm		500
TOTAL:	8000	12,810

Data Requirements		
Masks, Common Constants		300
Process Control Blocks	32 words/process	32*A
Peripheral Configuration Tables	1 word/device	B
Device Failure Tables	1 bit/device	B/32
Device Status Tables	1 bit/device	B/32
Alarm Tables		300
Main Storage Tally	#blocks/32	H/32
Message ID	1 word/message	C
System Current Values	1 word/current result	D
Timeline Status		200
**Analog Scan Tables	3-4 words/sensor	4*E
**Digital Scan Tables	1 word/sensor	F
Digital Status Tables	1 bit/sensor	F/32
Ready List	2 words/process	2*G
Active List	3 words/process	3*G
Data Fixed Buffer Area	5 words/peripheral device	5*B
Analog Output	1 word/device	I
Digital Output	1 word/device	J

Data Storage = 800 + 32*A + 6 1/16*B + C + D + 4*E + 1 1/32*F

+ 5*G + H/32 + I + J

TABLE 2. PRELIMINARY MAIN STORAGE REQUIREMENTS ESTIMATE

Process Modules	SCU	Storage
Process Dispatcher		100
Clock Update		140
Activate Process (TURNON)		140
Terminate Process (TURNOFF)		140
Delay		100
Release Process (RELEAS)		100
Suspend Process (SUSPEN)		100
Hold/Resume		150
Vote		150
Sync		80
Examine Active List		200
Data Storage Allocator		610
Input Initiator		380
Input Terminator		750
Output Initiator		360
Output Terminator		700
Analog Input Initiator		800
Analog Input Continuator		1500
Analog Output Initiator/Terminator		350
Digital Output Initiator		120
Digital Output Terminator		50
Mass Store Initiator		100
Mass Store Continuator		1100
Timeline Interpreter/Controller		800
Peripheral Reconfiguration		1000
CPU Reconfiguration	8000	
Alarm		500
Display Initiator		470
Display Continuator		500
Timeline Display		520
Panel Scan		600
Digital Input		200
TOTAL:	8000	12,810

SECTION IX. CONCLUSIONS AND RECOMMENDATIONS

Analysis of past similar computer projects, combined with study of the Space Shuttle Phase B Contractor reports, yields certain observations and conclusions relative to desirable system, computer hardware, and computer software characteristics:

- Software development costs are highly dependent on computer loading. Not recognizing this dependence in the past has led to notorious underestimation of the programming effort required for projects similar to the Reusable Shuttle (7).
- Hardware redundancy and fault tolerant software add materially to the complexity of the flight executive.
- In view of the preceding, software development must begin early in the design cycle. Analyst/Programmers must work with engineers in the development of the avionics system.
- Verification requirements must impact on hardware/software design from the beginning to the end of the design cycle, since approximately 45 percent of software effort is consumed by verification.
- Detailed man-machine interface requirements must be developed, based on astronaut-pilot needs and recommendations. Particular emphasis must be placed on critical mission phases, such as docking and landing.

Recognizing that further study is required on all phases of the avionics system, the following recommendations are presented:

- The proposed flight executive should be taken as a baseline. Efficacy of the Executive in meeting Space Shuttle requirements can be studied either by non-deterministic simulation or by implementation on a computer system similar to the flight computer.

---

[7] Boehm, B. W.: Some Information Processing Implications of Air Force Space Missions in the 1970s, Astronautics and Aeronautics, New York, N. Y., January 1971.

- Software verification requirements must be compiled as soon as possible. These requirements include test bed configuration, levels of testing, change control procedures, documentation standards, programmer/analyst management policies, computer test hardware requirements (for instance the role of emulation, digital versus hybrid simulation, test support software requirements, including data base formulation and maintenance). Interfaces between orbiter and booster engine and other avionics subsystems must be defined. These overall requirements interact with the avionics hardware and must be developed in conjunction with the avionics hardware.

- Verification can be considered as encompassing the following areas: integrated vehicle, orbiter, booster, subsystems, and software-only. Different verification requirements exist for software-only versus the various relative amounts of integrated hardware/software verification. For example, the software-only verification can be effected using present large scale digital computer systems with the appropriate additional programming, combined with conventional tactics (for example, desk debugging). Higher levels of integration, however, require computer facilities similar, if not identical to, the flight computer. This is the likely role of an emulation strategy - to provide the flight computer instruction set within the test bed environment.

- Hardware/software tradeoffs must be biased toward hardware implementation. The following features are useful for executive functions: hardware primitives, base register addressing, index registers, push-down stack implementation, memory protect, fault triggering, restricted storage area references, multiport memory access, priority interrupt, and hardware (possibly microcode) initialization and restart. Floating point hardware is recommended for applications programming.

- Certain error recovery procedures are presented for illustrative purposes. Final procedures, however, must be defined by a team of senior analyst/programmers and engineers, in certain cases subject to review of the astronaut-pilot.

- The computer loading design point (in terms of computation requirements compared with available CPU cycles) should be taken as 50%. Available CPU cycles is a crude measure of the parameter of interest, throughput. In terms of utilization of main storage, the recommended design point is 70%.

Page Intentionally Left Blank

APPENDICES

Supporting data for the Reusable Shuttle Flight Executive are provided in these appendices. This information is divided into three classes:

- Functional Procedure Descriptions,
- Process Attribute Description, and
- Functional Flowcharts.

Each of these classes is given an Appendix. In Appendix A, Functional Procedure Descriptions, information delineating the interaction of the process with the other processes in the system is provided. Appendix B, Process Attribute Description, contains information specific to a given process. Appendix C, Functional Flowcharts, is made up of process descriptions in the form of high-level functional logic diagrams.

The intention is that information of this kind will be utilized in the construction and maintenance of the data base required for the non-deterministic simulation and other aspects of the verification of the programming system.

APPENDIX A. FUNCTIONAL PROCEDURE DESCRIPTIONS

The following information is required for these Functional Procedure Descriptions:

Procedure Identifier: The process name followed by a unique, alphanumeric process identifier. This identifier will be used by the utility programs to access the process from the system libraries.

Purpose: A brief paragraph describing the functions accomplished by the process.

Approach: A brief, but complete, account of the methodology used to accomplish the process functions. Mathematical techniques must be portrayed. Stability considerations, if any, must be included.

External Procedures Referenced: A simple list of the processes called by the program. Such external processes can be cross-referenced in a utility program to determine the possible effect of modifying a process.

External Data Referenced: A simple list of the external data items utilized by the process. Considerations similar to the above apply.

FUNCTIONAL PROCEDURE DESCRIPTION

Procedure Identifier: **PROCESS DISPATCHER (PRDSPR)**

Purpose: **The Process Dispatcher examines entries on the Ready list to select the highest priority level entry awaiting execution. If the Ready list is empty, the process branches to the EXAM process in order to determine whether or not new processes have become critical. Upon detection of a process awaiting execution, the Process Dispatcher initializes the process by loading the contents of the Process Control Block into the active area and jumps to the process location to begin execution.**

Approach: **Applications processes should use the SVC call EXIT to location EXECB to avoid possible deadlocks.**

External Procedure Referenced: **Active Process**

External Data Referenced: **Global data, Process Control Blocks**

FUNCTIONAL PROCEDURE DESCRIPTION

Procedure Identifier: CLOCK UPDATE (CLCKPD)

Purpose: The purpose of the software clock update routine is to maintain current time in the software time of day clocks, to decrement analog scan class timers, to decrement watch-dog timers and to decrement execution delta T values for entries in the Active list.

Approach: The appropriate software clocks are incremented or decremented as required upon receipt of the clock periodic signal.

External Procedure Referenced: Examine Active list

External Data Referenced: Global data

FUNCTIONAL PROCEDURE DESCRIPTION

Procedure Identifier: EXAMINE ACTIVE LIST for Critical Process (EXAM)

Purpose: This process allows time dependent scheduling. The highest priority process with execution delta T equal to zero is inserted into the Ready list for execution.

Approach: This routine is entered on a periodic clock cycle. Execution delta T's are checked starting at the highest priority level and working to the lowest. Time critical processes are waked in order of priority. Necessary program accounting for the process and for gathering performance statistics is accomplished.

External Procedure Referenced: PROCESS DISPATCHER, DATA STORAGE ALLOCATOR, WAKE, Performance Statistics Processes

External Data Referenced: Global data

FUNCTIONAL PROCEDURE DESCRIPTION

Procedure Identifier: TURN ON (TURNON)

Purpose: Activate a Process places process control words in the Active list at the indicated priority level. If the execution delta T is zero, the WAKE primitive is invoked, placing the process ID directly on the Ready list for execution.

Approach: Three time bases can be used: Time of Day, Incremental, Relative to Time Zero. These bases are converted to incremental and the Process Control Word and Execution Delta T are inserted into the Active list at the indicated priority level.

External Procedure Referenced: WAKE, Data Storage Allocator

External Data Referenced: Process Control Blocks, Global data

FUNCTIONAL PROCEDURE DESCRIPTION

Procedure Identifier: **DELAY A PROCESS (DLYPRC)**

Purpose: **Delay provides an elementary method of causing periodic execution of a process.**

Approach: **Options are provided to exit to EXECA or EXECB. EXECB is preferred. The delay requested in the call sequence is placed in the Active List Process Control Word. The process control word is deleted from the Ready List.**

External Procedure Referenced: **Activate a Process, Process Dispatcher**

External Data Referenced: **Process Control Word, Global Data**

FUNCTIONAL PROCEDURE DESCRIPTION

Procedure Identifier: **TERMINATE A PROCESS (TURNOF)**

Purpose: **TURNOF provides the capability to terminate a process so that subsequent activation will require re-initialization. To be executed at some future time the process must be waked. Data storage is returned to the available pool.**

Approach: **Options are given to exit to EXECA or EXECB in the Process Dispatcher. The process control word is removed from the Ready list.**

External Procedure Referenced: **Process Dispatcher, Data Storage Allocator**

External Data Referenced: **Global data**

FUNCTIONAL PROCEDURE DESCRIPTION

Procedure Identifier: **SUSPEND A PROCESS (SUSI.O)**

Purpose: **SUSPEN provides capability to halt a process without executing a halt instruction in the CPU. The process will remain in a suspended state until released by the RLSPRC routine.**

Approach: **Suspend markers are set in the respective process control words in the Ready or Active list entry. Optional exit to EXECA or EXECB of the Process Dispatcher is provided.**

External Procedure Referenced: **Process Dispatcher**

External Data Referenced: **Global data**

FUNCTIONAL PROCEDURE DESCRIPTION

Procedure Identifier: **RELEASE A PROCESS (RLSPRC)**

Purpose: **The purpose of RLSPRC is to place a suspended process in the execution state.**

Approach: **Suspend markers are removed from the process control words in the Ready and Active lists. Optional exit to EXECB or the the process is provided.**

External Procedure Referenced: **PROCESS DISPATCHER, Applications process**

External Data Referenced: **Process control words, Global data**

FUNCTIONAL PROCEDURE DESCRIPTION

Procedure Identifier: VOTE (VOTE)

Purpose: VOTE causes the Running Voting Controlling (RVC) computer to transmit a value for voting to the slave computers. Data description (analog, digital) and dead band may also be transmitted. In addition, the RVC raises a Vote signal line to the System Control Unit (SCU). For the case of the slave computers ($\overline{RVC}$) VOTE causes input of the values from the master computer. Software to compare the input values with call sequence values is activated. In addition, a vote signal is sent to the SCU with a disagree signal in the event of comparison failure.

Approach: The computer status word is input and state determined (RVC or $\overline{RVC}$). For the RVC, after data transmission is initiated, the process is put in a suspended state. The slave computers are placed in a suspended state after the voting results have been transmitted to the SCU. When evaluation of the computer voting results and required reconfiguration is complete, the SCU releases the suspended processes.

External Procedure Referenced: SUSPEND, SCU, OUTPUT INITIATOR, Application process

External Data Referenced: SCU select codes, Global data input buffers

FUNCTIONAL PROCEDURE DESCRIPTION

Procedure Identifier: SYNC (SYNC)

Purpose: The purpose of SYNC is to cause a synchronization request signal to be transmitted to the SCU with the process then suspended. The SCU analyzes input from the computers. If all computers respond properly, the SCU releases the suspended process. In the event that error conditions are noted, the SCU causes error recovery through reconfiguration, when possible.

Approach: The SYNC process selects the synchronization signal and causes transmission to the SCU. The process then enters the suspended state using SSPND. Processes are released by the SCU using RELEASE.

External Procedure Referenced: SSPND, OUTPUT INITIATOR

External Data Referenced: Global data, synchronization commands

FUNCTIONAL PROCEDURE DESCRIPTION

Procedure Identifier: DATA STORAGE ALLOCATOR (DTSTRG)

Purpose: The DATA STORAGE ALLOCATOR (DSA) provides allocation and control of data storage areas of main memory.

Approach: The call sequence will be analyzed for errors and the error notification message and return sent using the OUTPUT INITIATOR, if required. Availability tables of main memory blocks will be maintained. For allocation, the availability tables will be searched for contiguous storage equal to the requested storage. If not available, the best available return will be taken. If available, the storage will be dedicated and appropriate accounting performed. To return storage, the storage is set available in the availability table and a garbage collecting run performed. This implies dynamic relocation of buffer areas and consequent programming overhead.

External Procedure Referenced: PROCESS DISPATCHER, OUTPUT INITIATOR, performance statistics processes, applications process, executive process

External Data Referenced: Global data, Storage Availability tables, Message Definition tables

FUNCTIONAL PROCEDURE DESCRIPTION

Procedure Identifier: **INPUT INITIATOR (NPTNTT)**

Purpose: **The Input Initiator has responsibility for allocating input buffers for message receipt. Dedicated buffers are assumed for message originating at the Keyboard Input Devices or from the Uplink. The message header must specify additional buffers required.**

Approach: **The Input Initiator operates in conjunction with the data bus controller (DBC). Message control words are appended to the control word list if the DBC is active. If the DBC is inactive, it must be activated.**

External Procedure Referenced: **Data Storage Allocator, Data Bus Control Processes**

External Data Referenced: **Global data**

FUNCTIONAL PROCEDURE DESCRIPTION

Procedure Identifier: INPUT CONTINUATOR (NPTCNT)

Purpose: The Input Continuator handles programming considerations associated with the end of message signal from the data bus controller. Hardware faults, if present, are analyzed. A hard failure will cause execution of hardware diagnostics. Reconfiguration will be invoked, if required. Conversely, if the message is error free, the classification will be derived and the correct processes activated. In addition, the watch-dog timer must be reset.

Approach: The Input Continuator assumes that the data bus controller will handle receipt of message only signaling either the end of message or an intermediate error occurrence. Message input has been started by the Input Initiator acting in conjunction with the data bus controller.

External Procedure Referenced: Hardware diagnostics, Output Initiator, Peripheral Reconfiguration, Data Storage Allocator

External Data Referenced: Global data

FUNCTIONAL PROCEDURE DESCRIPTION

Procedure Identifier: OUTPUT INITIATOR (TPTNTR)

Purpose: The OUTPUT INITIATOR (OI) performs call sequence checking, parameter passing, message formatting, request initiation and optional return selection for messages output to peripheral devices. The IO operates in conjunction with the data bus controller to accomplish message output.

Approach: After parameter checking, the OI gets the Device ID from the call sequence and searches the peripheral configuration table for the device internal hardware address. This address and other pertinent parameters are formatted into the message to be sent. Execution control words are constructed and transferred to the data bus controller. If the data bus controller is not active, it must be initiated. If an immediate return is requested, the OI branches to the applications process; otherwise, the return is to EXECB of the PROCESS DISPATCHER.

External Procedure Referenced: PROCESS DISPATCHER, applications process

External Data Referenced: Peripheral Configuration table, Execution Control words, Global data

FUNCTIONAL PROCEDURE DESCRIPTION

Procedure Identifier: OUTPUT CONTINUATOR (TPTCTR)

Purpose: The OUTPUT CONTINUATOR (OC) handles programming considerations associated with a message complete signal from the data bus controller. Transient hardware failures are separated from "solid" failures and reconfiguration invoked, if required. Notification messages are initiated and the Device Failure table updated. When preceding status has been updated, the OC transfers required data to the performance statistics gathering processes.

Approach: Upon receipt of the message complete signal, the OC process resets the watch-dog timer. Subsequently, the output status word is fetched and examined for error conditions. In the event of error conditions, transient failures are separated from hard failures by repeated attempted transmissions. If a hard failure is detected, the hardware diagnostics are invoked and the PERIPHERAL RECONFIGURATION called, if required. Notification messages are output as required and the optional return to the applications process or to EXECB of the PROCESS DISPATCHER taken.

External Procedure Referenced: PROCESS DISPATCHER, OUTPUT INITIATOR, hardware diagnostics, performance statistics gathering processes

External Data Referenced: Device Failure tables, Global data, output status word

FUNCTIONAL PROCEDURE DESCRIPTION

Procedure Identifier: ANALOG INPUT INITIATOR (NLGNNR)

Purpose: ANALOG INPUT INITIATOR (AII) provides initialization for input of the various scan classes of analog devices. AII is activated periodically to examine scan classes for criticality. Data referring to active scan classes is transferred to the ANALOG INPUT CONTINUATOR (AIC) and INPUT/OUTPUT INITIATORS for action.

Approach: After decrement of Execution Delta T's, the EXAM routine checks scan class intervals (among others) for criticality. Detecting an active scan class causes the AII to be waked. The AII in turn performs programming initialization necessary to cause input of the analog devices using the INPUT/OUTPUT INITIATOR/CONTINUATOR's and the ANALOG SCAN CONTINUATOR.

External Procedure Referenced: DELAY, PROCESS DISPATCHER, OUTPUT INITIATOR, ANALOG SCAN CONTINUATOR

External Data Referenced: Global data, Peripheral Device Configuration tables, SCAN Result tables, Analog ID tables

FUNCTIONAL PROCEDURE DESCRIPTION

Procedure Identifier: ANALOG INPUT CONTINUATOR (NLGNCR)

Purpose: The ANALOG INPUT CONTINUATOR (AIC) handles programming considerations associated with input of a group of analog sensors from each area multiplexor. The AIC functions in conjunction with the data bus controller to input the sensors that have been scanned by the ANALOG INPUT INITIATOR.

Approach: Upon receipt of the scan data ready signal, the AIC gets the active scan class data, allocates input buffers, if required, and prepares control words for the INPUT INITIATOR (II) and initiates the II. When the end of message signal is received, the AIC performs limit checking, alarming and units conversion, as required, and stores results in the Analog Scan Results tables.

External Procedure Referenced: PROCESS DISPATCHER, DATA STORAGE ALLOCATOR, INPUT INITIATOR, OUTPUT INITIATOR ALARM, ENGINEERING UNITS CONVERSION

External Data Referenced: Global data, Analog Input data, Analog Scan Results tables, input buffers

FUNCTIONAL PROCEDURE DESCRIPTION

Procedure Identifier: ANALOG OUTPUT INITIATOR (NLGOPI)

Purpose: The Analog Output Initiator (AOI) provides programming interface for applications process output of voltages or currents. The AOI operates in conjunction with the data bus controller (DBC). Call sequence errors are logged using the Output Initiator. Return is to EXECB of the Process Dispatcher or to the applications process.

Approach: Call sequence parameters are checked and passed. Based on these parameters, engineering units to digital counts conversion is accomplished for the output device. The control word is then constructed, message formatted, and output accomplished via the data bus controller (Output Initiator). Optional immediate return to application process or jump to EXECB of Process Dispatcher is provided.

External Procedure Referenced: Output Initiator, Process Dispatches, Application process

External Data Referenced: Global data, Peripheral Configuration tables, Data Bus Controller execution list

FUNCTIONAL PROCEDURE DESCRIPTION

Procedure Identifier: ANALOG OUTPUT CONTINUATOR (NLGOPT)

Purpose: The Analog Output Continuator (AOC) provides optional delayed return to the applications process or to EXECB of the Process Dispatcher. In addition, AOC resets the watch-dog timer and performs error analysis functions on the output request. Upon an error preventing completion of the output, logging of the anomaly is done and return executed through the call sequence error return.

Approach: After resetting the watch-dog timer, the AOC retrieves the output status word for the analog output. In case of the error situation described above, the error message is formatted and output using the Output Initiator. Return is selected from the call sequence to either EXECB of the Process Dispatcher or delayed return address of the applications process.

External Procedure Referenced: Process Dispatcher, Analog Output Initiator, Applications process

External Data Referenced: Global data, Output status word, Peripheral Configuration table

FUNCTIONAL PROCEDURE DESCRIPTION

Procedure Identifier: DIGITAL OUTPUT INITIATOR (DGTLTP)

Purpose: DIGITAL OUTPUT provides an interface to applications process for output of digital data to peripheral devices. Such data may be in the form of timed contact actions, momentary contact action, pulse information or such hardware manipulation as may be defined. DIGITAL OUTPUT operates with the data bus controller (DBC) using OUTPUT INITIATOR.

Approach: Using the device ID from the call sequence, the device interval address is found in the Peripheral Configuration tables. Using the device address and the output designation, a message is formatted for the OUTPUT INITIATOR. If the DBC is active, the execution control words are appended to the list for the DBC. If the DBC is not active, it must be initiated. Option of an immediate return to the applications process or a jump to EXECB of PROCESS DISPATCHER is provided.

External Procedure Referenced: PROCESS DISPATCHER, OUTPUT INITIATOR, applications process

External Data Referenced: Peripheral Configuration table, Digital status table, Global data

FUNCTIONAL PROCEDURE DESCRIPTION

Procedure Identifier: **DIGITAL OUTPUT CONTINUATOR (DGTLCR)**

Purpose: **The DIGITAL OUTPUT CONTINUATOR provides a check to determine error status that prevented successful output completion. Upon detection of such status, the fault is logged using the OUTPUT INITIATOR. Optional delayed return to the applications process or the EXECB of the PROCESS DISPATCHER are also provided.**

Approach: **Upon receipt of the output complete signal, the DIGITAL OUTPUT CONTINUATOR analyzes the output status word for errors. Detection of these errors causes action described above. Additionally, the call sequence is examined for return action required. Action taken is described above.**

External Procedure Referenced: **PROCESS DISPATCHER, OUTPUT INITIATOR**

External Data Referenced: **Global data, output status word**

FUNCTIONAL PROCEDURE DESCRIPTION

Procedure Identifier: MASS STORE INITIATOR (MSSRNT)

Purpose: The Mass Store Initiator (MSI) interprets the call sequence to determine operation required (Read, Write, Write end of file, Sense end of file, Rewind, Backspace). Acting in conjunction with the data bus controller (DBC), the MSI programs the initial conditions for the mass store operation.

Approach: The MSI formats a message composed of the selected information to manipulate the mass storage device. If the data bus controller is active, the control message pointer is appended to the execution list for the DBC and appropriate housekeeping functions associated with mass store manipulation and performance measurement are executed. If the DBC is not active, it must be initialized.

External Procedure Referenced: OUTPUT INITIATOR, PROCESS DISPATCHER, Applications process error return

External Data Referenced: DBC Item counter, Global data

FUNCTIONAL PROCEDURE DESCRIPTION

Procedure Identifier: MASS STORE CONTINUATOR (MSSSTR)

Purpose: The Mass Store Contimuator (MSC) handles programming considerations associated with an end of message signal from the data bus controller (DBC). Error conditions preventing completion are logged via the OUTPUT INITIATOR.

Approach: The MSC checks the data transfer status word to determine whether or not an error has occurred that prevented completion of the request. If, in fact, this has happened, the occurrence is logged and the error return in the call sequence is taken. If the request has been acted upon, the MSC determines whether further data transfers are required. New requests are initiated until the call sequence is fulfilled.

External Procedure Referenced: OUTPUT INITIATOR, PROCESS DISPATCHER

External Data Referenced: Global data, data transfer status word, Peripheral Configuration table

FUNCTIONAL PROCEDURE DESCRIPTION

Procedure Identifier: TIMELINE INTERPRETER/CONTROLLER (TMLNNT)

Purpose: The TIMELINE INTERPRETER/CONTROLLER (TIC) converts a sequence of timeline activities into a process schedule for execution.

Approach: TIC is entered periodically, the timeline status analyzed and compared with scheduled status. Discrepancies are alarmed, timeline history updated. A check is then made to determine whether or not all processes for a mission phase have been scheduled and if so, whether or not the phase is complete. A mission phase complete override may be established here. If all phases are done, then DELAY is entered. If all processes are not scheduled, they are scheduled as allowable. If a phase is complete, next phase initialization is accomplished. Timeline history is updated, performance statistic processes are entered and the usual option of returns taken.

External Procedure Referenced: PROCESS DISPATCHER, performance gather processes, DELAY, TURNON, OUTPUT INITIATOR

External Data Referenced: Global data, timeline status data

FUNCTIONAL PROCEDURE DESCRIPTION

Procedure Identifier: **PERIPHERAL RECONFIGURATION (PRLRCF)**

Purpose: **The PERIPHERAL RECONFIGURATION (PR) process selects a good peripheral device, if possible, for input/output (I/O) operations. Operating in conjunction with the hardware diagnostic routines (ST) and the INPUT/OUTPUT INITIATOR/CONTINUATOR's, the PR examines the Device Failure tables and the Peripheral Configuration tables to construct a select word for a valid device for an arbitrary operation.**

Approach: **Upon detection of a hard failure in a peripheral device, the PR process is waked. By examining the Device Failure tables and the Peripheral Configuration tables the PR process selects a valid alternate device. The select control word for this valid device is substituted for the failing device in the Device ID Selection tables. Therefore, processes attempting to use this failed device will instead use the substituted device.**

External Procedure Referenced: OUTPUT INITIATOR, PROCESS DISPATCHER, HARDWARE DIAGNOSTICS

External Data Referenced: Peripheral Configuration tables, Device ID Selection tables, Global data

FUNCTIONAL PROCEDURE DESCRIPTION

Procedure Identifier: CPU RECONFIGURATION (CPRCNF)

Purpose: The CPRCNF process directs the System Control Unit (SCU) in the detection of anomalies in the A&B computer operation. This is done by analysis of status signals and interrupts sent from the computers to the SCU and control signals sent from the SCU to the computers. The CPRCNF analyzes VOTE, SYNC and error notifications, directing the computers in reconfiguration sequences, if required.

Approach: The CPRCNF has several entries: periodic, vote, sync, watch-dog timer, and error. Statistics concerning system status are collated using stacking procedures. Logical procedures detect anomalies and reconfiguration sequences are initiated. For SYNC and VOTE requests, the watch-dog timer is set and status lists are periodically examined to detect vote and/or sync state. In the case of SYNC, the computers are released when in synchronous ignition or error analysis procedures initiated. For the VOTE case, voting results are analyzed and reconfiguration sequences initiated when required.

External Procedure Referenced: RLSPRC, reconfiguration sequences, statistics accumulation process, performance statistics processes, PROCESS DISPATCHER

External Data Referenced: Computer status words, Global data

Procedure Identifier: ALARM (ALARM)

Purpose: The ALARM process maintains current system alarm status, provides alarm notification as required, controls the alarm portion of the maintenance recording and provides an interface to the performance statistics gathering process.

Approach: The ALARM process is called upon detection of an anomaly. The call sequence provides Device ID, anomaly classification, The ALARM process analyzes alarm history tables, formats notification message if required, and initiates the message using the OUTPUT INITIATOR. Maintenance recording is also done, if required. Data are transmitted to performance measurement routines, as required.

External Procedure Referenced: OUTPUT INITIATOR, PERFORMANCE MEASUREMENT, MAINTENANCE RECORDING

External Data Referenced: Global data, Alarm History tables

FUNCTIONAL PROCEDURE DESCRIPTION

Procedure Identifier: DISPLAY INITIATOR/CANCEL (DSPYNR)

Purpose: The DISPLAY INITIATOR provides initial conditions for the CRT displays. Each display required will have an ordered list of processes that operate to change the initial map along with values defining the original map. The DISPLAY INITIATOR will wake these processes after selecting the output CRT and transmitting the initial map. Displays are cancelled by transmitting a cancel request using the OUTPUT INITIATOR.

Approach: Call sequence parameters yield the display, output device and other required data. The initial display is formatted into a message for the OUTPUT INITIATOR and transmitted to the output device using the data bus controller. Processes on the list associated with the display are waked. Option to return to the applications process or to EXECB are exercised. For display cancellation, the cancel request is formatted into a message and transmitted to the device using the OUTPUT INITIATOR.

External Procedure Referenced: OUTPUT INITIATOR, PROCESS DISPATCHER, DATA STORAGE ALLOCATOR, applications processor

External Data Referenced: Cancel control word, peripheral configuration tables, Global data

FUNCTIONAL PROCEDURE DESCRIPTION

Procedure Identifier: DISPLAY CONTINUATOR (DSPYCR)

Purpose: The purpose of DISPLAY CONTINUATOR is to provide display refreshment via the display refresh signal. Processes on the image manipulation list are waked and the changed image transmitted to the requested output device using the OUTPUT INITIATOR.

Approach: Upon receipt of the refresh signal the DISPLAY CONTINUATOR wakes processes on the image manipulation list. Upon completion, the new image is transmitted using the data bus controller. The message is formatted and execution control words transferred to the DBC. If the DBC is not active, it must be initiated. Exit is taken to EXECB of the PROCESS DISPATCHER.

External Procedure Referenced: PROCESS DISPATCHER, OUTPUT INITIATOR, applications processes

External Data Referenced: Global data, execution control word, Peripheral Configuration table

FUNCTIONAL PROCEDURE DESCRIPTION

Procedure Identifier: TIMELINE DISPLAY (TMLNDS)

Purpose: TIMELINE DISPLAY (TLD) will provide a procedure to display on a CRT selected portions of the mission timeline. The process will operate in conjunction with the DISPLAY/INITIATOR/CANCEL/CONTINUATOR.

Approach: The call sequence will determine the subset of the timeline sequence to display. The TLD will access the pertinent subset of the timeline and format a call to the DISPLAY INITIATOR. The TIMELINE INITIATOR will treat the display in the same way that other displays are handled. Timeline status display will be accomplished similarly to the above.

External Procedure Referenced: DISPLAY INITIATOR, PROCESS DISPATCHER, applications process

External Data Referenced: Global data

Procedure Identifier: **PANEL SCAN (PNLSCN)**

Purpose: Panel Scan provides facilities to determine the status of the panel switches. Panel Scan is activated by a change of state signal or on a periodic basis. Using the INPUT/OUTPUT INITIATORS and the Peripheral Configuration tables, the status of the devices are input and stored in the Device Status tables. Based on values of the input, arbitrary processes may be waked.

Approach: Upon detection of a change of state signal or of the process becoming time critical, the Panel Scan process is activated. Using values from the Peripheral Configuration tables, messages are prepared to select and input the contents of the registers associated with the various switch positions. These values are converted as required and processes waked as necessary.

External Procedure Referenced: INPUT INITIATOR, OUTPUT INITIATOR, PROCESS DISPATCHER, DELAY, TURN ON, Applications processes

External Data Referenced: Global data, Peripheral Configuration tables

FUNCTIONAL PROCEDURE DESCRIPTION

Procedure Identifier: DIGITAL INPUT (DGTLNP)

Purpose: The Digital Input Routine (DIR) has responsibility for reading the contents of digital registers and storing the values in preselected buffer areas. The DIR operates in conjunction with the data bus controller (DBC) using the INPUT INITIATOR. Error conditions are logged using the OUTPUT INITIATOR. Periodic operation is provided using DELAY.

Approach: Control words are constructed for each area multiplexor and respective digital values are input by formatting a message for the OUTPUT INITIATOR. Upon ready condition, the data are input using the INPUT INITIATOR. This is done for all area multiplexors. Execution control words must be transmitted to the DBC for output and input. If the controller is not active, it must be initiated. The time increment until next execution is passed to DELAY for periodic process running.

External Procedure Referenced: OUTPUT INITIATOR, INPUT INITIATOR, DELAY

External Data Referenced: Peripheral Configuration tables, Execution control word, Global data, Data bus controller, queue length

APPENDIX B. PROCESS ATTRIBUTE DESCRIPTIONS

1. Name. Each program is assigned a unique alphanumeric name. The name will be sufficient to locate and define the module in libraries, internal storage, or external storage and is formed by the rules for acronym definition.

2. Size estimates. Estimates are provided here for number of lines of code and space required for local data. Word lengths are assumed 32 bits. All sizes are given in decimal, and lines of code are qualified as either assembly code (ASM) or problem-oriented language (POL) code, such as FORTRAN, BASIC, etc.

3. Relative statement type. The percent of each different instruction class is estimated (logical or computational).

4. Execution condition. The frequency of execution (number of times per second) and the number of operations per execution (number of instructions executed per call).

5. Complexity. The complexity of a procedure is measured by the number of loops, number of paths, number of flow diagram blocks, and block exit density. These factors impact design intimacy, complication of coding, and thoroughness of checkout. The exit density calculation will be a valuable measure of complexity since it relates decisions and paths at module level.

a. Number of loops. The enumeration of repetitively executed sequences of instructions. Loop level is multiplied by the number of loops at each level (1, 2, ...) to account for nestedness.

b. Number of paths. The number of transitions that can be made among procedure subdivisions. This is the same as the total number of exits from all blocks.

c. Number of blocks. The total number of flow diagram blocks.

d. Block exit density. Value of (b) divided by value of (c).

6. Type. This attribute is specified by the usage. A procedure is either a simple procedure or a function.

a. Simple. This refers to a procedure that accepts input parameters and returns output parameters only through use of a formal parameter list or through global variables.

b. Function. A procedure which accepts input through a formal parameter list and returns a single output through a hardware register is a function. It may return an output that is in the machine format of an integer, real, boolean, complex, or special variable.

c. Reentrant. This is a procedure that can be called repeatedly at any stage of execution and properly complete each call. This implies that the program module may not be dynamically modified and does not store intermediate results locally.

d. Recursive. A procedure which calls itself (through the use of push-down lists) is said to be recursive. The call may be direct or indirect through another procedure. A recursive procedure is not strictly reentrant since it cannot be called at any point in its execution. It is, however, reentrant in the sense that the recursion is accomplished by repeated calls (reenters at the beginning) to itself.

7. Priority. An integer from 0 to n (where 0 is the highest priority) that indicates the estimated relative importance or urgency for execution of this procedure.

8. Development time. The estimated man-months required for total programming implementation (requirements analysis through checkout, including documentation).

9. Residence requirements. Main memory allocation requirements for procedures, transient areas, overlays, and bulk storage considerations will be estimated; an indication of storage requirements when the program is in a dormant state is included.

10. Events causing execution. An indication of what event(s) must occur in order to cause execution of the procedure is given. This should expose transition routes among procedures and show the initiating conditions and mechanisms for procedure connectivity. Examples are direct call by another procedure, indirect call by queuing, and interrupt activation. A list of the procedures that reference this procedure is given.

11. Output data description. Definition of the output guaranteed by the process. This output will be considered as used by external processes. The output must be demarcated as to range, frequency or other descriptive parameters depending on the process function. Any modification of these parameters must be investigated for system interactions.

12. Input requirements. Definition of the input data required by the process. The process, in effect, guarantees that if these input specifications are met, then the output will be as described above. Similar considerations apply.

13. Remarks. Miscellaneous notations.

Revision No. _____
Date _____

PROCESS ATTRIBUTE DESCRIPTION

1.	Name:	Process Dispatcher
2.	Size Estimates:	
	a. Code:	100
	b. Local Data:	0
3.	Relative Statement Type Percent:	
	a. Computational:	20
	b. Logical:	80
4.	Execution Condition:	
	a. Frequency:	Called on each activation of new process
	b. Number of Operations:	Variable
5.	Complexity:	
	a. Number of Loops:	2
	b. Number of Paths:	14
	c. Number of Blocks:	8
	d. Block Exit Density:	1.75
6.	Type:	
	a. Simple:	No
	b. Function:	
	(1) Integer:	No
	(2) Real:	No
	(3) Boolean:	No
	(4) Complex:	No
	(5) Special:	Yes
	c. Reentrant:	No
	(1) Recursive:	
	(2) Non-Recursive:	
	d. Non-Reentrant:	Yes
7.	Priority:	0
8.	Development Time:	TBD
9.	Residence Requirements:	100
10.	Events Causing Execution:	
	a. Periodic:	Not directly
	b. Signal or Interrupt:	Yes
	c. Queue Processing:	Possibly
	d. Direct Call:	Yes
11.	Output Data Description:	NA
12.	Input Requirements:	Call Sequence
13.	Remarks:	

Revision No. _____
Date _____

PROCESS ATTRIBUTE DESCRIPTION

1.	Name:	Clock Update
2.	Size Estimates:	
	a. Code:	140
	b. Local Data:	
3.	Relative Statement Type Percent:	
	a. Computational:	30
	b. Logical:	70
4.	Execution Condition:	
	a. Frequency:	Clock Cycle Dependent
	b. Number of Operations:	
5.	Complexity:	
	a. Number of Loops:	3
	b. Number of Paths:	16
	c. Number of Blocks:	14
	d. Block Exit Density:	1.13
6.	Type:	
	a. Simple:	No
	b. Function:	
	(1) Integer:	Yes
	(2) Real:	No
	(3) Boolean:	Yes
	(4) Complex:	No
	(5) Special:	Yes
	c. Reentrant:	No
	(1) Recursive:	
	(2) Non-Recursive:	
	d. Non-Reentrant:	Yes
7.	Priority:	0
8.	Development Time:	TBD
9.	Residence Requirements:	140
10.	Events Causing Execution:	
	a. Periodic:	Yes
	b. Signal or Interrupt:	Yes
	c. Queue Processing:	No
	d. Direct Call:	No
11.	Output Data Description:	NA
12.	Input Requirements:	NA
13.	Remarks:	

Dependent on requirements to maintain software clocks and watch-dog timers.

Revision No. _____
Date _____

PROCESS ATTRIBUTE DESCRIPTION

1.	Name:	Examine Active List for Critical Process
2.	Size Estimates:	
	a. Code:	200
	b. Local Data:	
3.	Relative Statement Type Percent:	
	a. Computational:	30
	b. Logical:	70
4.	Execution Condition:	
	a. Frequency:	Entered after Clock Interrupt
	b. Number of Operations:	200
5.	Complexity:	
	a. Number of Loops:	2
	b. Number of Paths:	21
	c. Number of Blocks:	19
	d. Block Exit Density:	1.01
6.	Type:	
	a. Simple:	No
	b. Function:	
	(1) Integer:	No
	(2) Real:	No
	(3) Boolean:	No
	(4) Complex:	No
	(5) Special:	Yes
	c. Reentrant:	Partially
	(1) Recursive:	
	(2) Non-Recursive:	
	d. Non-Reentrant:	Yes
7.	Priority:	0
8.	Development Time:	TBD
9.	Residence Requirements:	200
10.	Events Causing Execution:	
	a. Periodic:	Partially
	b. Signal or Interrupt:	No
	c. Queue Processing:	No
	d. Direct Call:	Yes
11.	Output Data Description:	NA
12.	Input Requirements:	NA
13.	Remarks:	

Revision No. _____
Date _____

PROCESS ATTRIBUTE DESCRIPTION

1.	Name:	TURNON
2.	Size Estimates:	
	a. Code:	130
	b. Local Data:	10
3.	Relative Statement Type Percent:	
	a. Computational:	30
	b. Logical:	70
4.	Execution Condition:	
	a. Frequency:	System loading
	b. Number of Operations:	100
5.	Complexity:	
	a. Number of Loops:	2
	b. Number of Paths:	16
	c. Number of Blocks:	11
	d. Block Exit Density:	1.45
6.	Type:	
	a. Simple:	Yes
	b. Function:	
	(1) Integer:	Yes
	(2) Real:	No
	(3) Boolean:	No
	(4) Complex:	No
	(5) Special:	No
	c. Reentrant:	No
	(1) Recursive:	
	(2) Non-Recursive:	
	d. Non-Reentrant:	
7.	Priority:	0
8.	Development Time:	TBD
9.	Residence Requirements:	140
10.	Events Causing Execution:	
	a. Periodic:	No
	b. Signal or Interrupt:	No
	c. Queue Processing:	No
	d. Direct Call:	Yes
11.	Output Data Description:	NA
12.	Input Requirements:	Call Sequence
13.	Remarks:	

Revision No. _____
Date _____

PROCESS ATTRIBUTE DESCRIPTION

1.	Name:	TERMINATE A PROCESS
2.	Size Estimates:	
	a. Code:	140
	b. Local Data:	0
3.	Relative Statement Type Percent:	
	a. Computational:	10
	b. Logical:	90
4.	Execution Condition:	
	a. Frequency:	System Load
	b. Number of Operations:	Variable
5.	Complexity:	
	a. Number of Loops:	2
	b. Number of Paths:	8
	c. Number of Blocks:	6
	d. Block Exit Density:	1.33
6.	Type:	
	a. Simple:	Yes
	b. Function:	
	(1) Integer:	Yes
	(2) Real:	No
	(3) Boolean:	No
	(4) Complex:	No
	(5) Special:	No
	c. Reentrant:	No
	(1) Recursive:	No
	(2) Non-Recursive:	
	d. Non-Reentrant:	Yes
7.	Priority:	0
8.	Development Time:	TBD
9.	Residence Requirements:	140
10.	Events Causing Execution:	
	a. Periodic:	No
	b. Signal or Interrupt:	No
	c. Queue Processing:	No
	d. Direct Call:	Yes
11.	Output Data Description:	NA
12.	Input Requirements:	Call Sequence
13.	Remarks:	

Revision No. _____
Date _____

PROCESS ATTRIBUTE DESCRIPTION

1.	Name:	DELAY A PROCESS
2.	Size Estimates:	
	a. Code:	90
	b. Local Data:	10
3.	Relative Statement Type Percent:	
	a. Computational:	20
	b. Logical:	80
4.	Execution Condition:	
	a. Frequency:	Used for periodic process execution
	b. Number of Operations:	Variable
5.	Complexity:	
	a. Number of Loops:	2
	b. Number of Paths:	8
	c. Number of Blocks:	7
	d. Block Exit Density:	1.05
6.	Type:	
	a. Simple:	Yes
	b. Function:	
	(1) Integer:	Yes
	(2) Real:	No
	(3) Boolean:	No
	(4) Complex:	No
	(5) Special:	No
	c. Reentrant:	No
	(1) Recursive:	
	(2) Non-Recursive:	
	d. Non-Reentrant:	Yes
7.	Priority:	0
8.	Development Time:	TBD
9.	Residence Requirements:	100
10.	Events Causing Execution:	
	a. Periodic:	No
	b. Signal or Interrupt:	No
	c. Queue Processing:	No
	d. Direct Call:	Yes
11.	Output Data Description:	NA
12.	Input Requirements:	Call Sequence
13.	Remarks:	

Revision No. _____
Date _____

PROCESS ATTRIBUTE DESCRIPTION

1.	Name:	SUSPEND A PROCESS
2.	Size Estimates:	
	a. Code:	90
	b. Local Data:	10
3.	Relative Statement Type Percent:	
	a. Computational:	10
	b. Logical:	90
4.	Execution Condition:	
	a. Frequency:	Low
	b. Number of Operations:	Variable
5.	Complexity:	
	a. Number of Loops:	2
	b. Number of Paths:	6
	c. Number of Blocks:	6
	d. Block Exit Density:	1.0
6.	Type:	
	a. Simple:	Yes
	b. Function:	
	(1) Integer:	Yes
	(2) Real:	No
	(3) Boolean:	No
	(4) Complex:	No
	(5) Special:	No
	c. Reentrant:	No
	(1) Recursive:	
	(2) Non-Recursive:	
	d. Non-Reentrant:	Yes
7.	Priority:	0
8.	Development Time:	TBD
9.	Residence Requirements:	100
10.	Events Causing Execution:	
	a. Periodic:	No
	b. Signal or Interrupt:	No
	c. Queue Processing:	No
	d. Direct Call:	Yes
11.	Output Data Description:	NA
12.	Input Requirements:	Call Sequence
13.	Remarks:	Planned usage for diagnostic applications

Revision No. _____
Date _____

PROCESS ATTRIBUTE DESCRIPTION

1.	Name:	RELEASE A PROCESS
2.	Size Estimates:	
	a. Code:	90
	b. Local Data:	10
3.	Relative Statement Type Percent:	
	a. Computational:	10
	b. Logical:	10
4.	Execution Condition:	
	a. Frequency:	System Load
	b. Number of Operations:	70
5.	Complexity:	
	a. Number of Loops:	2
	b. Number of Paths:	4
	c. Number of Blocks:	4
	d. Block Exit Density:	1.0
6.	Type:	
	a. Simple:	Yes
	b. Function:	
	(1) Integer:	No
	(2) Real:	No
	(3) Boolean:	Yes
	(4) Complex:	No
	(5) Special:	No
	c. Reentrant:	No
	(1) Recursive:	
	(2) Non-Recursive:	
	d. Non-Reentrant:	Yes
7.	Priority:	0
8.	Development Time:	TBD
9.	Residence Requirements:	100
10.	Events Causing Execution:	
	a. Periodic:	No
	b. Signal or Interrupt:	No
	c. Queue Processing:	No
	d. Direct Call:	Yes
11.	Output Data Description:	NA
12.	Input Requirements:	Call Sequence
13.	Remarks:	

Revision No. _____
Date _____

PROCESS ATTRIBUTE DESCRIPTION

1.	Name:	VOTE
2.	Size Estimates:	
	a. Code:	140
	b. Local Data:	10
3.	Relative Statement Type Percent:	
	a. Computational:	30
	b. Logical:	70
4.	Execution Condition:	
	a. Frequency:	System Load
	b. Number of Operations:	70
5.	Complexity:	
	a. Number of Loops:	3
	b. Number of Paths:	26
	c. Number of Blocks:	22
	d. Block Exit Density:	1.2
6.	Type:	
	a. Simple:	
	b. Function:	
	(1) Integer:	Yes
	(2) Real:	No
	(3) Boolean:	No
	(4) Complex:	No
	(5) Special:	Yes
	c. Reentrant:	No
	(1) Recursive:	
	(2) Non-Recursive:	
	d. Non-Reentrant:	Yes
7.	Priority:	0
8.	Development Time:	TBD
9.	Residence Requirements:	150
10.	Events Causing Execution:	
	a. Periodic:	Not directly
	b. Signal or Interrupt:	No
	c. Queue Processing:	No
	d. Direct Call:	Yes
11.	Output Data Description:	NA
12.	Input Requirements:	Call Sequence
13.	Remarks:	

Revision No. _____
Date _____

PROCESS ATTRIBUTE DESCRIPTION

1.	Name:	SYNC
2.	Size Estimates:	
	a. Code:	70
	b. Local Data:	10
3.	Relative Statement Type Percent:	
	a. Computational:	5
	b. Logical:	95
4.	Execution Condition:	
	a. Frequency:	System Load
	b. Number of Operations:	30
5.	Complexity:	
	a. Number of Loops:	3
	b. Number of Paths:	11
	c. Number of Blocks:	9
	d. Block Exit Density:	1.22
6.	Type:	
	a. Simple:	Yes
	b. Function:	
	(1) Integer:	No
	(2) Real:	No
	(3) Boolean:	No
	(4) Complex:	No
	(5) Special:	Yes
	c. Reentrant:	No
	(1) Recursive:	
	(2) Non-Recursive:	
	d. Non-Reentrant:	Yes
7.	Priority:	0
8.	Development Time:	TBD
9.	Residence Requirements:	80
10.	Events Causing Execution:	
	a. Periodic:	Not directly
	b. Signal or Interrupt:	No
	c. Queue Processing:	No
	d. Direct Call:	Yes
11.	Output Data Description:	NA
12.	Input Requirements:	Call Sequence
13.	Remarks:	

Revision No. _____
Date _____

PROCESS ATTRIBUTE DESCRIPTION

1.	Name:	Data Storage Allocator
2.	Size Estimates:	
	a. Code:	600
	b. Local Data:	10
3.	Relative Statement Type Percent:	
	a. Computational:	20
	b. Logical:	80
4.	Execution Condition:	
	a. Frequency:	System Load
	b. Number of Operations:	Variable
5.	Complexity:	
	a. Number of Loops:	TBD
	b. Number of Paths:	TBD
	c. Number of Blocks:	TBD
	d. Block Exit Density:	TBD
6.	Type:	
	a. Simple:	Yes
	b. Function:	
	(1) Integer:	No
	(2) Real:	No
	(3) Boolean:	No
	(4) Complex:	No
	(5) Special:	Yes
	c. Reentrant:	No
	(1) Recursive:	
	(2) Non-Recursive:	
	d. Non-Reentrant:	Yes
7.	Priority:	0
8.	Development Time:	TBD
9.	Residence Requirements:	610
10.	Events Causing Execution:	
	a. Periodic:	No
	b. Signal or Interrupt:	No
	c. Queue Processing:	No
	d. Direct Call:	Yes
11.	Output Data Description:	NA
12.	Input Requirements:	Call Sequence
13.	Remarks:	

Revision No. _____
Date _____

PROCESS ATTRIBUTE DESCRIPTION

1.	Name:	Input Initiator
2.	Size Estimates:	
	a. Code:	350
	b. Local Data:	30
3.	Relative Statement Type Percent:	
	a. Computational:	10
	b. Logical:	90
4.	Execution Condition:	
	a. Frequency:	Function of System Load
	b. Number of Operations:	Variable
5.	Complexity:	
	a. Number of Loops:	4
	b. Number of Paths:	24
	c. Number of Blocks:	20
	d. Block Exit Density:	1.20
6.	Type:	
	a. Simple:	Yes
	b. Function:	
	(1) Integer:	No
	(2) Real:	No
	(3) Boolean:	No
	(4) Complex:	No
	(5) Special:	Yes
	c. Reentrant:	No
	(1) Recursive:	
	(2) Non-Recursive:	
	d. Non-Reentrant:	Yes
7.	Priority:	3
8.	Development Time:	TBD
9.	Residence Requirements:	380
10.	Events Causing Execution:	
	a. Periodic:	No
	b. Signal or Interrupt:	No
	c. Queue Processing:	No
	d. Direct Call:	Yes
11.	Output Data Description:	NA
12.	Input Requirements:	Call Sequence, interface
13.	Remarks:	

Revision No. _____
Date _____

PROCESS ATTRIBUTE DESCRIPTION

1.	Name:	Input Terminator
2.	Size Estimates:	
	a. Code:	750
	b. Local Data:	0
3.	Relative Statement Type Percent:	
	a. Computational:	10
	b. Logical:	90
4.	Execution Condition:	
	a. Frequency:	Function of System Load
	b. Number of Operations:	Variable
5.	Complexity:	
	a. Number of Loops:	20
	b. Number of Paths:	130
	c. Number of Blocks:	60
	d. Block Exit Density:	2.15
6.	Type:	
	a. Simple:	Yes
	b. Function:	
	(1) Integer:	No
	(2) Real:	No
	(3) Boolean:	No
	(4) Complex:	No
	(5) Special:	Yes
	c. Reentrant:	No
	(1) Recursive:	
	(2) Non-Recursive:	
	d. Non-Reentrant:	Yes
7.	Priority:	3
8.	Development Time:	TBD
9.	Residence Requirements:	750
10.	Events Causing Execution:	
	a. Periodic:	No
	b. Signal or Interrupt:	Yes
	c. Queue Processing:	No
	d. Direct Call:	No
11.	Output Data Description:	Interface Requirements
12.	Input Requirements:	
13.	Remarks:	

Date _____

PROCESS ATTRIBUTE DESCRIPTION

1.	Name:	Output Initiator
2.	Size Estimates:	
	a. Code:	300
	b. Local Data:	60
3.	Relative Statement Type Percent:	
	a. Computational:	30
	b. Logical:	70
4.	Execution Condition:	
	a. Frequency:	System Load
	b. Number of Operations:	
5.	Complexity:	
	a. Number of Loops:	4
	b. Number of Paths:	13
	c. Number of Blocks:	11
	d. Block Exit Density:	1.2
6.	Type:	
	a. Simple:	Yes
	b. Function:	
	(1) Integer:	No
	(2) Real:	No
	(3) Boolean:	No
	(4) Complex:	No
	(5) Special:	Yes
	c. Reentrant:	No
	(1) Recursive:	
	(2) Non-Recursive:	
	d. Non-Reentrant:	Yes
7.	Priority:	3
8.	Development Time:	TBD
9.	Residence Requirements:	360
10.	Events Causing Execution:	
	a. Periodic:	No
	b. Signal or Interrupt:	Possibly indirectly
	c. Queue Processing:	No
	d. Direct Call:	Yes
11.	Output Data Description:	Device & Data Bus Controller Output Rqmts.
12.	Input Requirements:	Call Sequence
13.	Remarks:	

Revision No. _____
Date _____

PROCESS ATTRIBUTE DESCRIPTION

1.	Name:	Output Terminator
2.	Size Estimates:	
	a. Code:	700
	b. Local Data:	0
3.	Relative Statement Type Percent:	
	a. Computational:	20
	b. Logical:	80
4.	Execution Condition:	
	a. Frequency:	System Load
	b. Number of Operations:	Variable
5.	Complexity:	
	a. Number of Loops:	4
	b. Number of Paths:	23
	c. Number of Blocks:	15
	d. Block Exit Density:	1.56
6.	Type:	
	a. Simple:	Yes
	b. Function:	
	(1) Integer:	No
	(2) Real:	No
	(3) Boolean:	No
	(4) Complex:	No
	(5) Special:	Yes
	c. Reentrant:	No
	(1) Recursive:	
	(2) Non-Recursive:	
	d. Non-Reentrant:	Yes
7.	Priority:	3
8.	Development Time:	TBD
9.	Residence Requirements:	700
10.	Events Causing Execution:	
	a. Periodic:	No
	b. Signal or Interrupt:	Yes
	c. Queue Processing:	No
	d. Direct Call:	No
11.	Output Data Description:	Output Device Requirements
12.	Input Requirements:	NA
13.	Remarks:	

Revision No. _____
Date _____

PROCESS ATTRIBUTE DESCRIPTION

1.	Name:	Analog Input Initiator
2.	Size Estimates:	
	a. Code:	800
	b. Local Data:	0
3.	Relative Statement Type Percent:	
	a. Computational:	20
	b. Logical:	80
4.	Execution Condition:	
	a. Frequency:	Scan Class Execution Rate
	b. Number of Operations:	
5.	Complexity:	
	a. Number of Loops:	15
	b. Number of Paths:	110
	c. Number of Blocks:	40
	d. Block Exit Density:	2.75
6.	Type:	
	a. Simple:	
	b. Function:	
	(1) Integer:	
	(2) Real:	
	(3) Boolean:	
	(4) Complex:	
	(5) Special:	
	c. Reentrant:	
	(1) Recursive:	
	(2) Non-Recursive:	
	d. Non-Reentrant:	
7.	Priority:	2
8.	Development Time:	TBD
9.	Residence Requirements:	800
10.	Events Causing Execution:	
	a. Periodic:	Yes
	b. Signal or Interrupt:	No
	c. Queue Processing:	No
	d. Direct Call:	Possibly
11.	Output Data Description:	Peripheral Device Requirements
12.	Input Requirements:	NA
13.	Remarks:	Functions may be done by hardware

(5) based on similar applications

Revision No. _____
Date _____

PROCESS ATTRIBUTE DESCRIPTION

1.	Name:	Analog Input Continuator
2.	Size Estimates:	
	a. Code:	1500
	b. Local Data:	0
3.	Relative Statement Type Percent:	
	a. Computational:	20
	b. Logical:	80
4.	Execution Condition:	
	a. Frequency:	Scan Class Execution Rate
	b. Number of Operations:	
5.	Complexity:	
	a. Number of Loops:	20
	b. Number of Paths:	150
	c. Number of Blocks:	100
	d. Block Exit Density:	1.50
6.	Type:	
	a. Simple:	No
	b. Function:	
	(1) Integer:	No
	(2) Real:	No
	(3) Boolean:	No
	(4) Complex:	No
	(5) Special:	Yes
	c. Reentrant:	No
	(1) Recursive:	
	(2) Non-Recursive:	
	d. Non-Reentrant:	Yes
7.	Priority:	3
8.	Development Time:	TBD
9.	Residence Requirements:	1500
10.	Events Causing Execution	
	a. Periodic:	Possibly
	b. Signal or Interrupt:	Possibly
	c. Queue Processing:	No
	d. Direct Call:	No
11.	Output Data Description:	Peripheral Device Requirements
12.	Input Requirements:	NA
13.	Remarks:	Functions may be hardware implemented

(5) based on similar applications

Revision No. _____
Date _____

PROCESS ATTRIBUTE DESCRIPTION

1.	Name:	Analog Output Initiator/Terminator
2.	Size Estimates:	
	a. Code:	350
	b. Local Data:	0
3.	Relative Statement Type Percent:	
	a. Computational:	15
	b. Logical:	85
4.	Execution Condition:	
	a. Frequency:	
	b. Number of Operations:	
5.	Complexity:	
	a. Number of Loops:	5
	b. Number of Paths:	16
	c. Number of Blocks:	10
	d. Block Exit Density:	1.60
6.	Type:	
	a. Simple:	No
	b. Function:	
	(1) Integer:	No
	(2) Real:	No
	(3) Boolean:	No
	(4) Complex:	No
	(5) Special:	Yes
	c. Reentrant:	No
	(1) Recursive:	
	(2) Non-Recursive:	
	d. Non-Reentrant:	Yes
7.	Priority:	3
8.	Development Time:	
9.	Residence Requirements:	350
10.	Events Causing Execution:	
	a. Periodic:	Possibly
	b. Signal or Interrupt:	Possibly, indirectly
	c. Queue Processing:	No
	d. Direct Call:	Yes
11.	Output Data Description:	Peripheral Device Requirements
12.	Input Requirements:	Call Sequence
13.	Remarks:	—

Revision No. _____
Date _____

PROCESS ATTRIBUTE DESCRIPTION

1.	Name:	Analog Output Continuator
2.	Size Estimates:	
	a. Code:	50
	b. Local Data:	
3.	Relative Statement Type Percent:	
	a. Computational:	10
	b. Logical:	90
4.	Execution Condition:	
	a. Frequency:	System Load
	b. Number of Operations:	TBD
5.	Complexity:	TBD
	a. Number of Loops:	TBD
	b. Number of Paths:	TBD
	c. Number of Blocks:	TBD
	d. Block Exit Density:	TBD
6.	Type:	TBD
	a. Simple:	TBD
	b. Function:	TBD
	(1) Integer:	TBD
	(2) Real:	TBD
	(3) Boolean:	TBD
	(4) Complex:	TBD
	(5) Special:	TBD
	c. Reentrant:	TBD
	(1) Recursive:	TBD
	(2) Non-Recursive:	TBD
	d. Non-Reentrant:	TBD
7.	Priority:	0
8.	Development Time:	TBD
9.	Residence Requirements:	50
10.	Events Causing Execution:	
	a. Periodic:	No
	b. Signal or Interrupt:	Yes
	c. Queue Processing:	No
	d. Direct Call:	No
11.	Output Data Description:	Peripheral Device Requirements
12.	Input Requirements:	As Above
13.	Remarks:	

Revision No. _____
Date _____

PROCESS ATTRIBUTE DESCRIPTION

1.	Name:	Digital Output Initiator
2.	Size Estimates:	
	a. Code:	100
	b. Local Data:	20
3.	Relative Statement Type Percent:	
	a. Computational:	20
	b. Logical:	80
4.	Execution Condition:	
	a. Frequency:	Periodic
	b. Number of Operations:	
5.	Complexity:	
	a. Number of Loops:	4
	b. Number of Paths:	14
	c. Number of Blocks:	11
	d. Block Exit Density:	1.27
6.	Type:	
	a. Simple:	No
	b. Function:	
	(1) Integer:	No
	(2) Real:	No
	(3) Boolean:	No
	(4) Complex:	No
	(5) Special:	Yes
	c. Reentrant:	No
	(1) Recursive:	
	(2) Non-Recursive:	
	d. Non-Reentrant:	Yes
7.	Priority:	3
8.	Development Time:	TBD
9.	Residence Requirements:	120
10.	Events Causing Execution:	
	a. Periodic:	Possibly
	b. Signal or Interrupt:	No
	c. Queue Processing:	No
	d. Direct Call:	Yes
11.	Output Data Description:	Peripheral Device Requirements
12.	Input Requirements:	Call Sequence
13.	Remarks:	

Revision No. _____
Date _____

PROCESS ATTRIBUTE DESCRIPTION

1.	Name:	Digital Output Continuator
2.	Size Estimates:	
	a. Code:	50
	b. Local Data:	0
3.	Relative Statement Type Percent:	
	a. Computational:	10
	b. Logical:	90
4.	Execution Condition:	
	a. Frequency:	System Load
	b. Number of Operations:	Variable
5.	Complexity:	
	a. Number of Loops:	4
	b. Number of Paths:	6
	c. Number of Blocks:	3
	d. Block Exit Density:	2.0
6.	Type:	
	a. Simple:	No
	b. Function:	
	(1) Integer:	No
	(2) Real:	No
	(3) Boolean:	No
	(4) Complex:	No
	(5) Special:	Yes
	c. Reentrant:	No
	(1) Recursive:	
	(2) Non-Recursive:	
	d. Non-Reentrant:	Yes
7.	Priority:	3
8.	Development Time:	TBD
9.	Residence Requirements:	50
10.	Events Causing Execution:	
	a. Periodic:	No
	b. Signal or Interrupt:	Yes
	c. Queue Processing:	No
	d. Direct Call:	No
11.	Output Data Description:	Peripheral Device Requirements
12.	Input Requirements:	Same as Above
13.	Remarks:	

Revision No. _____
Date _____

PROCESS ATTRIBUTE DESCRIPTION

1.	Name:	Mass Store Initiator
2.	Size Estimates:	
	a. Code:	100
	b. Local Data:	
3.	Relative Statement Type Percent:	
	a. Computational:	15
	b. Logical:	85
4.	Execution Condition:	
	a. Frequency:	Phase data, error recording
	b. Number of Operations:	Variable
5.	Complexity:	
	a. Number of Loops:	5
	b. Number of Paths:	14
	c. Number of Blocks:	10
	d. Block Exit Density:	1.40
6.	Type:	
	a. Simple:	No
	b. Function:	
	(1) Integer:	No
	(2) Real:	No
	(3) Boolean:	No
	(4) Complex:	No
	(5) Special:	Yes
	c. Reentrant:	No
	(1) Recursive:	
	(2) Non-Recursive:	
	d. Non-Reentrant:	Yes
7.	Priority:	3
8.	Development Time:	TBD
9.	Residence Requirements:	100
10.	Events Causing Execution:	
	a. Periodic:	No
	b. Signal or Interrupt:	No
	c. Queue Processing:	No, possible
	d. Direct Call:	Yes
11.	Output Data Description:	Device Requirements
12.	Input Requirements:	Call Sequence
13.	Remarks:	

Revision No. _____
Date _____

PROCESS ATTRIBUTE DESCRIPTION

1.	Name:	Mass Store Continuator
2.	Size Estimates:	
	a. Code:	1100
	b. Local Data:	0
3.	Relative Statement Type Percent:	
	a. Computational:	10
	b. Logical:	90
4.	Execution Condition:	
	a. Frequency:	Mission phase loading, error recording
	b. Number of Operations:	Variable
5.	Complexity:	
	a. Number of Loops:	TBD
	b. Number of Paths:	TBD
	c. Number of Blocks:	TBD
	d. Block Exit Density:	TBD
6.	Type:	
	a. Simple:	No
	b. Function:	
	(1) Integer:	No
	(2) Real:	No
	(3) Boolean:	No
	(4) Complex:	No
	(5) Special:	Yes
	c. Reentrant:	No
	(1) Recursive:	
	(2) Non-Recursive:	
	d. Non-Reentrant:	Yes
7.	Priority:	3
8.	Development Time:	TBD
9.	Residence Requirements:	1100
10.	Events Causing Execution:	
	a. Periodic:	No
	b. Signal or Interrupt:	Yes
	c. Queue Processing:	No
	d. Direct Call:	No
11.	Output Data Description:	Device Requirements
12.	Input Requirements:	NA
13.	Remarks:	

Revision No. _____

Date _____

PROCESS ATTRIBUTE DESCRIPTION

1.	Name:	Timeline Interpreter/Controller
2.	Size Estimates:	
	a. Code:	800
	b. Local Data:	
3.	Relative Statement Type Percent:	
	a. Computational:	30
	b. Logical:	70
4.	Execution Condition:	
	a. Frequency:	Skeleton periodically
	b. Number of Operations:	Variable
5.	Complexity:	
	a. Number of Loops:	7
	b. Number of Paths:	24
	c. Number of Blocks:	19
	d. Block Exit Density:	1.3
6.	Type:	
	a. Simple:	No
	b. Function:	
	(1) Integer:	No
	(2) Real:	No
	(3) Boolean:	No
	(4) Complex:	No
	(5) Special:	Yes
	c. Reentrant:	No
	(1) Recursive:	
	(2) Non-Recursive:	
	d. Non-Reentrant:	Yes
7.	Priority:	
8.	Development Time:	
9.	Residence Requirements:	800
10.	Events Causing Execution:	
	a. Periodic:	Yes
	b. Signal or Interrupt:	Possibly
	c. Queue Processing:	No
	d. Direct Call:	No
11.	Output Data Description:	
12.	Input Requirements:	
13.	Remarks:	

PROCESS ATTRIBUTE DESCRIPTION

1.	Name:	Peripheral Reconfiguration
2.	Size Estimates:	
	a. Code:	700
	b. Local Data:	50
3.	Relative Statement Type Percent:	
	a. Computational:	20
	b. Logical:	80
4.	Execution Condition:	
	a. Frequency:	Response to solid peripheral failure
	b. Number of Operations:	
5.	Complexity:	
	a. Number of Loops:	11
	b. Number of Paths:	44
	c. Number of Blocks:	35
	d. Block Exit Density:	1.25
6.	Type:	
	a. Simple:	Yes
	b. Function:	
	(1) Integer:	No
	(2) Real:	No
	(3) Boolean:	No
	(4) Complex:	No
	(5) Special:	No
	c. Reentrant:	Yes
	(1) Recursive:	
	(2) Non-Recursive:	
	d. Non-Reentrant:	Yes
7.	Priority:	0
8.	Development Time:	TBD
9.	Residence Requirements:	750
10.	Events Causing Execution:	
	a. Periodic:	No
	b. Signal or Interrupt:	Yes, possibly
	c. Queue Processing:	No
	d. Direct Call:	Yes
11.	Output Data Description:	Peripheral Configuration Table Design Rqm
12.	Input Requirements:	Call Sequence
13.	Remarks:	

Revision No. _____
Date _____

PROCESS ATTRIBUTE DESCRIPTION

1.	Name:	CPU Reconfiguration
2.	Size Estimates:	
	a. Code:	4K
	b. Local Data:	4K
3.	Relative Statement Type Percent:	
	a. Computational:	20
	b. Logical:	80
4.	Execution Condition:	
	a. Frequency:	Response to CPU failure
	b. Number of Operations:	Variable
5.	Complexity:	
	a. Number of Loops:	30
	b. Number of Paths:	119
	c. Number of Blocks:	84
	d. Block Exit Density:	1.41
6.	Type:	
	a. Simple:	No
	b. Function:	
	(1) Integer:	Yes
	(2) Real:	No
	(3) Boolean:	Yes
	(4) Complex:	No
	(5) Special:	Yes
	c. Reentrant:	No
	(1) Recursive:	
	(2) Non-Recursive:	
	d. Non-Reentrant:	Yes
7.	Priority:	0
8.	Development Time:	TBD
9.	Residence Requirements:	SCU
10.	Events Causing Execution:	
	a. Periodic:	Yes
	b. Signal or Interrupt:	Yes
	c. Queue Processing:	No
	d. Direct Call:	Yes
11.	Output Data Description:	Reconfiguration Sequencer, Register Loading
12.	Input Requirements:	Discussed in body of report
13.	Remarks:	About 4500 words are estimated as required to implement reconfiguration sequences. Savings could be achieved by consolidating similar parts of sequences.

Revision No. _____
Date _____

PROCESS ATTRIBUTE DESCRIPTION

1.	Name:	ALARM
2.	Size Estimates:	
	a. Code:	500
	b. Local Data:	0
3.	Relative Statement Type Percent:	
	a. Computational:	20
	b. Logical:	80
4.	Execution Condition:	
	a. Frequency:	System Load
	b. Number of Operations:	
5.	Complexity:	
	a. Number of Loops:	6
	b. Number of Paths:	60
	c. Number of Blocks:	32
	d. Block Exit Density:	1.85
6.	Type:	
	a. Simple:	No
	b. Function:	
	(1) Integer:	No
	(2) Real:	No
	(3) Boolean:	No
	(4) Complex:	No
	(5) Special:	Yes
	c. Reentrant:	No
	(1) Recursive:	
	(2) Non-Recursive:	
	d. Non-Reentrant:	Yes
7.	Priority:	3
8.	Development Time:	TBD
9.	Residence Requirements:	500
10.	Events Causing Execution:	
	a. Periodic:	Possibly
	b. Signal or Interrupt:	No
	c. Queue Processing:	No
	d. Direct Call:	Yes
11.	Output Data Description:	Peripheral Device Requirements
12.	Input Requirements:	Call Sequence
13.	Remarks:	Alarm history tables required

Revision No. _____
Date _____

PROCESS ATTRIBUTE DESCRIPTION

1.	Name:	Display Initiator
2.	Size Estimates:	
	a. Code:	450
	b. Local Data:	20
3.	Relative Statement Type Percent:	
	a. Computational:	20
	b. Logical:	80
4.	Execution Condition:	
	a. Frequency:	System Load
	b. Number of Operations:	Variable
5.	Complexity:	
	a. Number of Loops:	5
	b. Number of Paths:	17
	c. Number of Blocks:	13
	d. Block Exit Density:	1.25
6.	Type:	
	a. Simple:	Yes
	b. Function:	
	(1) Integer:	No
	(2) Real:	No
	(3) Boolean:	No
	(4) Complex:	No
	(5) Special:	Yes
	c. Reentrant:	No
	(1) Recursive:	
	(2) Non-Recursive:	
	d. Non-Reentrant:	Yes
7.	Priority:	3
8.	Development Time:	TBD
9.	Residence Requirements:	470
10.	Events Causing Execution:	
	a. Periodic:	No
	b. Signal or Interrupt:	Possibly indirectly
	c. Queue Processing:	Possibly
	d. Direct Call:	Yes
11.	Output Data Description:	Display device requirements
12.	Input Requirements:	Call Sequence
13.	Remarks:	Initiation procedure is hardware dependent.

Revision No. _____
Date _____

PROCESS ATTRIBUTE DESCRIPTION

1.	Name:	Display Continuator
2.	Size Estimates:	
	a. Code:	450
	b. Local Data:	50 (process list)
3.	Relative Statement Type Percent:	
	a. Computational:	20
	b. Logical:	80
4.	Execution Condition:	
	a. Frequency:	Refresh Rate
	b. Number of Operations:	Variable
5.	Complexity:	
	a. Number of Loops:	5
	b. Number of Paths:	13
	c. Number of Blocks:	10
	d. Block Exit Density:	1.3
6.	Type:	
	a. Simple:	No
	b. Function:	
	(1) Integer:	No
	(2) Real:	No
	(3) Boolean:	No
	(4) Complex:	No
	(5) Special:	Yes
	c. Reentrant:	No
	(1) Recursive:	
	(2) Non-Recursive:	
	d. Non-Reentrant:	Yes
7.	Priority:	3
8.	Development Time:	TBD
9.	Residence Requirements:	
10.	Events Causing Execution:	
	a. Periodic:	Yes
	b. Signal or Interrupt:	Yes
	c. Queue Processing:	No
	d. Direct Call:	No
11.	Output Data Description:	Display device requirements
12.	Input Requirements:	NA
13.	Remarks:	Hardware dependent

Revision No. _____
Date _____

PROCESS ATTRIBUTE DESCRIPTION

1.	Name:	Timeline Display Initiate/Cancel
2.	Size Estimates:	
	a. Code:	500
	b. Local Data:	20
3.	Relative Statement Type Percent:	
	a. Computational:	20
	b. Logical:	80
4.	Execution Condition:	
	a. Frequency:	Response to Timeline Status Display request
	b. Number of Operations:	Variable
5.	Complexity:	
	a. Number of Loops:	10
	b. Number of Paths:	28
	c. Number of Blocks:	18
	d. Block Exit Density:	1.55
6.	Type:	
	a. Simple:	No
	b. Function:	
	(1) Integer:	No
	(2) Real:	No
	(3) Boolean:	No
	(4) Complex:	No
	(5) Special:	Yes
	c. Reentrant:	No
	(1) Recursive:	
	(2) Non-Recursive:	
	d. Non-Reentrant:	Yes
7.	Priority:	3
8.	Development Time:	TBD
9.	Residence Requirements:	520
10.	Events Causing Execution:	
	a. Periodic:	No
	b. Signal or Interrupt:	Possibly indirectly
	c. Queue Processing:	No
	d. Direct Call:	Yes
11.	Output Data Description:	Input required by DISPLAY INITIATOR
12.	Input Requirements:	Call Sequence
13.	Remarks:	Request is hardware oriented

Revision No. _____
Date _____

PROCESS ATTRIBUTE DESCRIPTION

1.	Name:	Panel Scan
2.	Size Estimates:	
	a. Code:	550
	b. Local Data:	50 (Process response list)
3.	Relative Statement Type Percent:	
	a. Computational:	20
	b. Logical:	80
4.	Execution Condition:	
	a. Frequency:	System Load
	b. Number of Operations:	Variable
5.	Complexity:	
	a. Number of Loops:	5
	b. Number of Paths:	11
	c. Number of Blocks:	9
	d. Block Exit Density:	1.22
6.	Type:	
	a. Simple:	No
	b. Function:	
	(1) Integer:	No
	(2) Real:	No
	(3) Boolean:	No
	(4) Complex:	No
	(5) Special:	Yes
	c. Reentrant:	No
	(1) Recursive:	
	(2) Non-Recursive:	
	d. Non-Reentrant:	Yes
7.	Priority:	3
8.	Development Time:	TBD
9.	Residence Requirements:	600
10.	Events Causing Execution:	
	a. Periodic:	Possibly
	b. Signal or Interrupt:	Possibly
	c. Queue Processing:	No
	d. Direct Call:	No
11.	Output Data Description:	Process requirements associated with switches, Peripheral Device Requirements
12.	Input Requirements:	Peripheral Device
13.	Remarks:	Possibly periodic operation or based on change of state signal

Revision No. _____
Date _____

PROCESS ATTRIBUTE DESCRIPTION

1.	Name:	Digital Input
2.	Size Estimates:	
	a. Code:	200
	b. Local Data:	0
3.	Relative Statement Type Percent:	
	a. Computational:	10
	b. Logical:	90
4.	Execution Condition:	
	a. Frequency:	Scan Frequency TBD
	b. Number of Operations:	
5.	Complexity:	
	a. Number of Loops:	5
	b. Number of Paths:	13
	c. Number of Blocks:	8
	d. Block Exit Density:	1.62
6.	Type:	
	a. Simple:	No
	b. Function:	
	(1) Integer:	No
	(2) Real:	No
	(3) Boolean:	No
	(4) Complex:	No
	(5) Special:	Yes
	c. Reentrant:	No
	(1) Recursive:	
	(2) Non-Recursive:	
	d. Non-Reentrant:	Yes
7.	Priority:	3
8.	Development Time:	TBD
9.	Residence Requirements:	200
10.	Events Causing Execution:	
	a. Periodic:	Yes
	b. Signal or Interrupt:	No
	c. Queue Processing:	No
	d. Direct Call:	Possibly
11.	Output Data Description:	Digital Status Tables
12.	Input Requirements:	Peripheral Device Requirements
13.	Remarks:	Possible hardware implementation

APPENDIX C. FUNCTIONAL FLOWCHARTS

High-level functional logic diagrams are rendered for critical processes in the Flight Executive. Typical examples of highly hardware-dependent processes are presented.

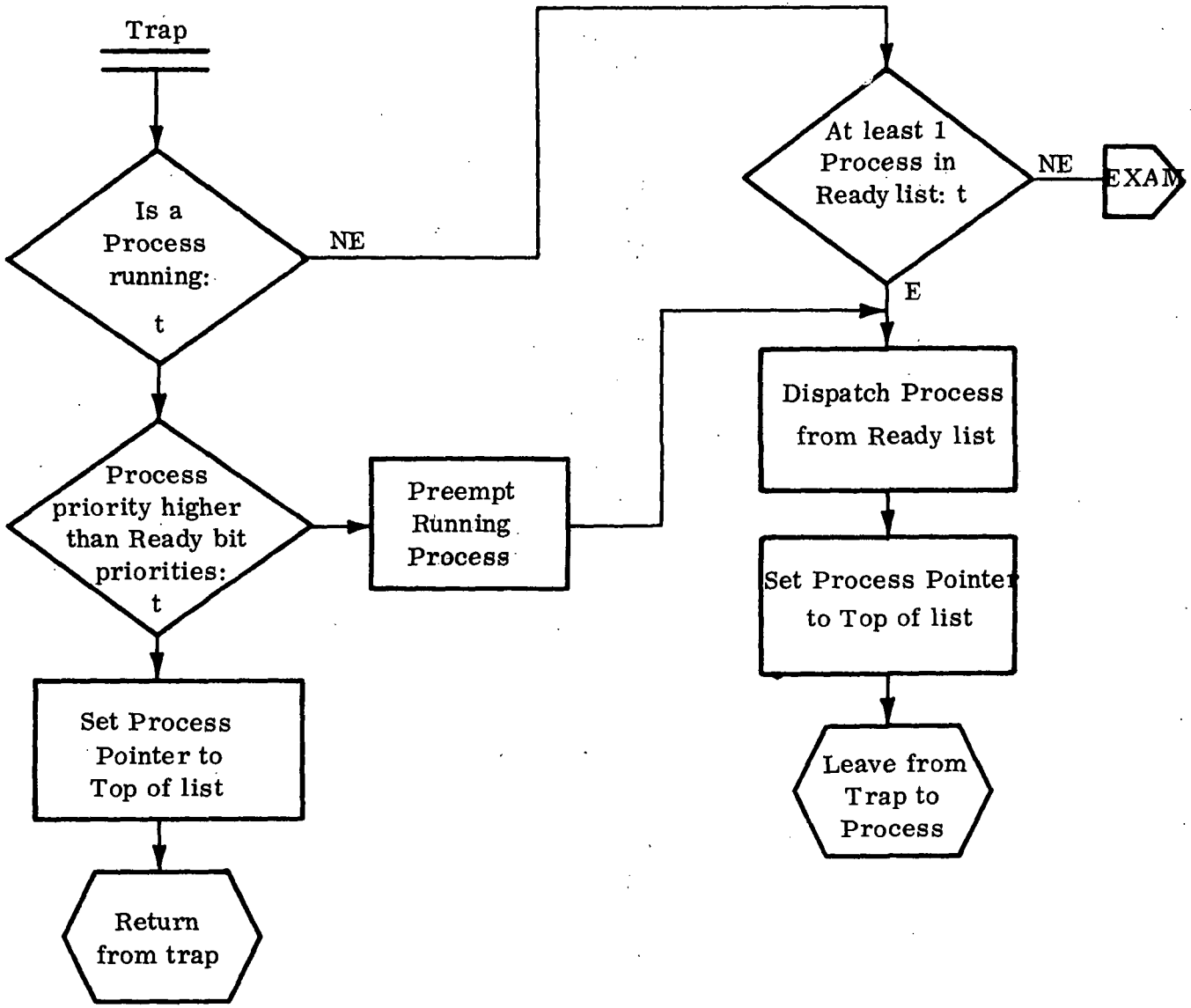


FIGURE C1. PROCESS DISPATCHER (SHEET 1 OF 2)

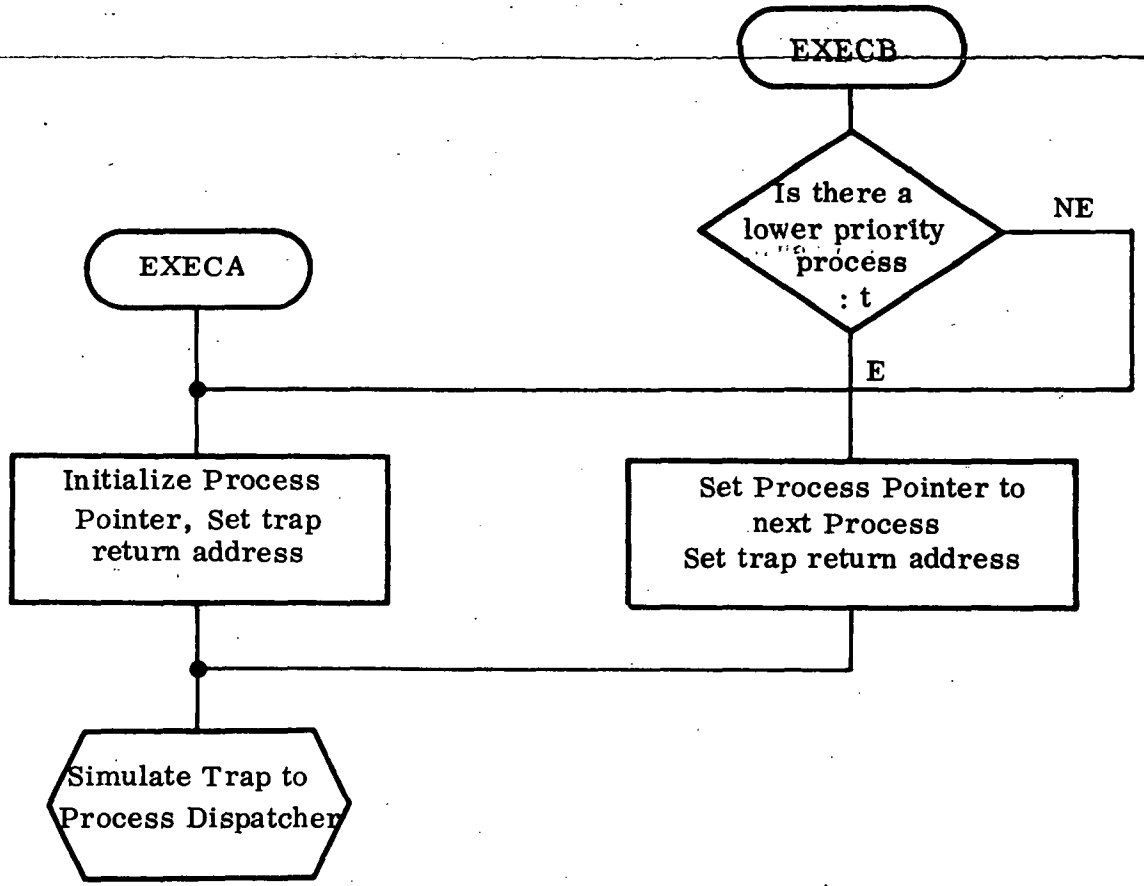


FIGURE C1. PROCESS DISPATCHER SOFTWARE ENTRIES (SHEET 2 OF 2)

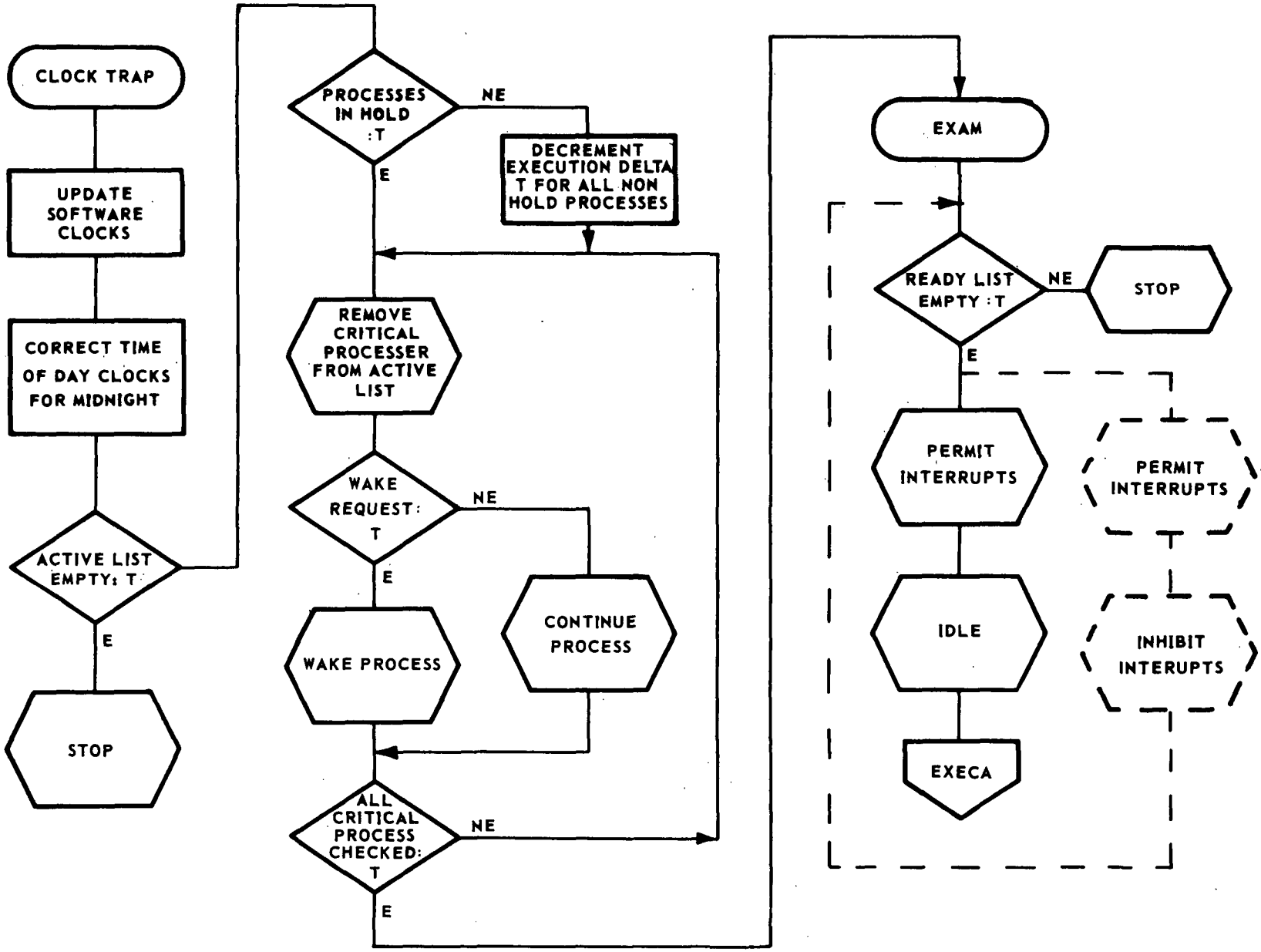


FIGURE C2. SOFTWARE CLOCK UPDATE AND EXAMINE

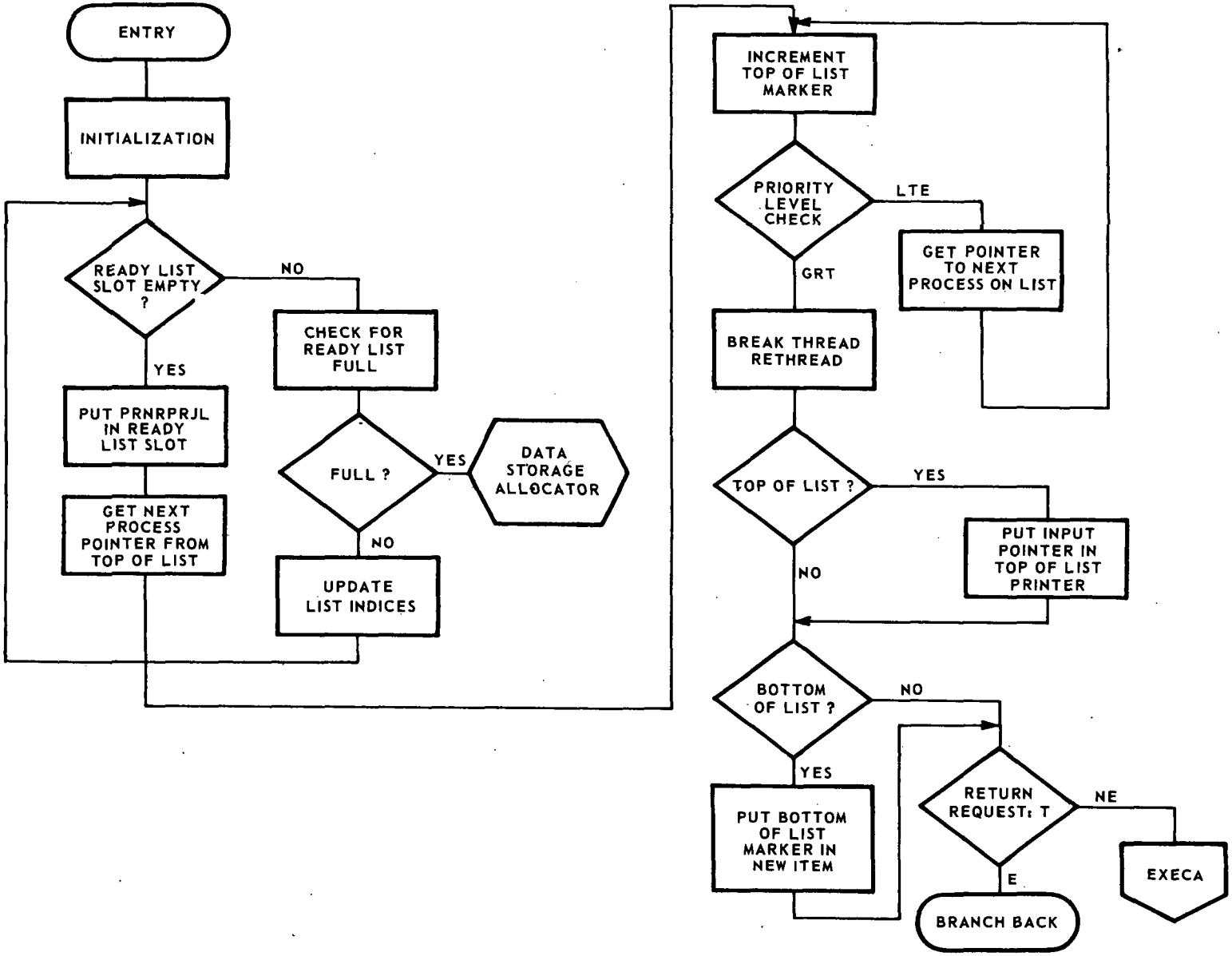


FIGURE C3. INSERT IN READY LIST

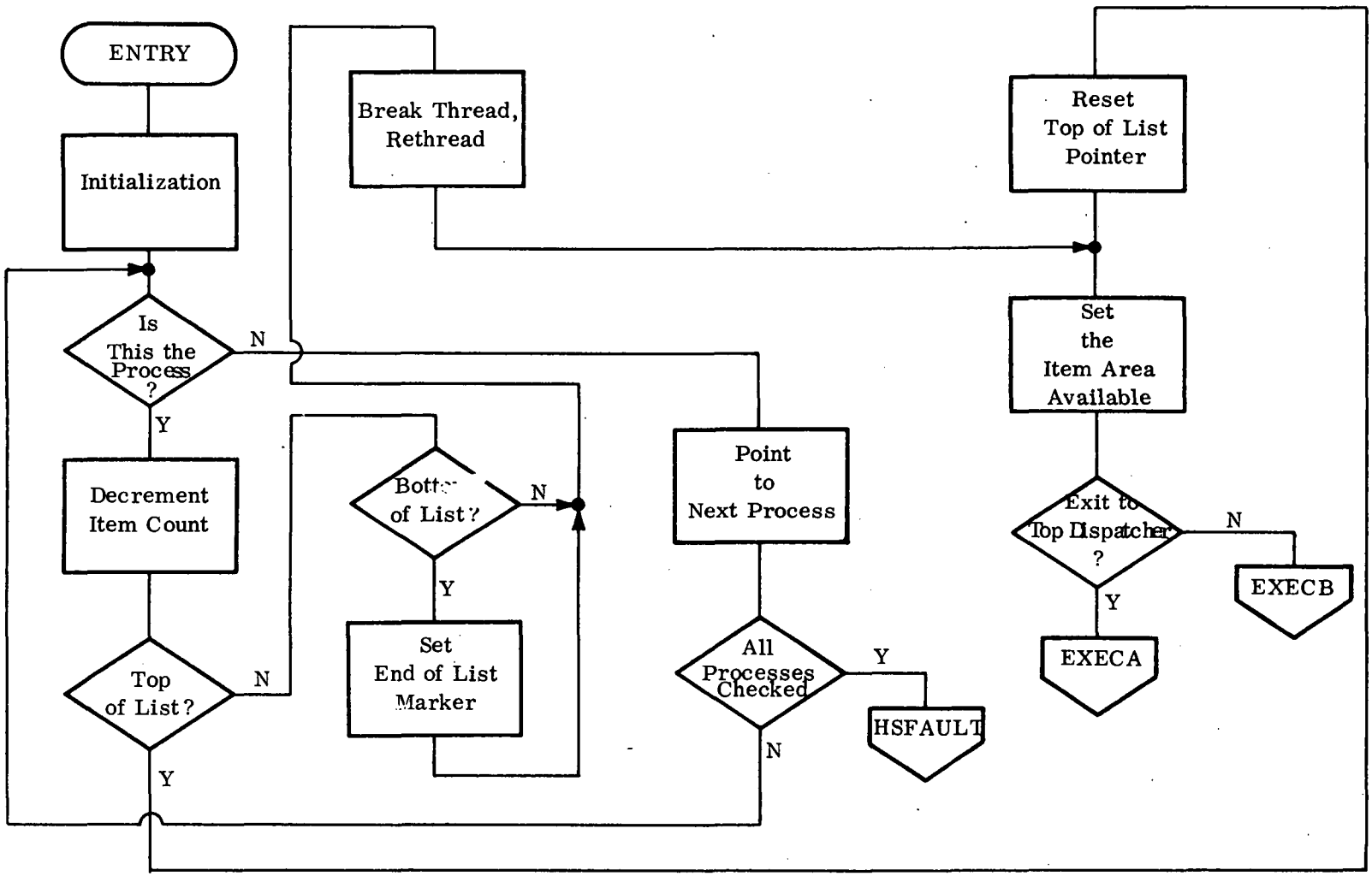


FIGURE C4. REMOVE – PRELIMINARY

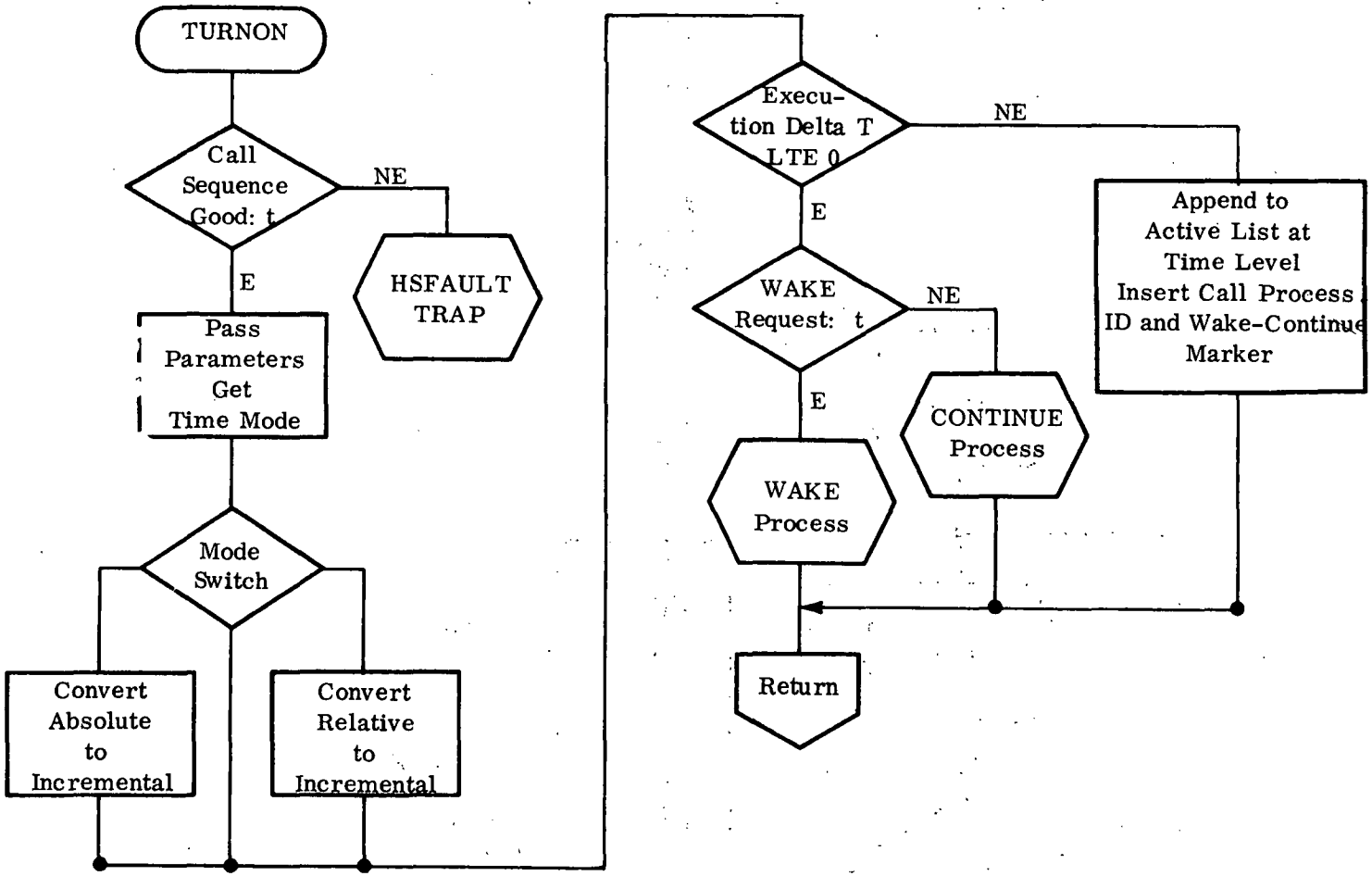


FIGURE C5. TURNON – ACTIVATE A PROCESS

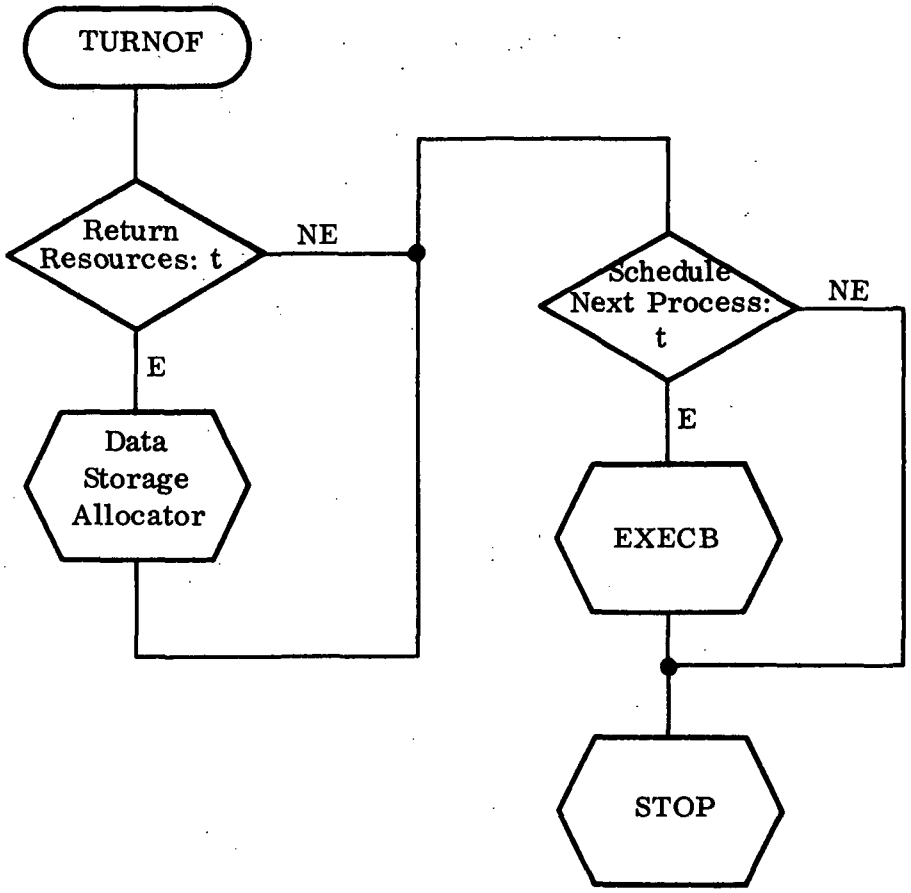


FIGURE C6. TURNOF – TERMINATE A PROCESS

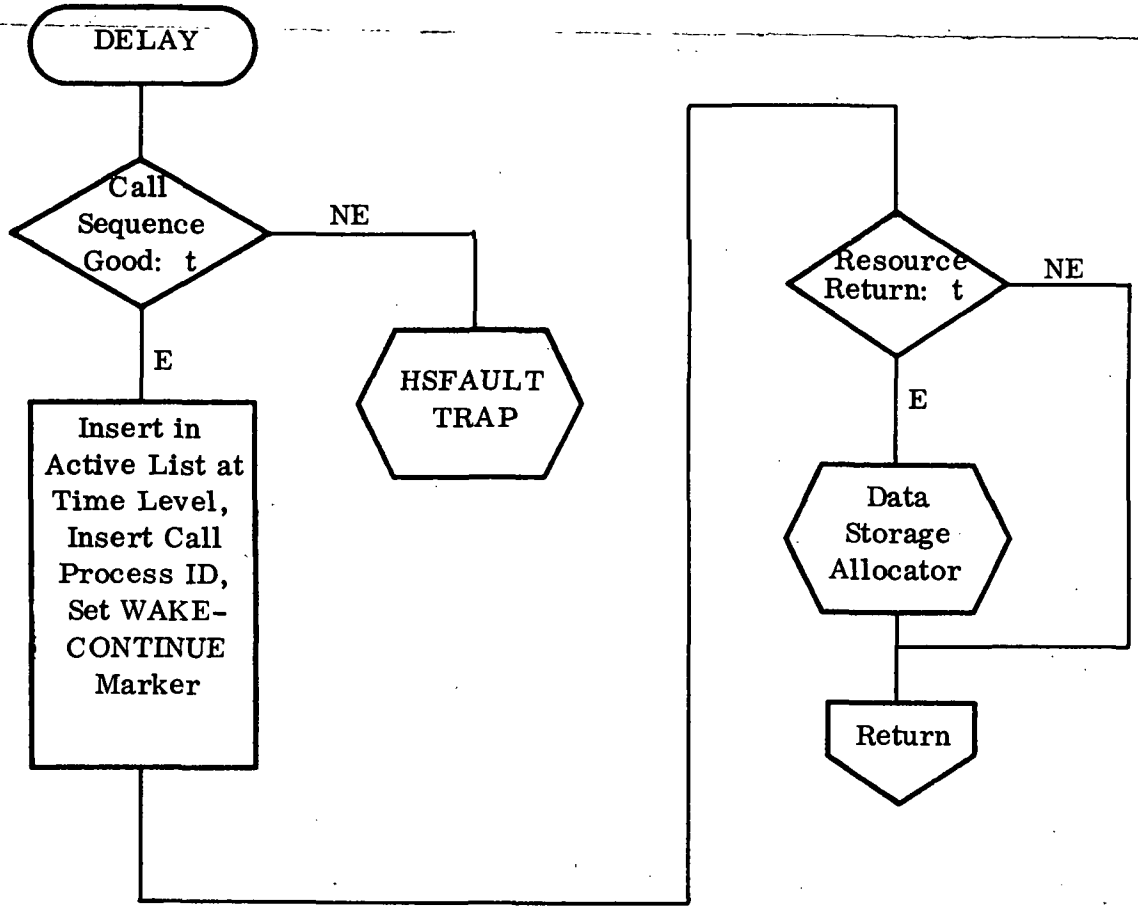
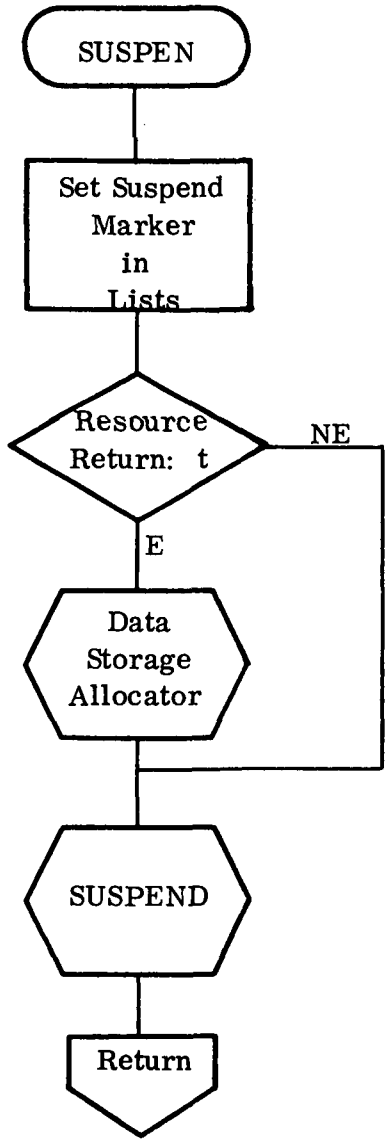


FIGURE C7. DELAY



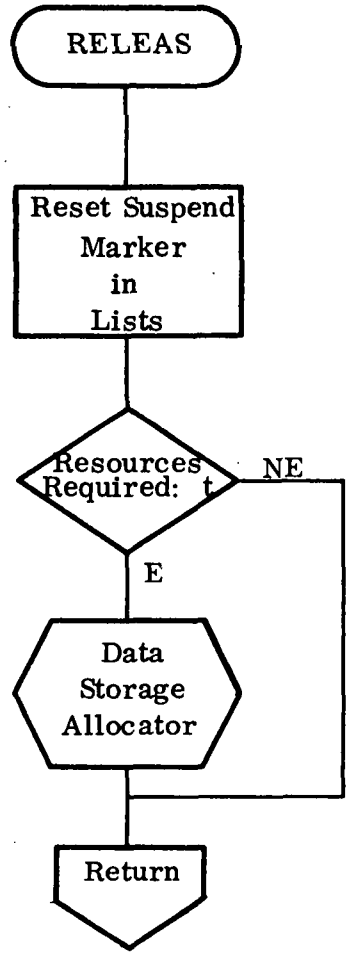


FIGURE C8. SUSPEN – RELEAS

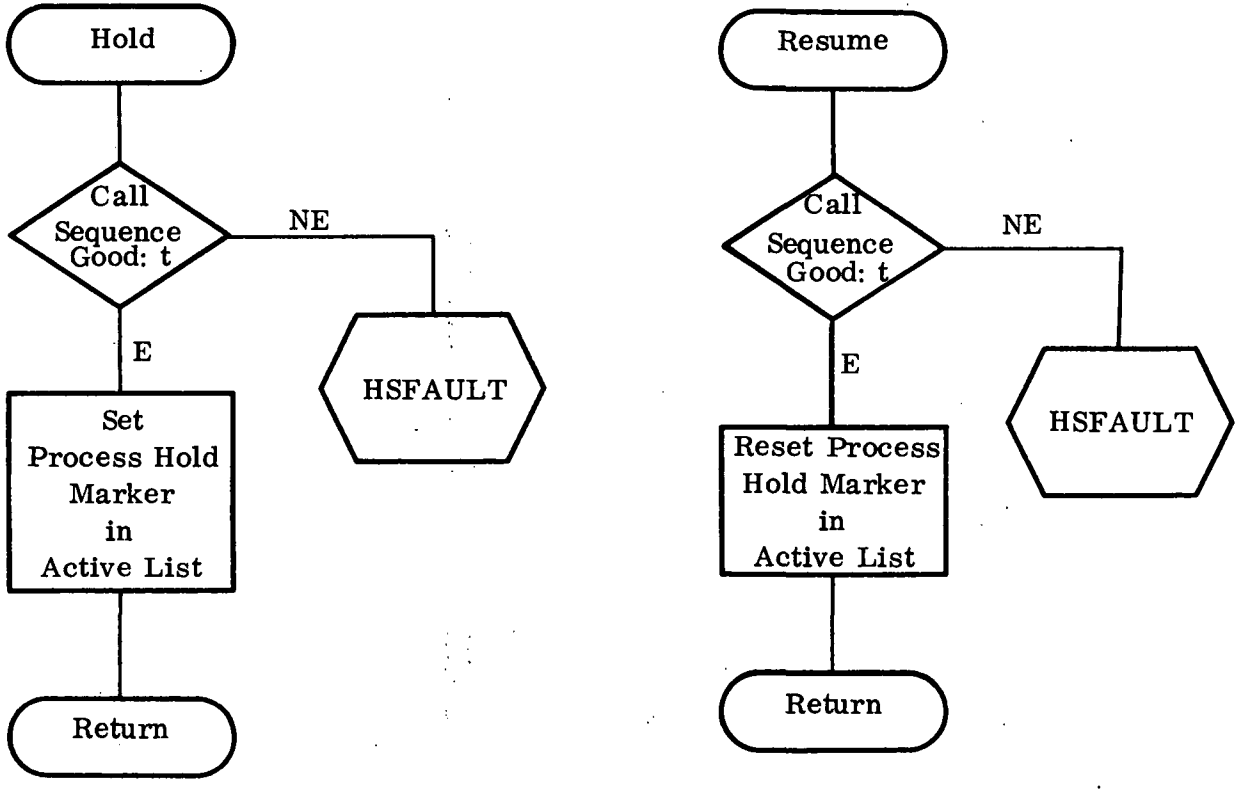


FIGURE C9. HOLD/RESUME

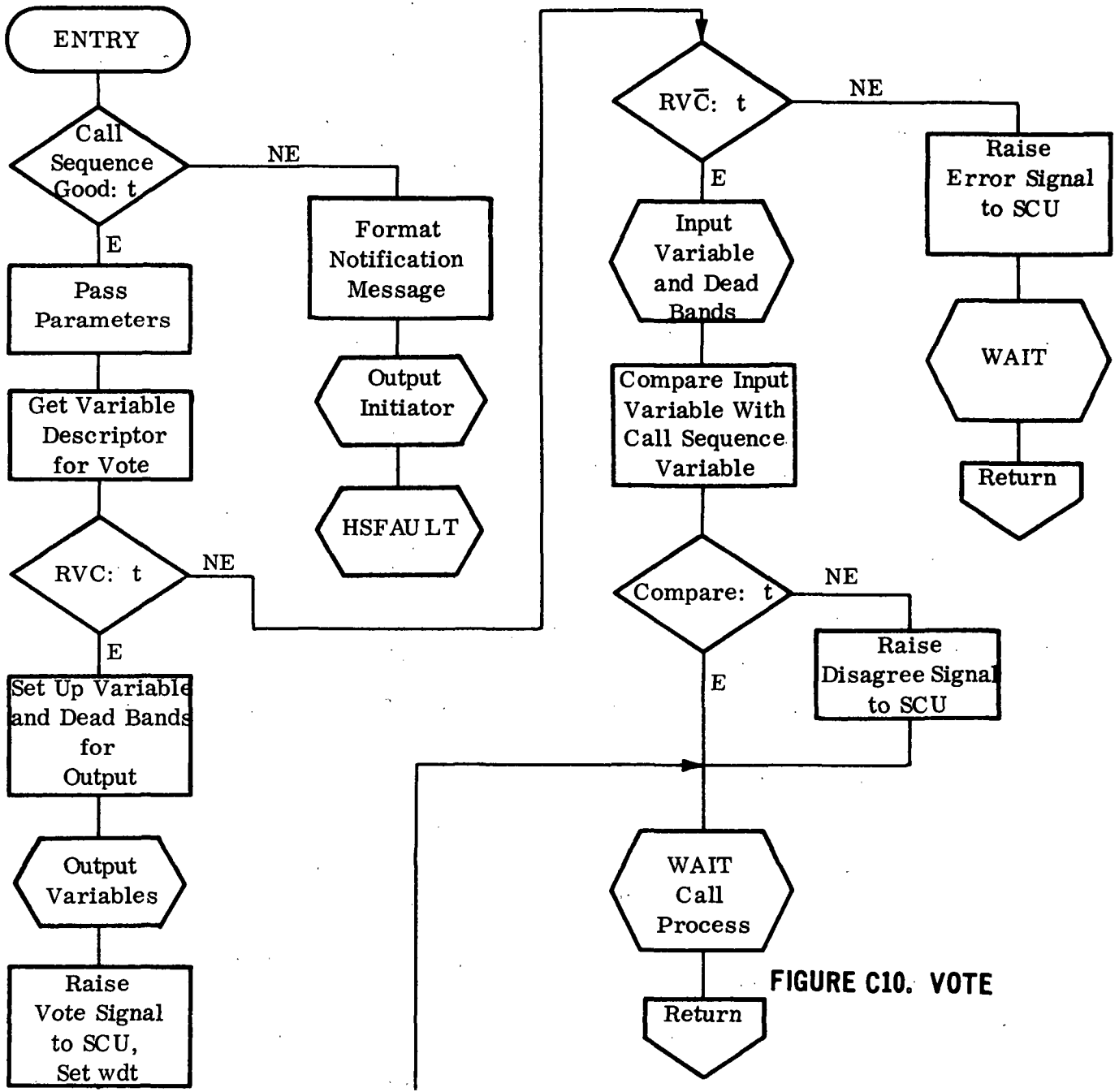


FIGURE C10. VOTE

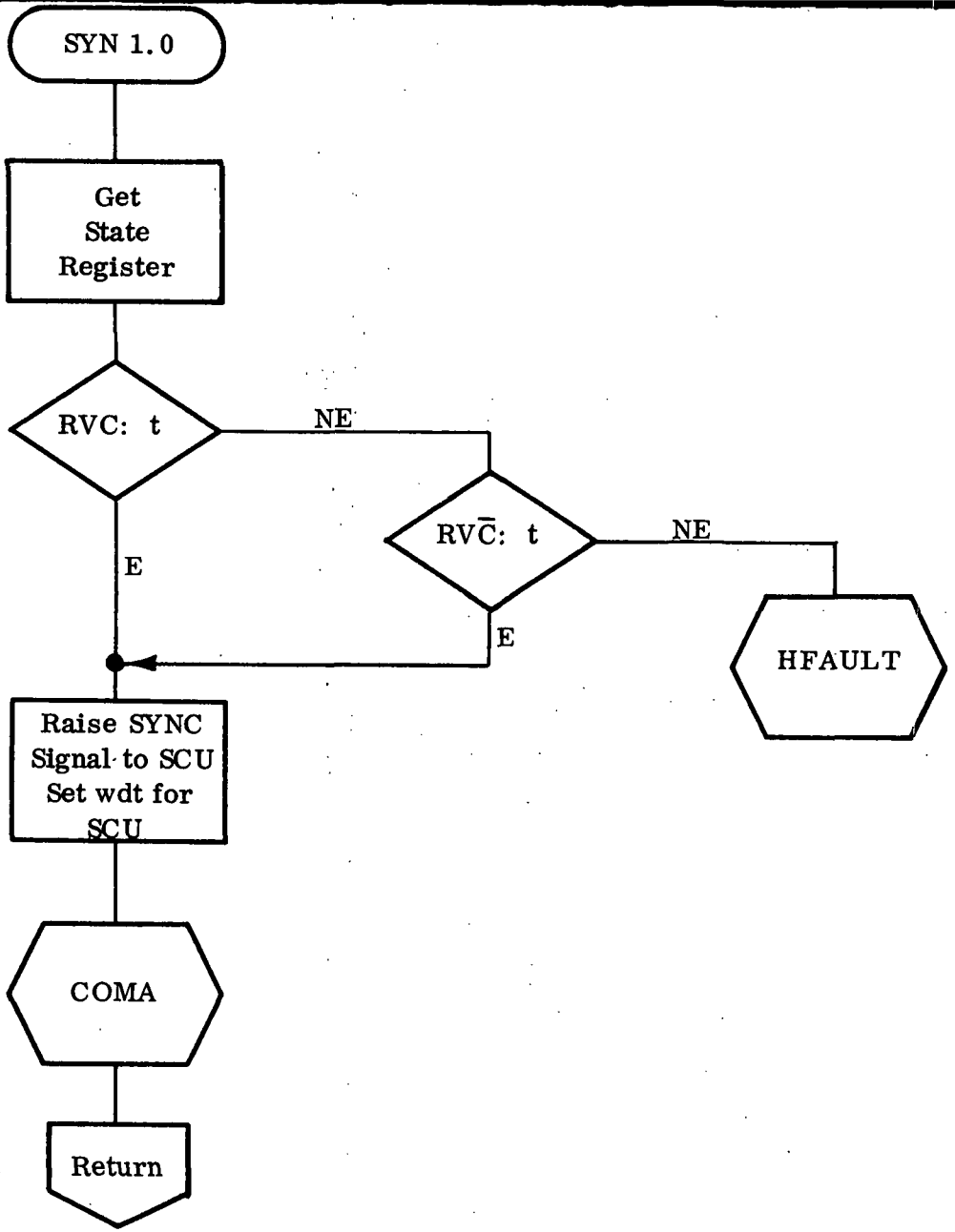


FIGURE C11. SYNC

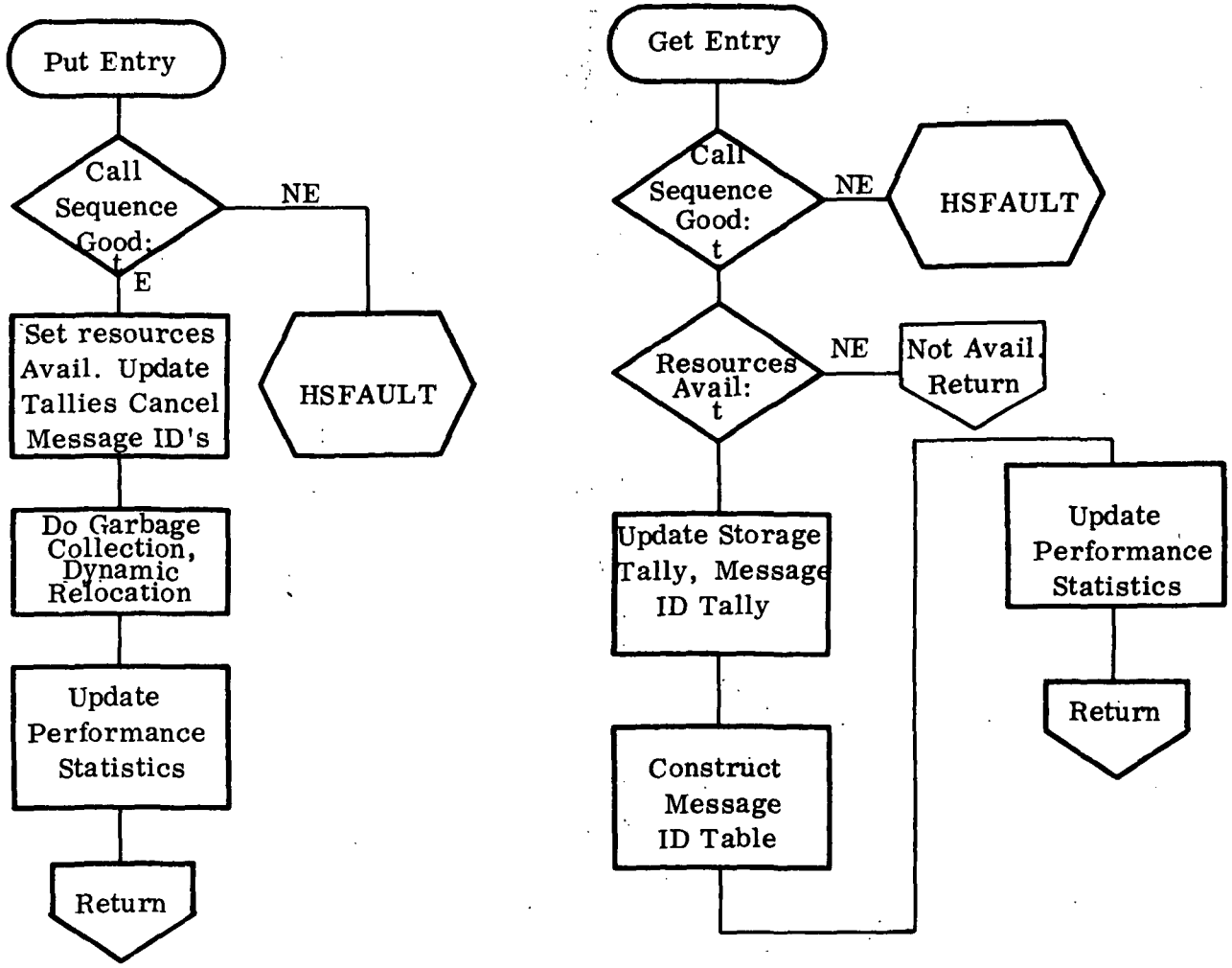


FIGURE C12. DATA STORAGE ALLOCATOR

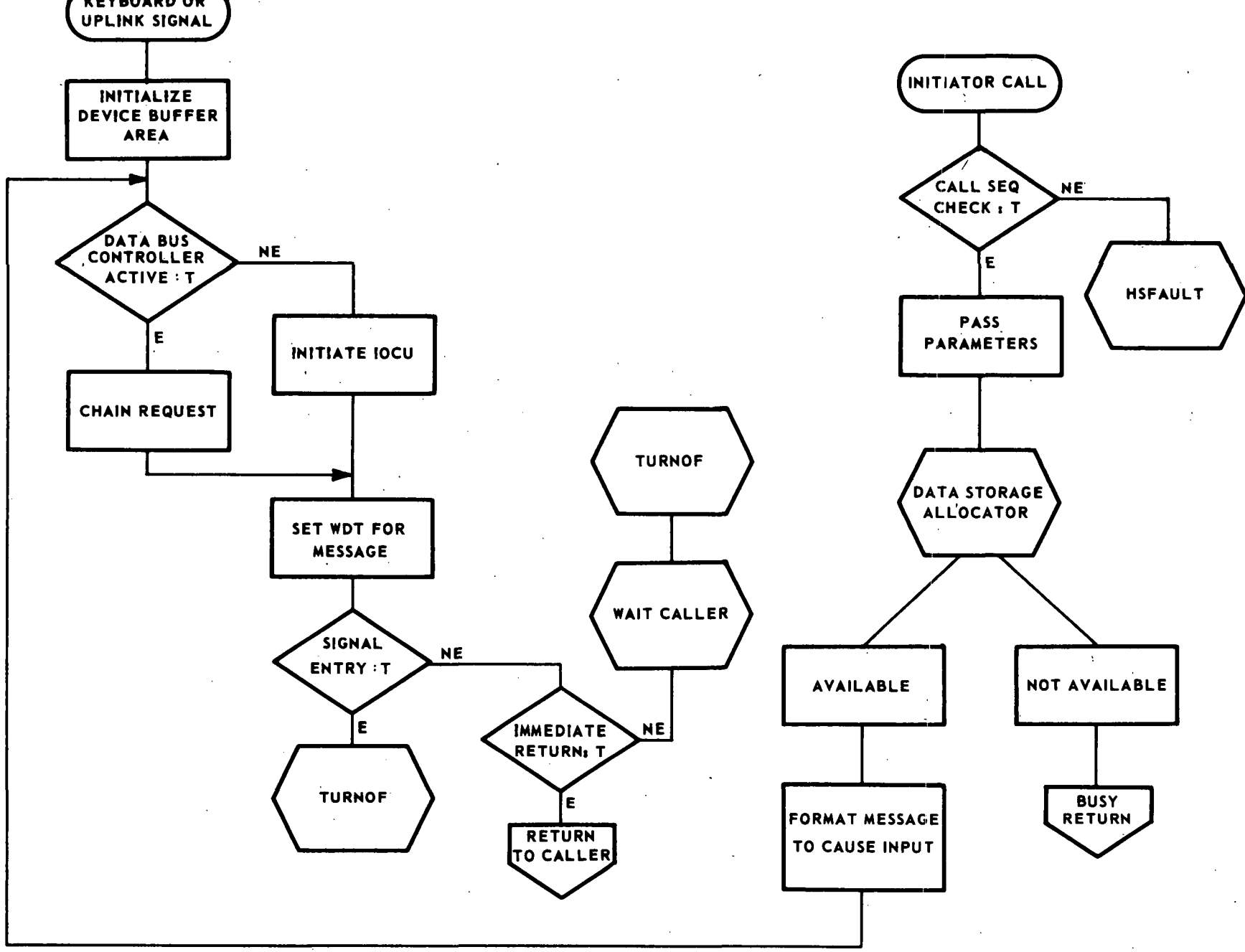


FIGURE C13. INPUT INITIATOR

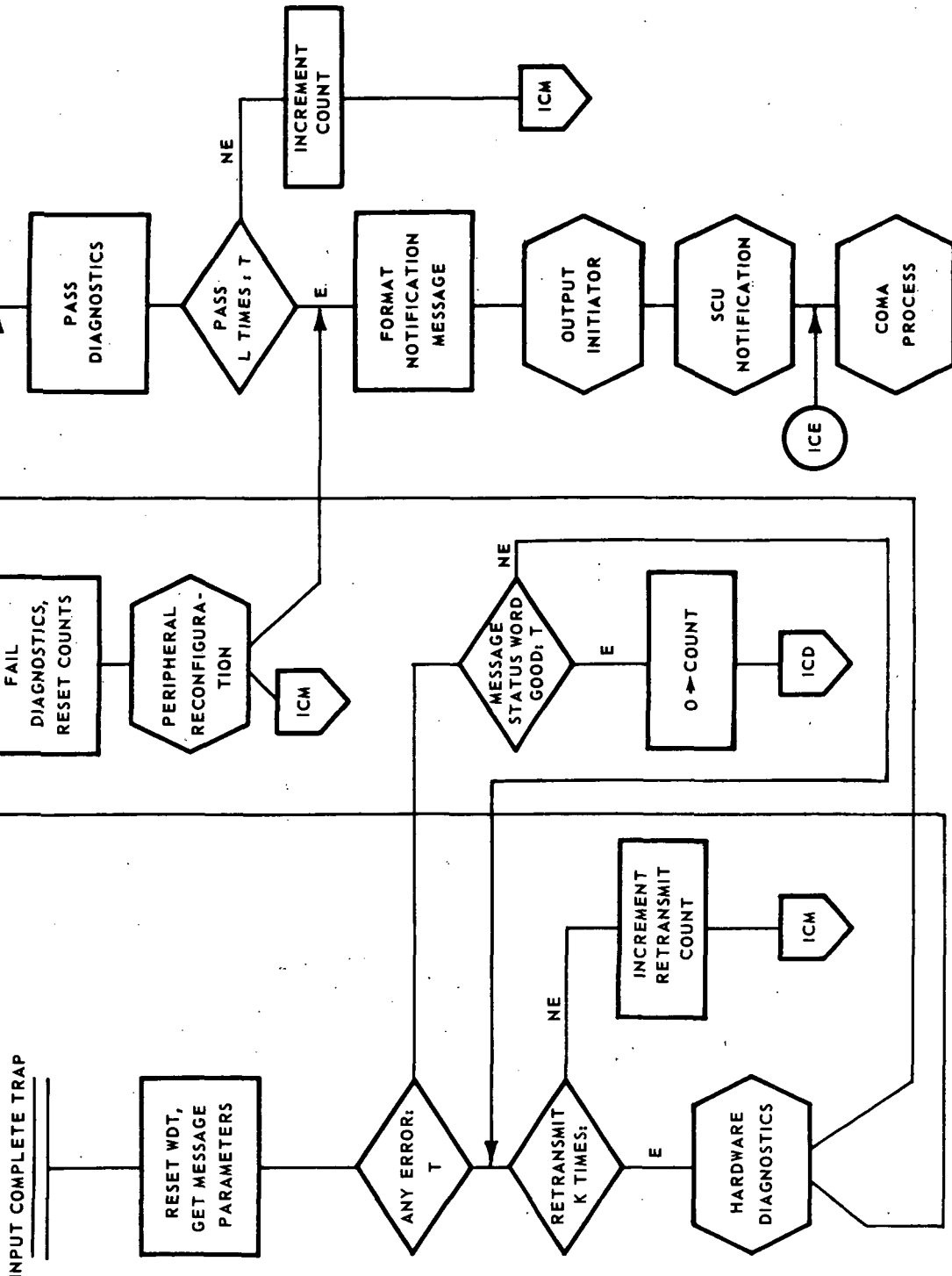


FIGURE C14. INPUT TERMINATOR (SHEET 1 OF 6)

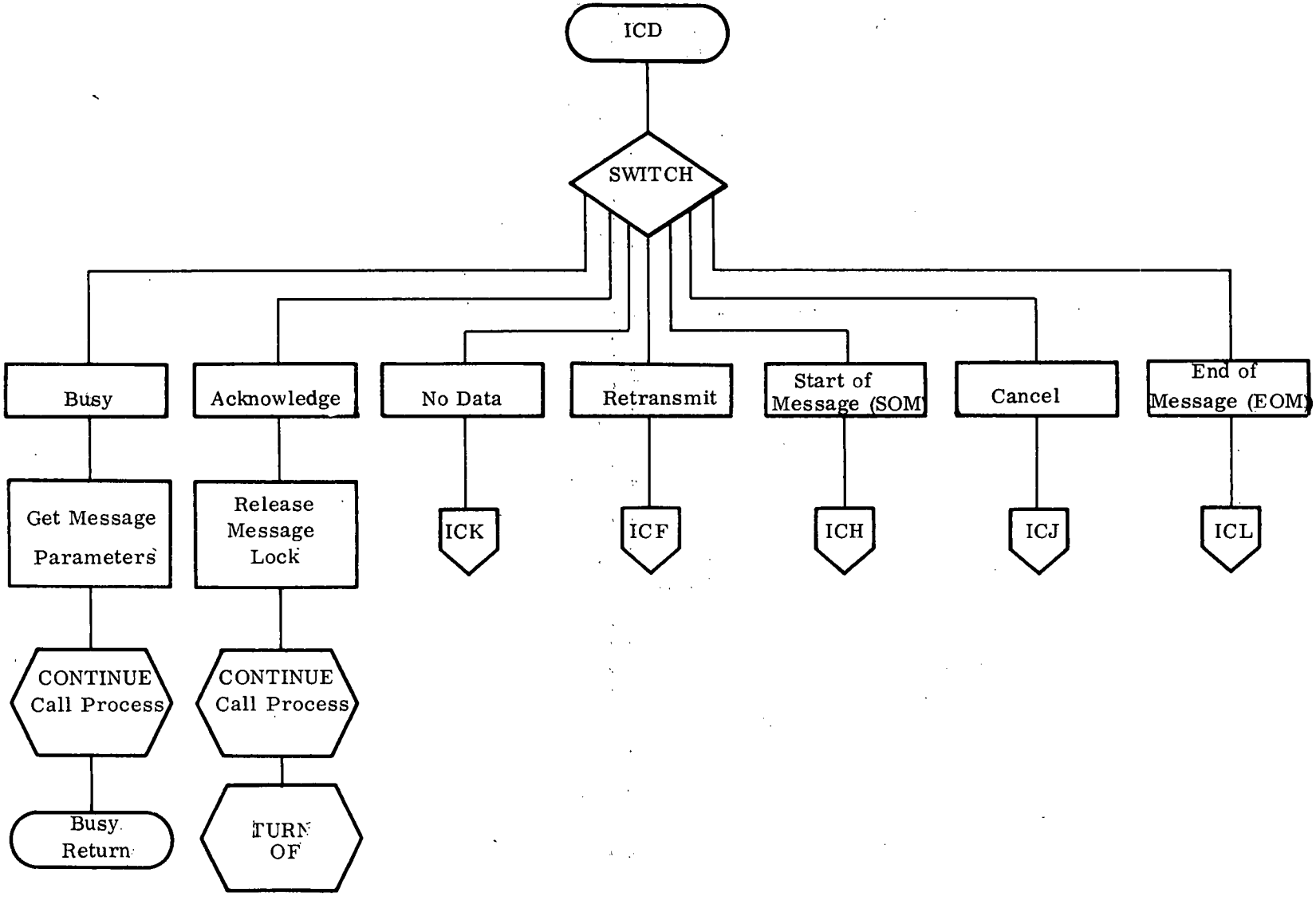


FIGURE C14. INPUT TERMINATOR (SHEET 2 OF 6)

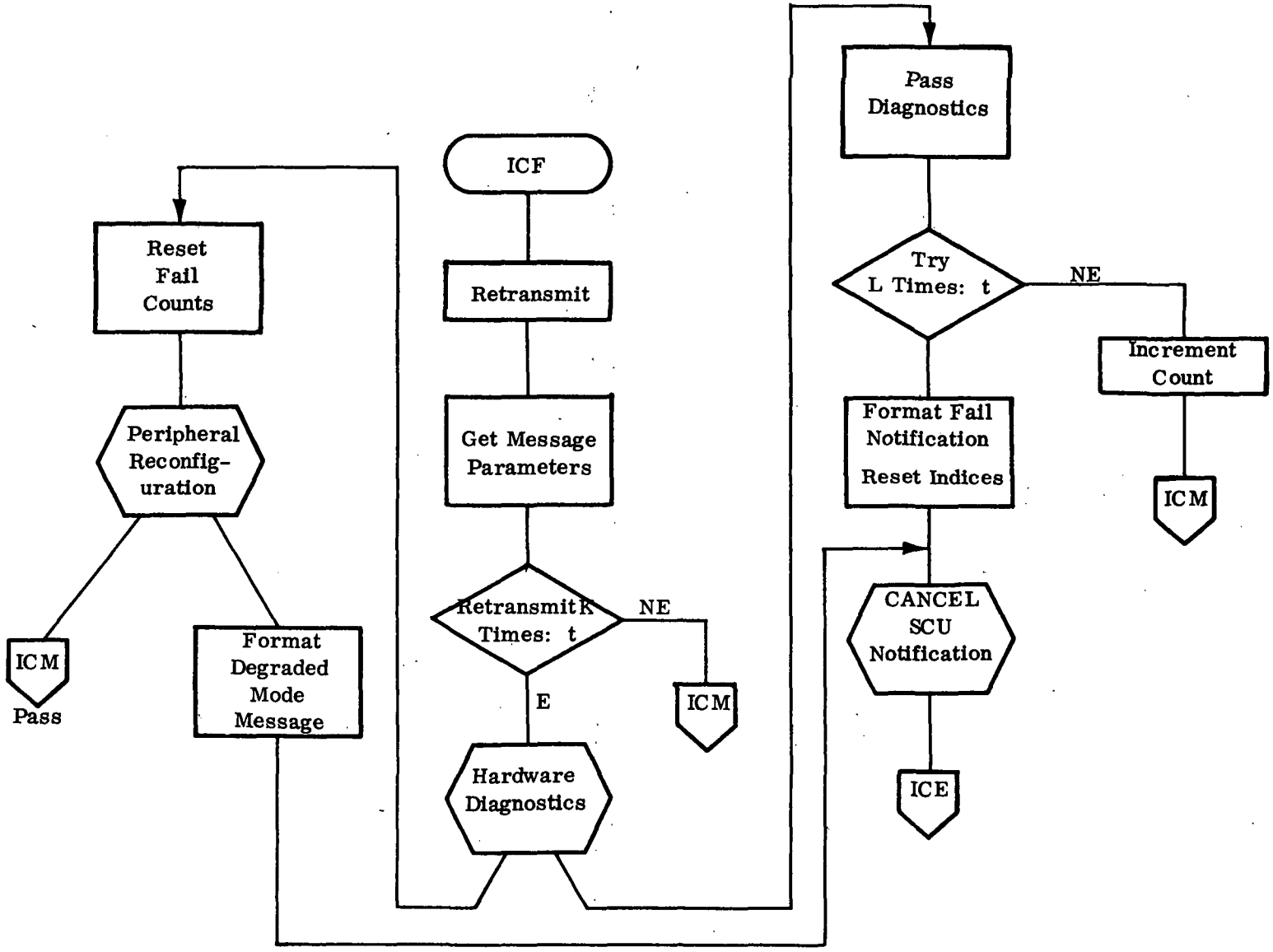


FIGURE C14. INPUT TERMINATOR (SHEET 3 OF 6)

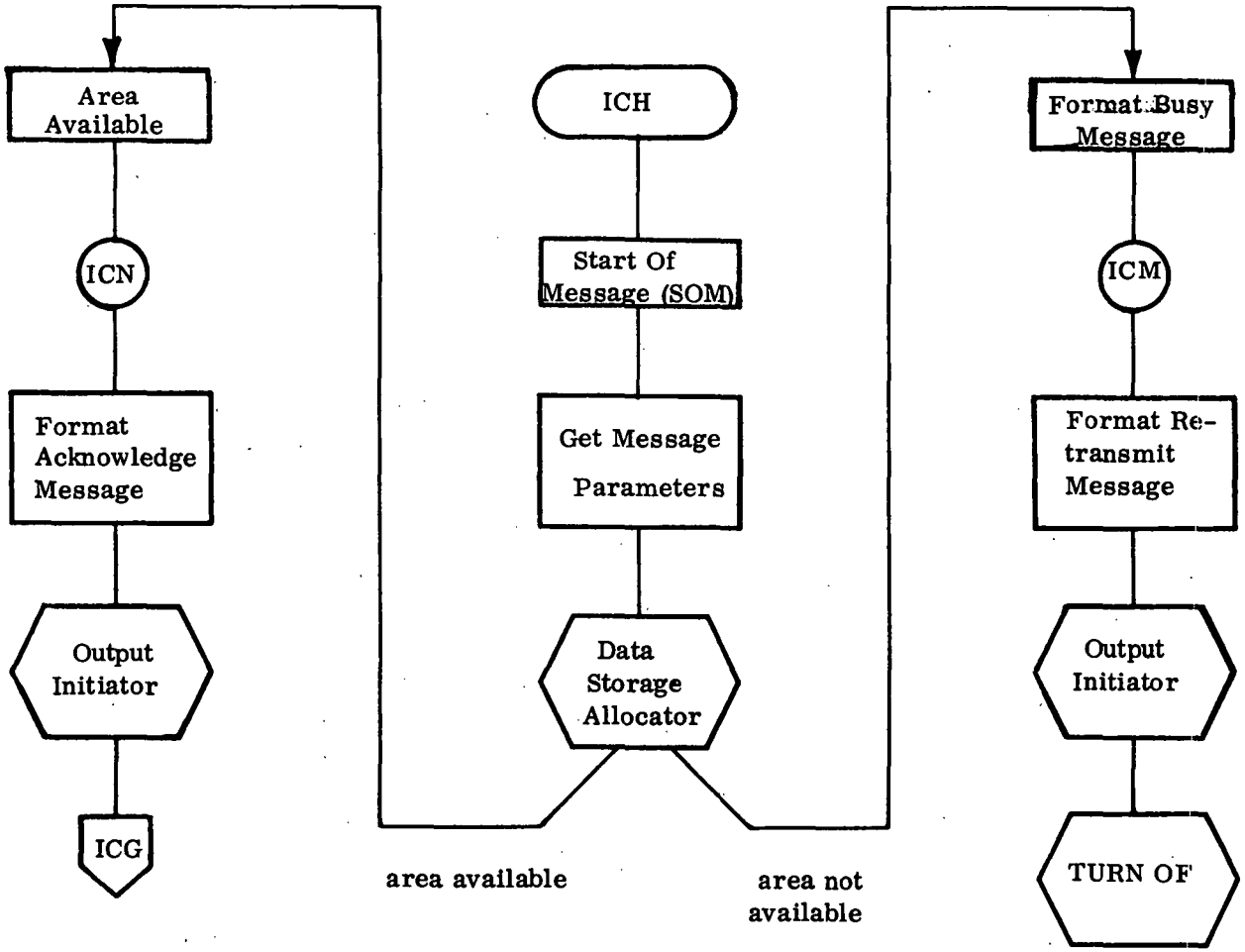
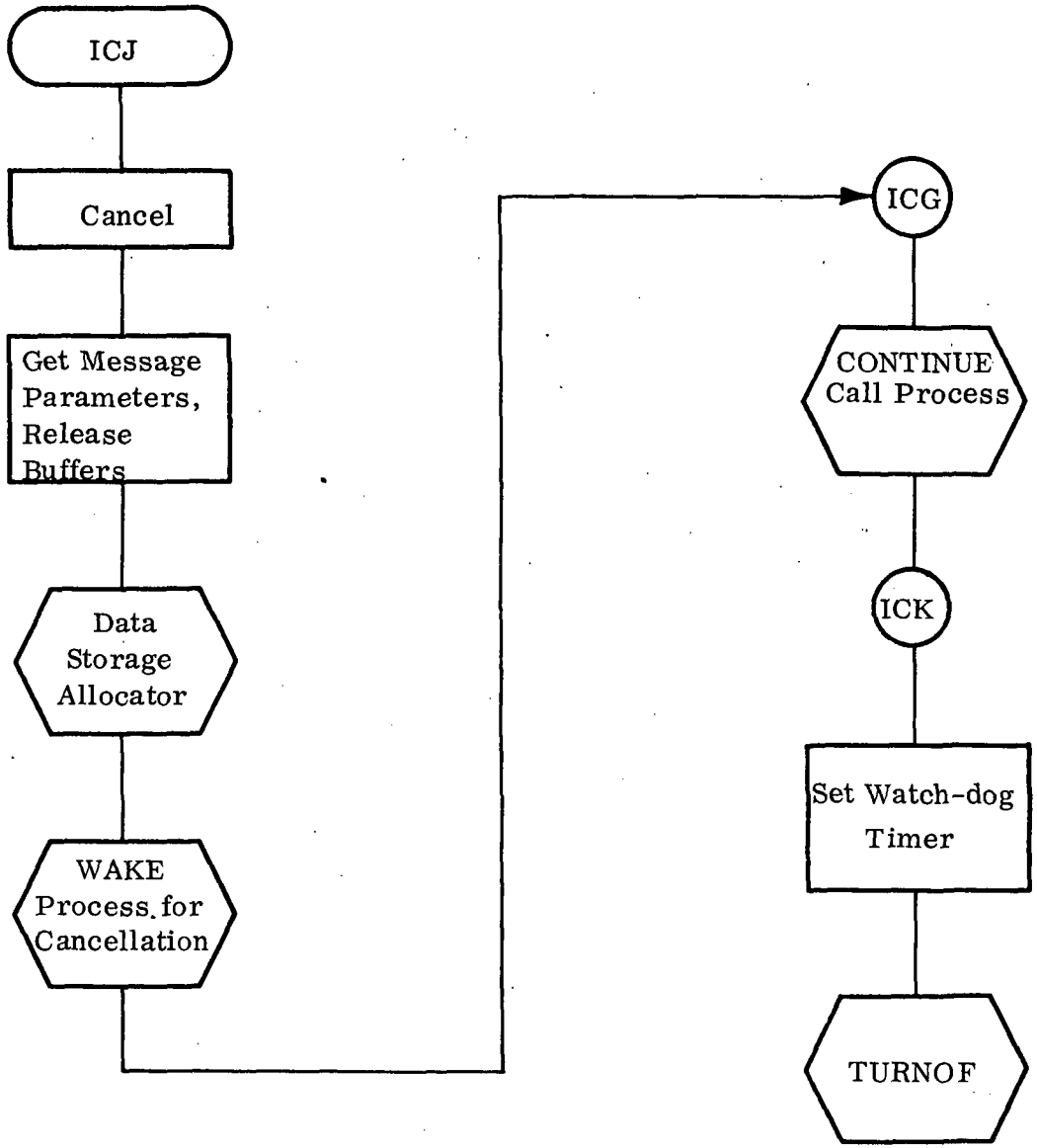


FIGURE C14. INPUT TERMINATOR (SHEET 4 OF 6)


ICJ
Cancel
Get Message Parameters, Release Buffers
Data Storage Allocator
WAKE Process for Cancellation
ICG
CONTINUE Call Process
ICK
Set Watch-dog Timer
TURNOF

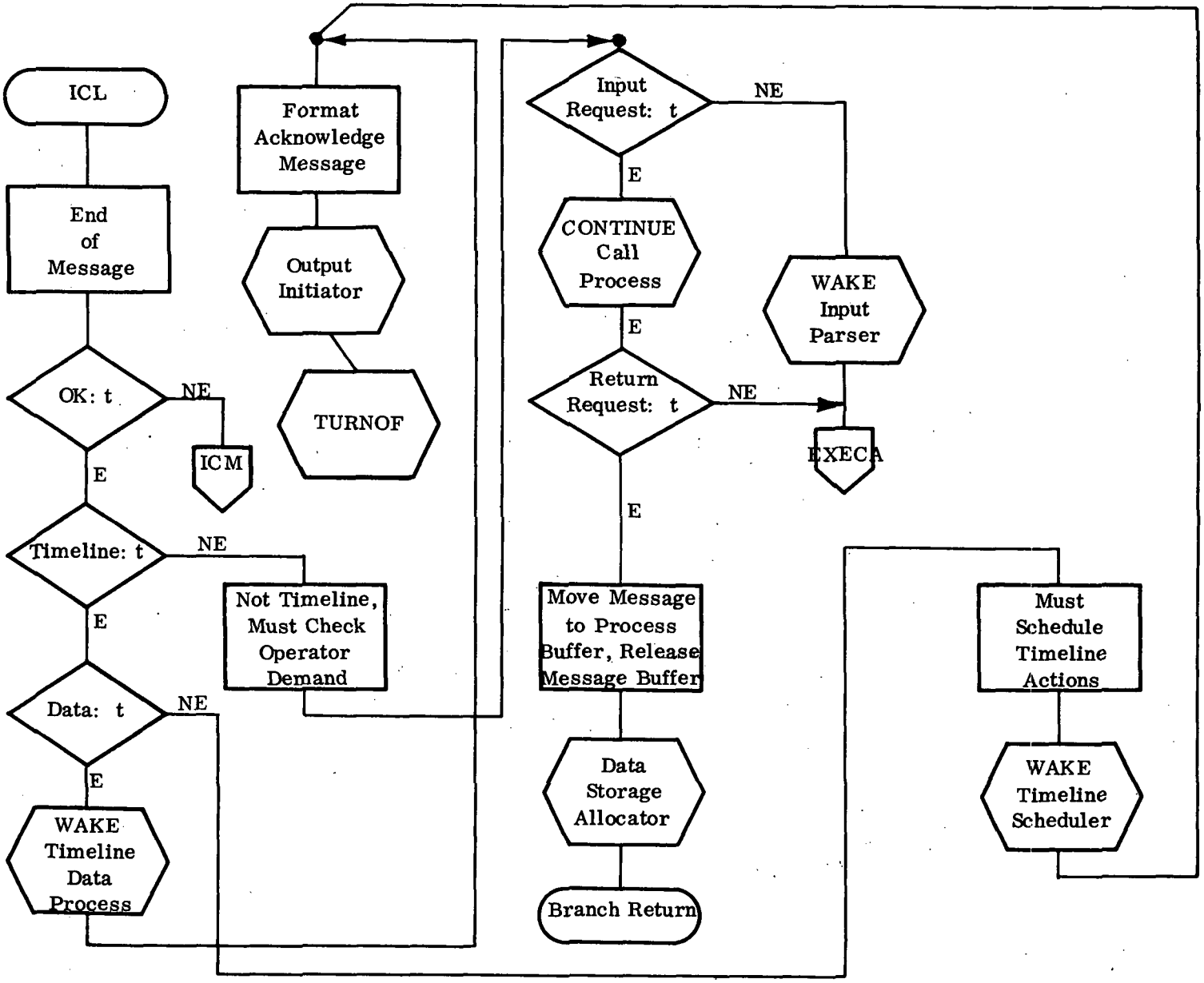


FIGURE C14. INPUT TERMINATOR (SHEET 6 OF 6)

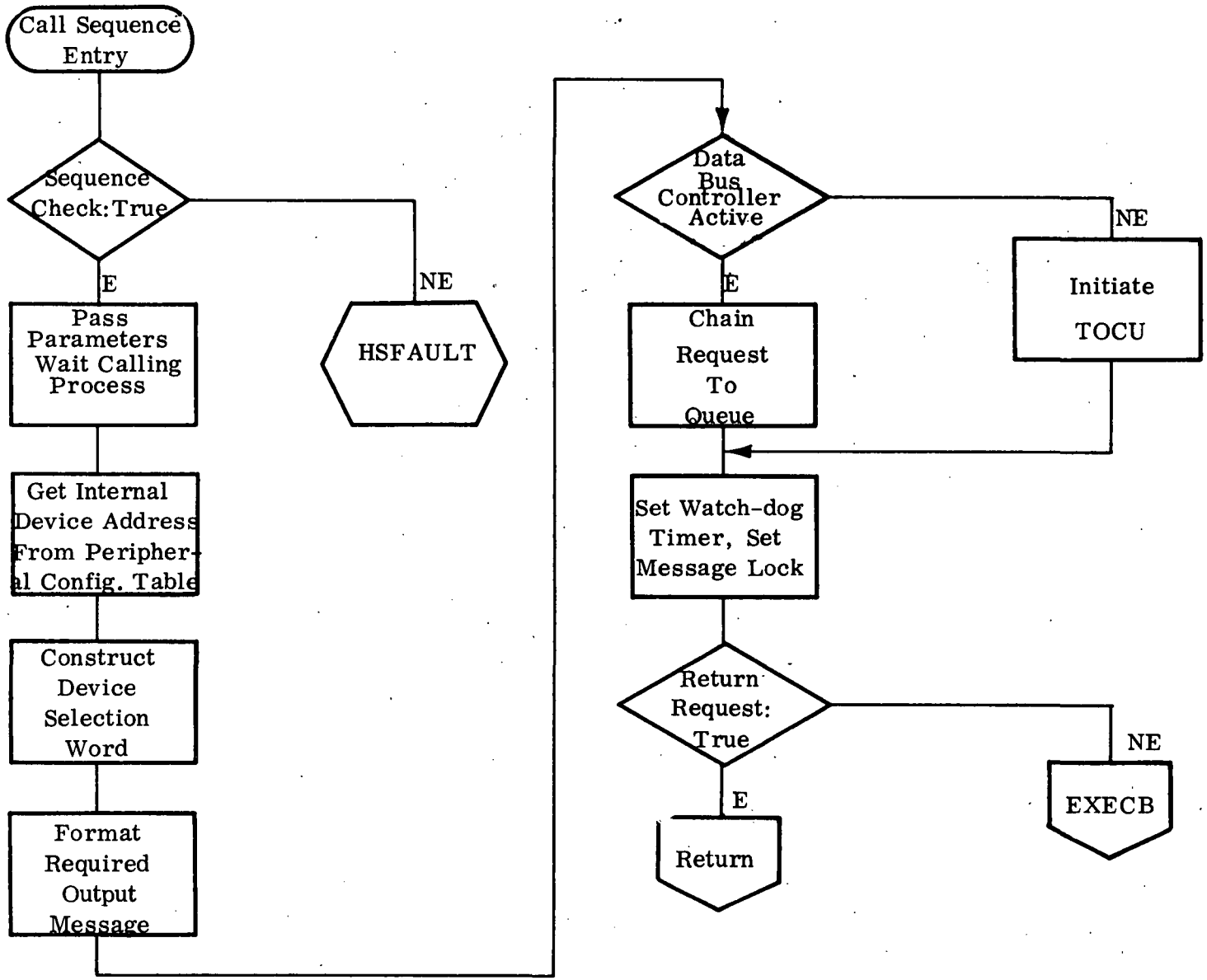


FIGURE C15. OUTPUT INITIATOR

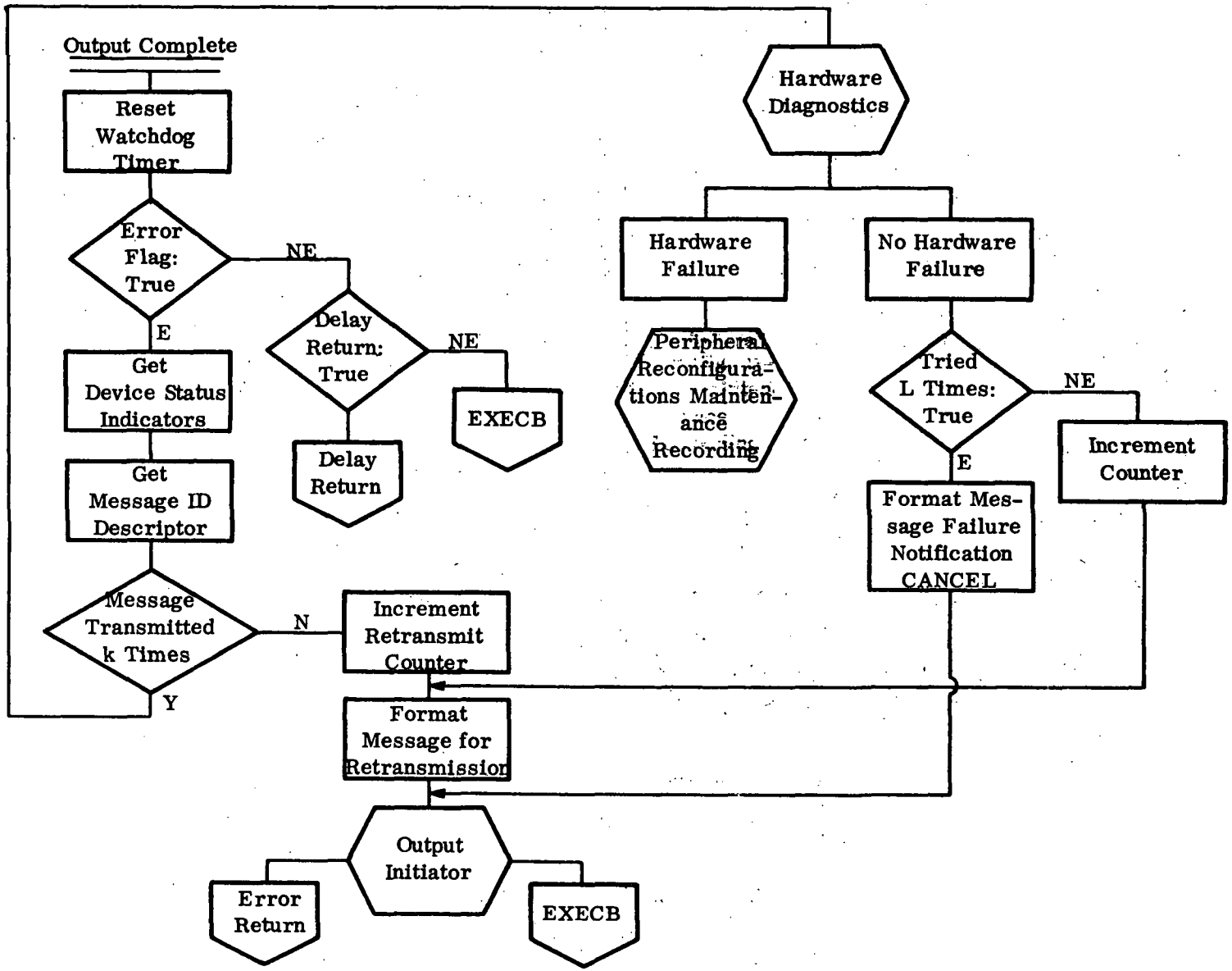


FIGURE C16. OUTPUT TERMINATOR

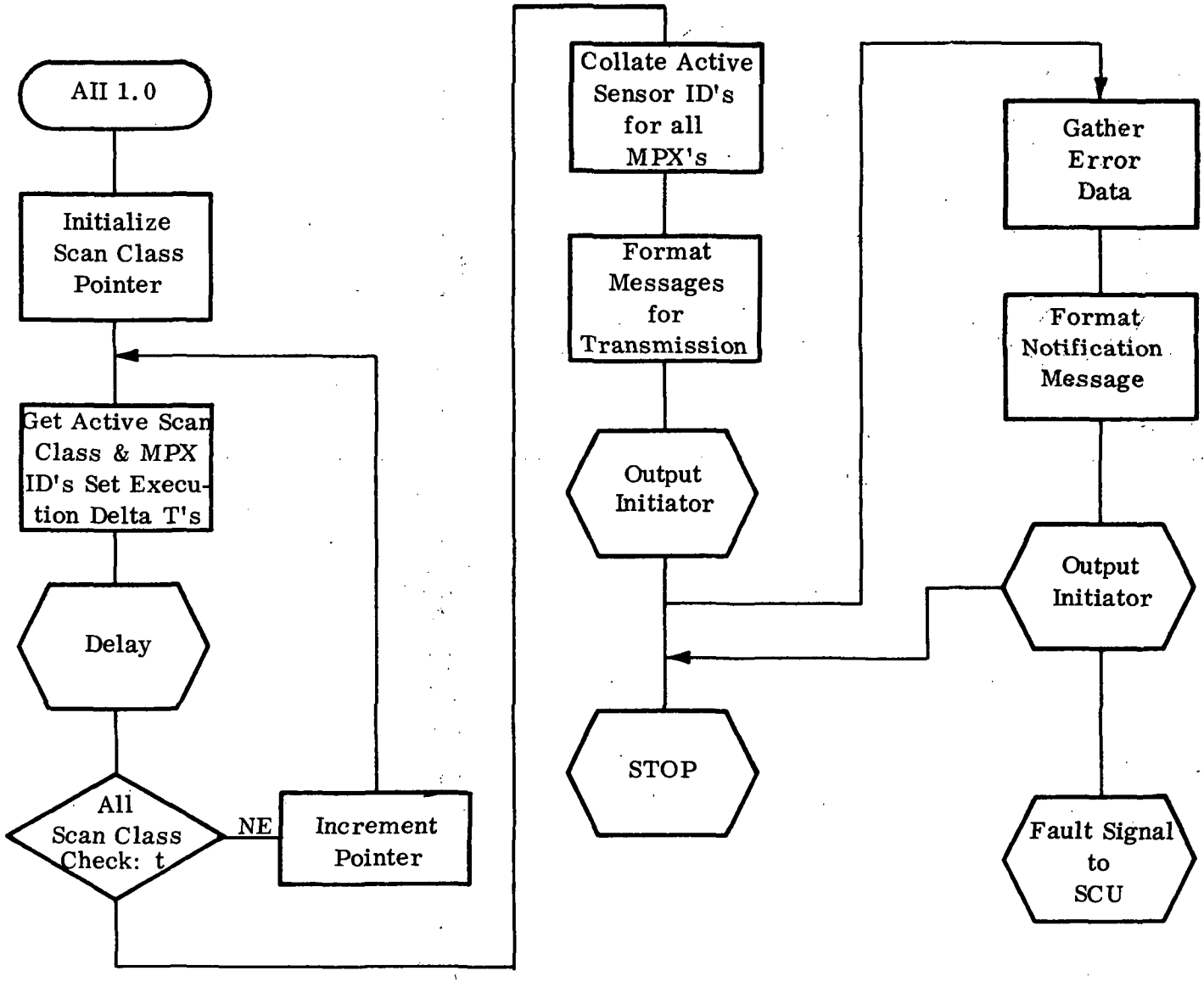


FIGURE C17. ANALOG INPUT INITIATOR

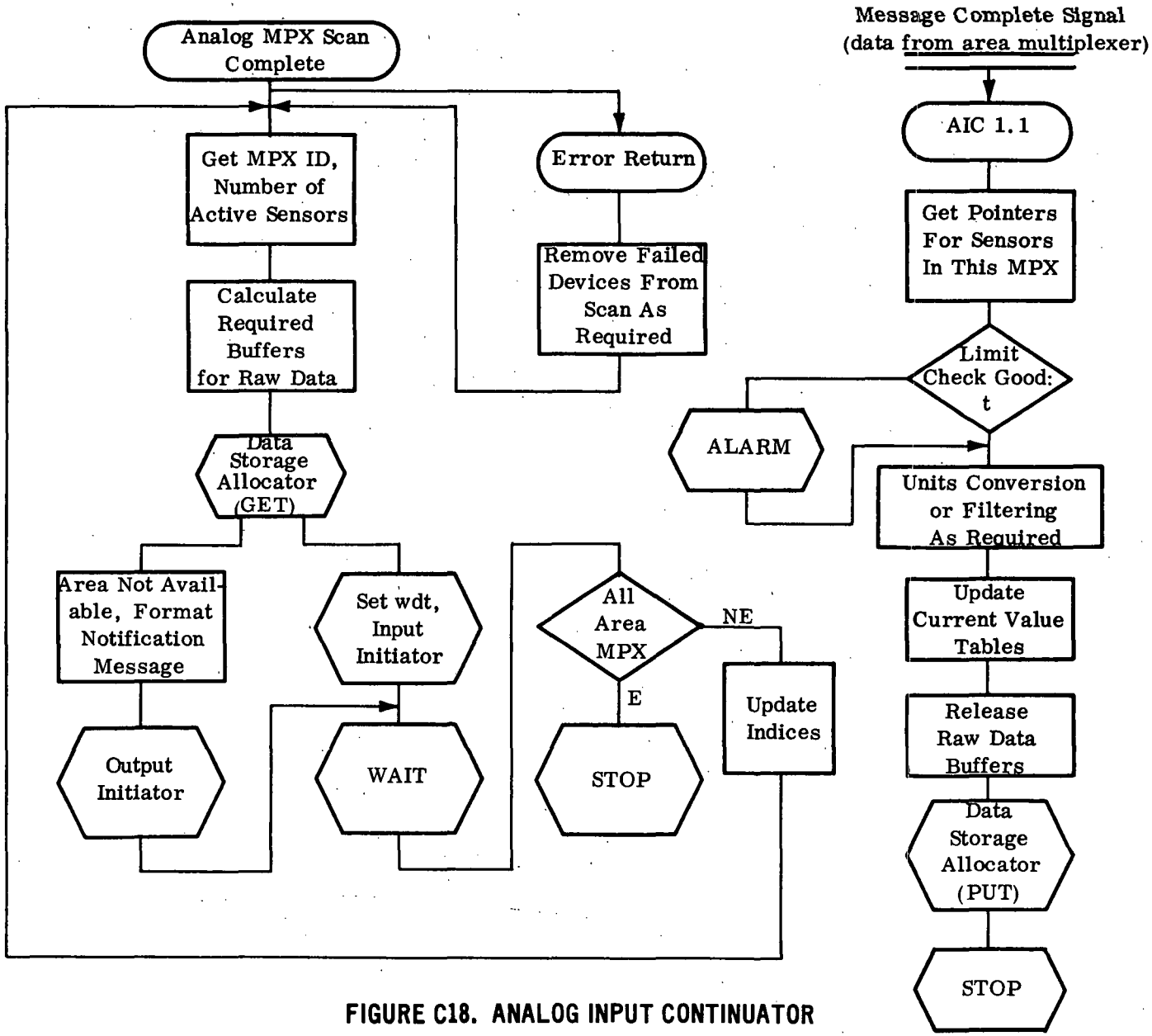


FIGURE C18. ANALOG INPUT CONTINUATOR

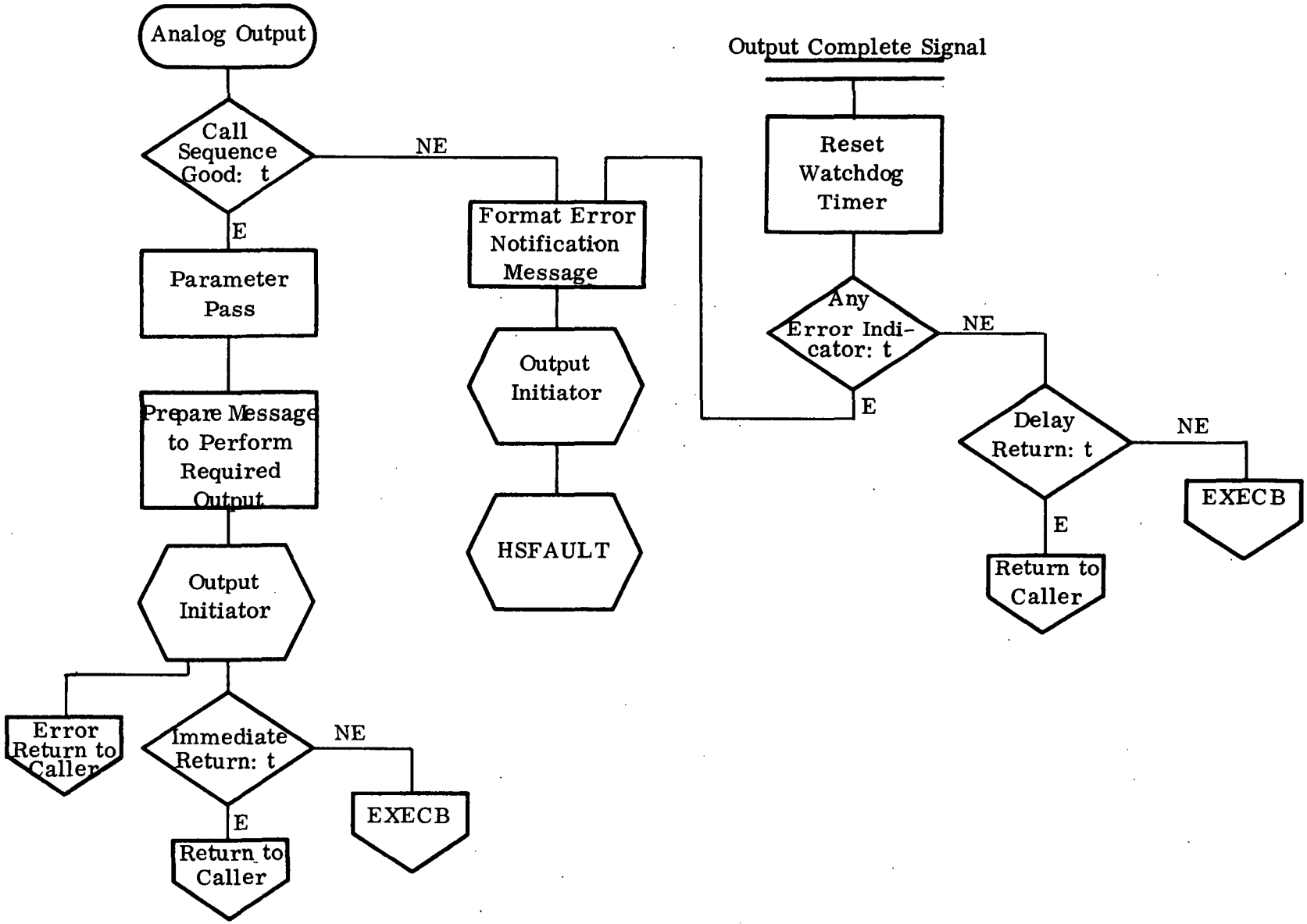


FIGURE C19. ANALOG OUTPUT INITIATOR/TERMINATOR

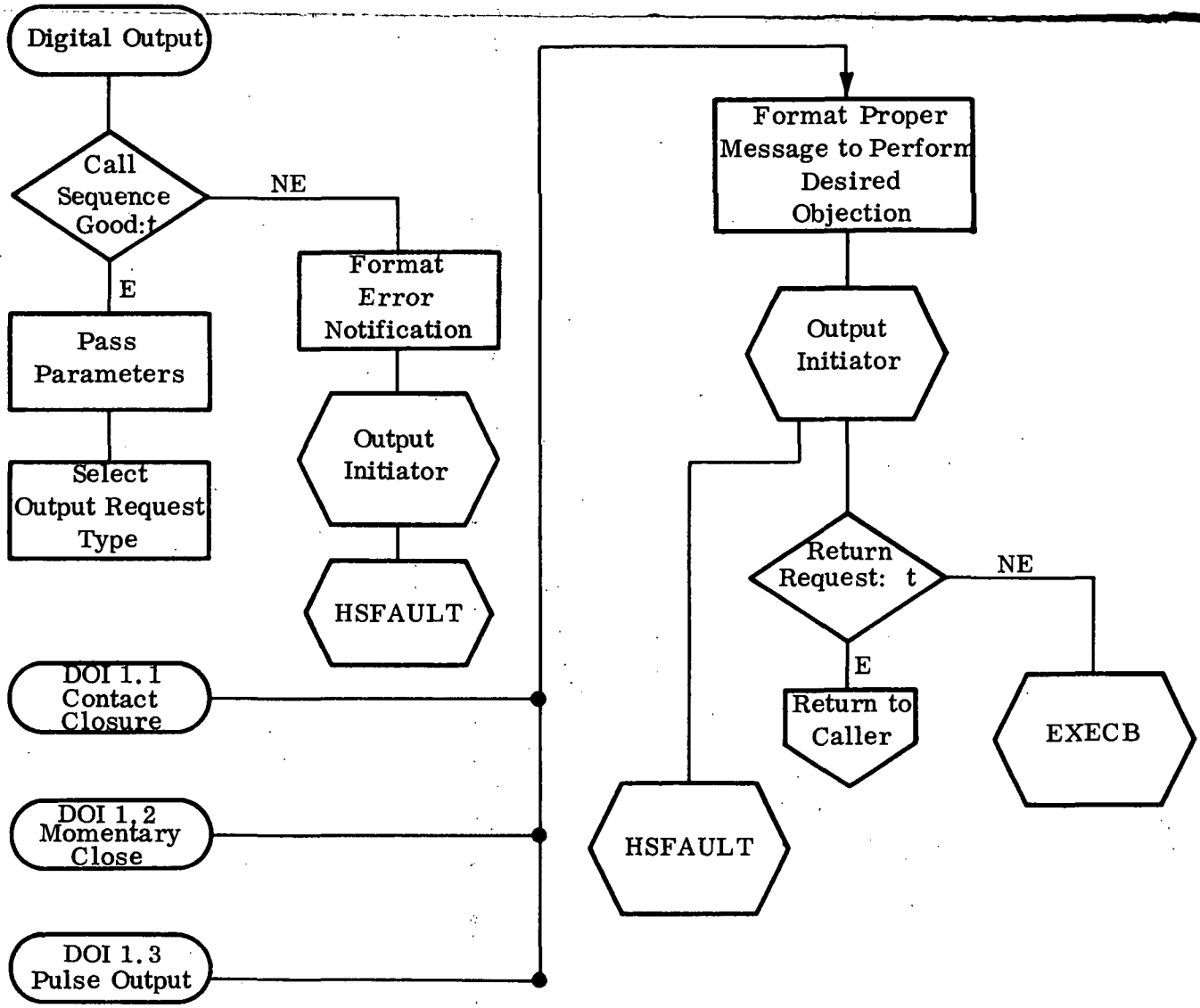


FIGURE C20. DIGITAL OUTPUT INITIATOR

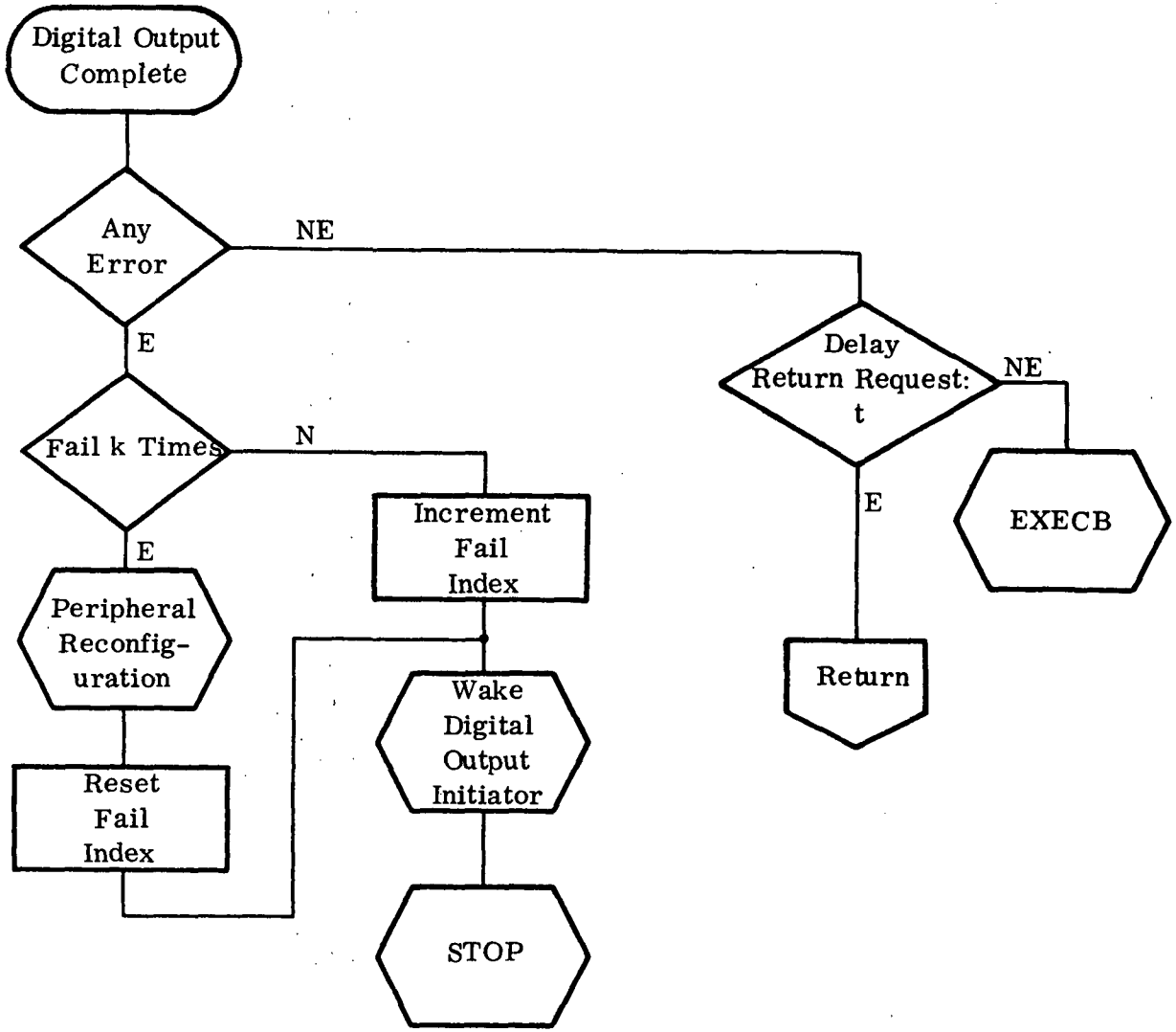


FIGURE C21. DIGITAL OUTPUT TERMINATOR

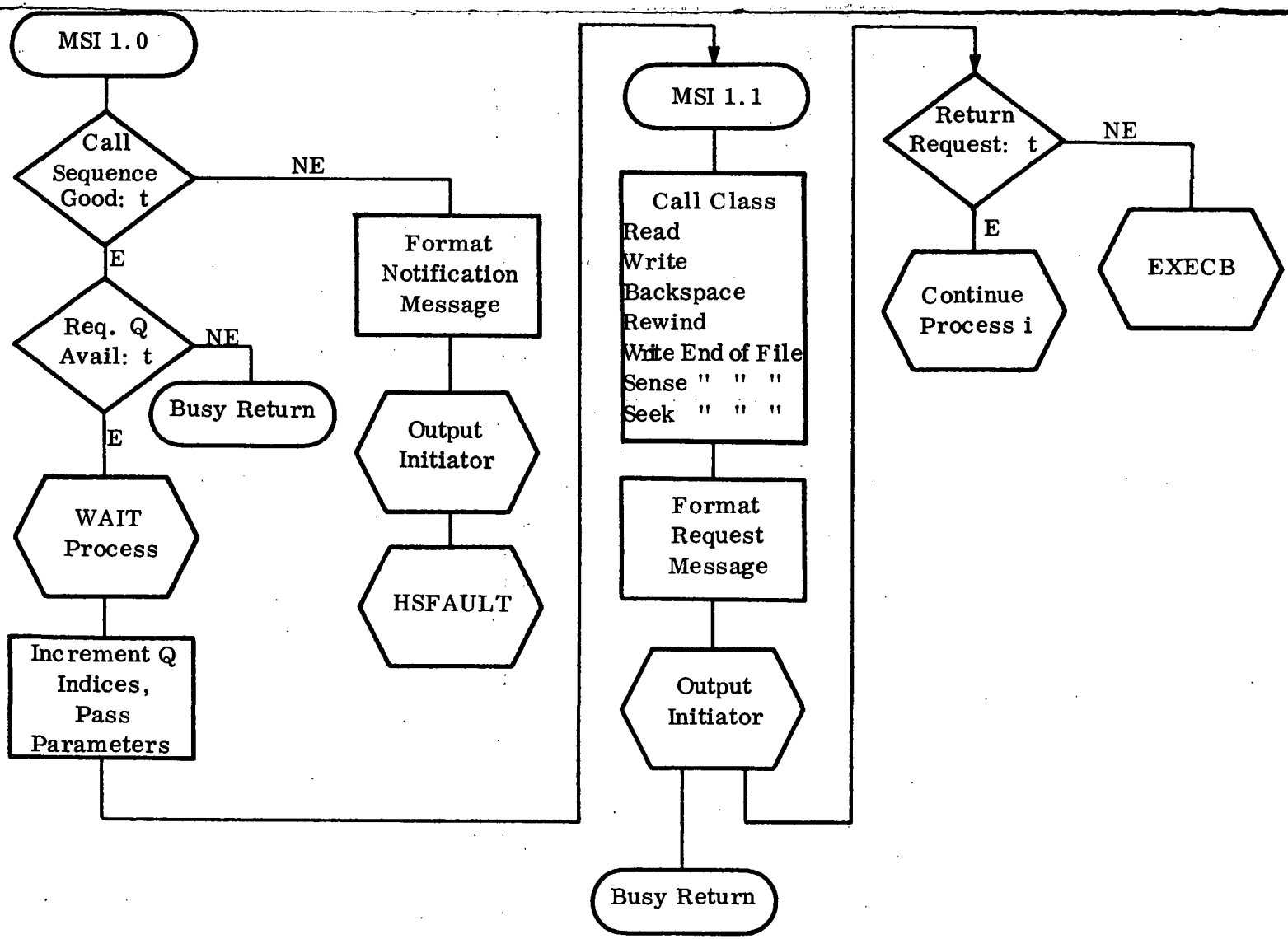


FIGURE C22. MASS STORE INITIATOR

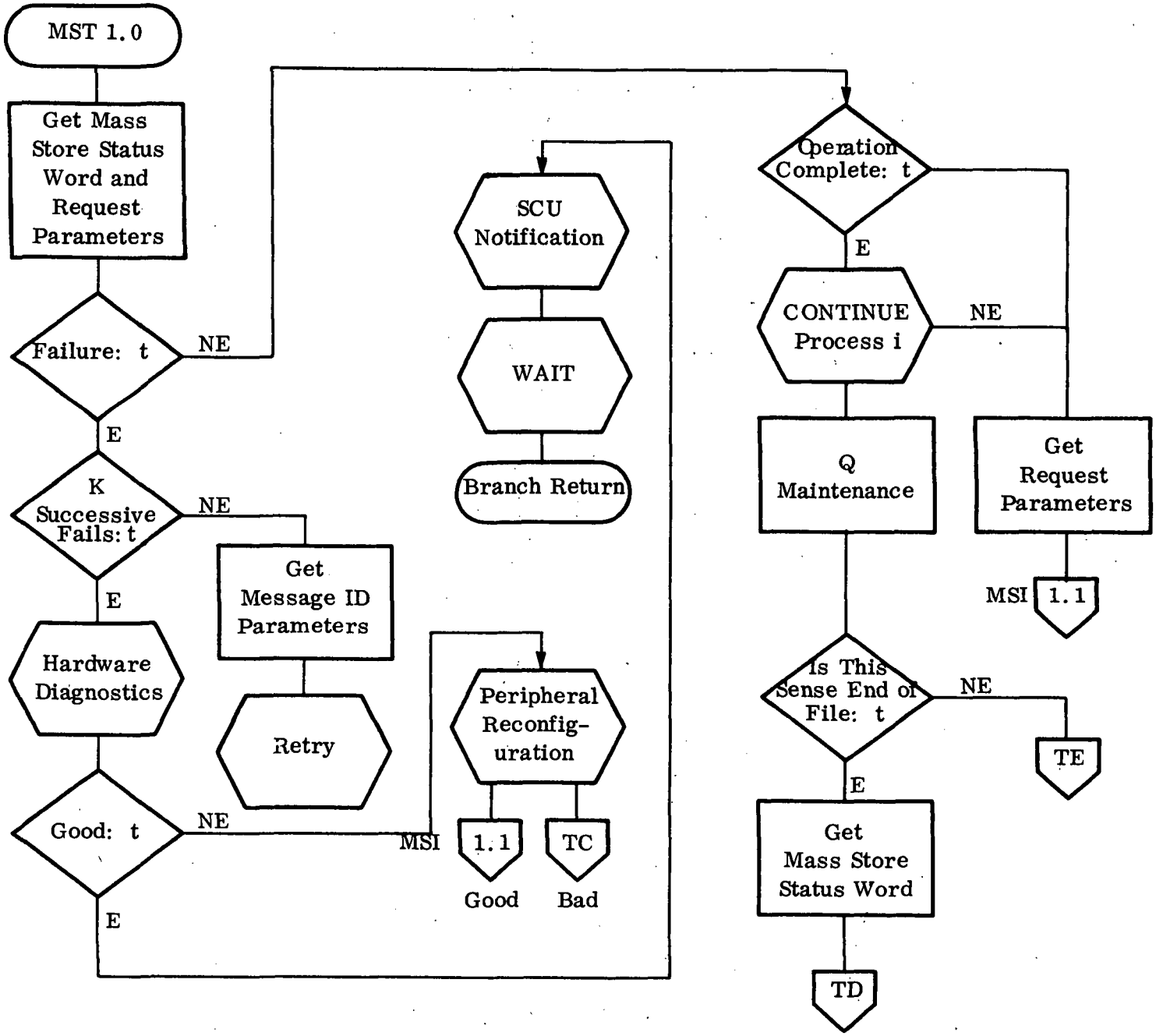


FIGURE C23. MASS STORE TERMINATOR (SHEET 1 OF 2)

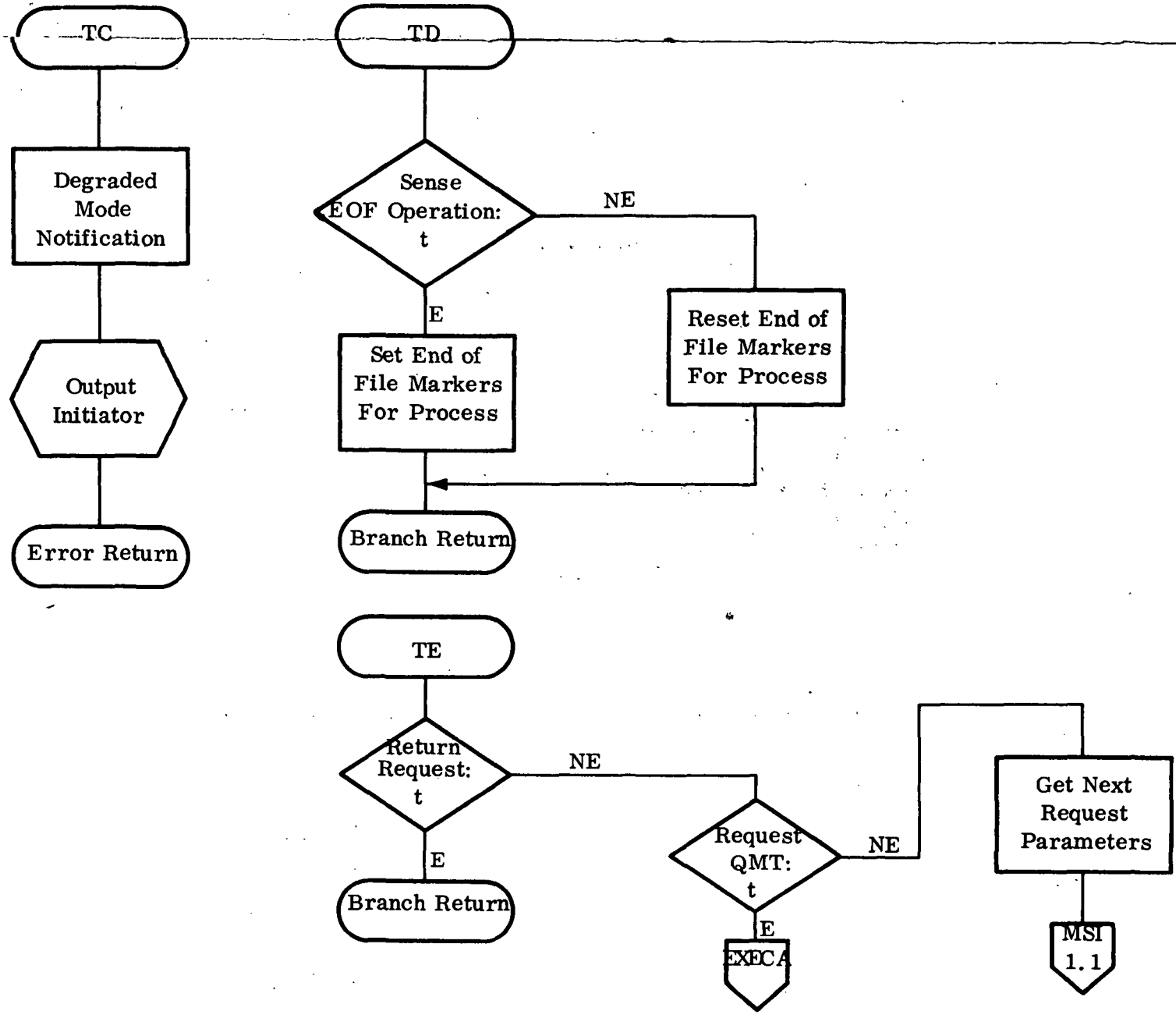


FIGURE C23. MASS STORAGE TERMINATOR (SHEET 2 OF 2)

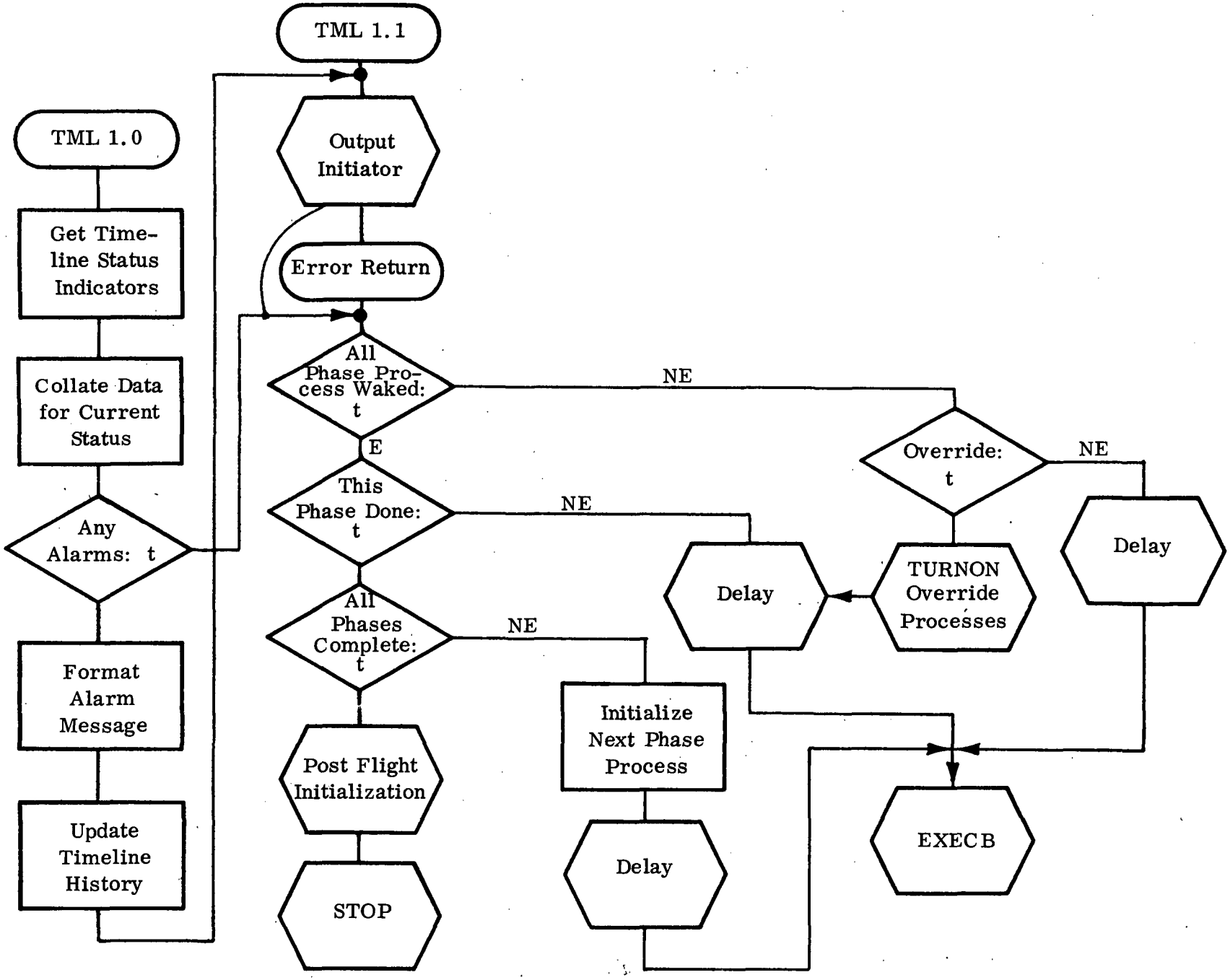


FIGURE C24. TIMELINE INTERPRETER/CONTROLLER (SHEET 1 OF 2)

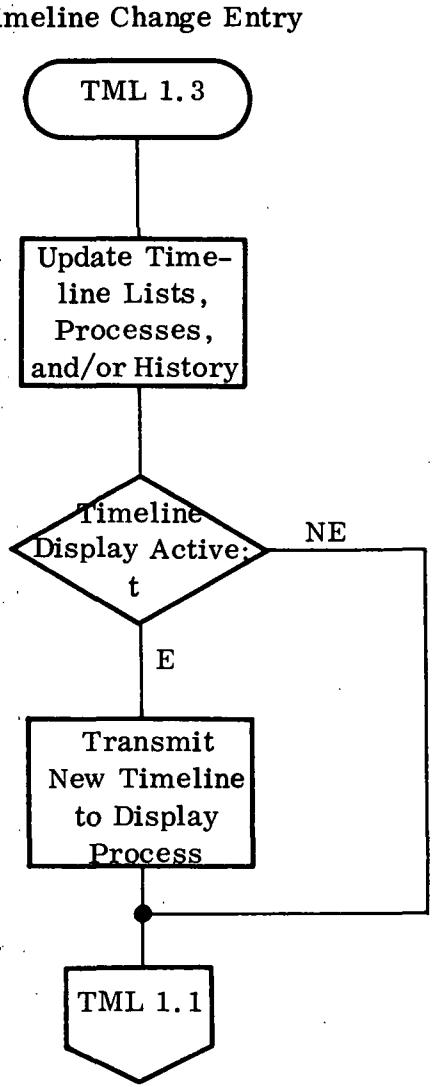


FIGURE C24. TIME CHANGE (SHEET 2 OF 2)

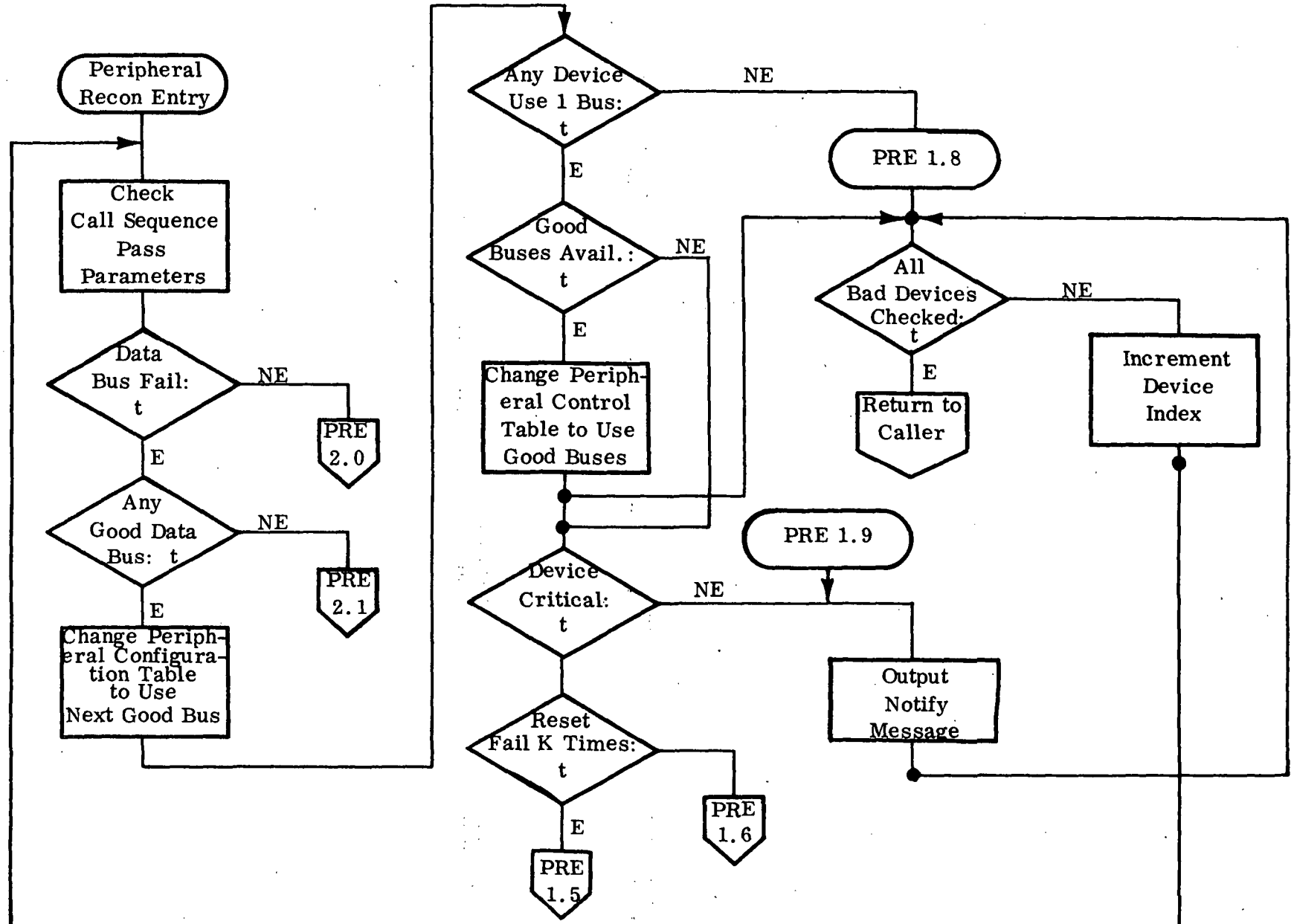


FIGURE C-25. PERIPHERAL RECONFIGURATION (SHEET 1 OF 3)

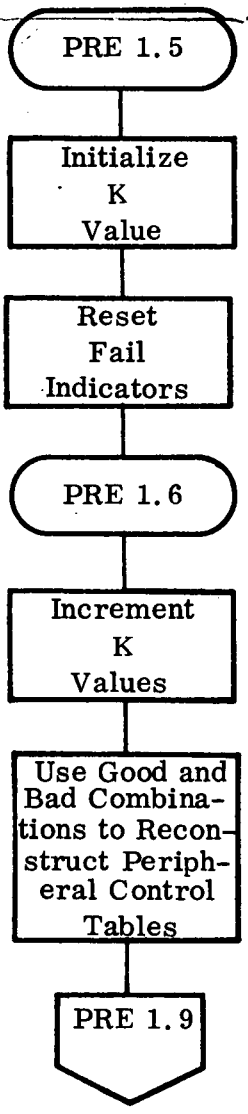


FIGURE C25. PERIPHERAL RECONFIGURATION (SHEET 2 OF 3)

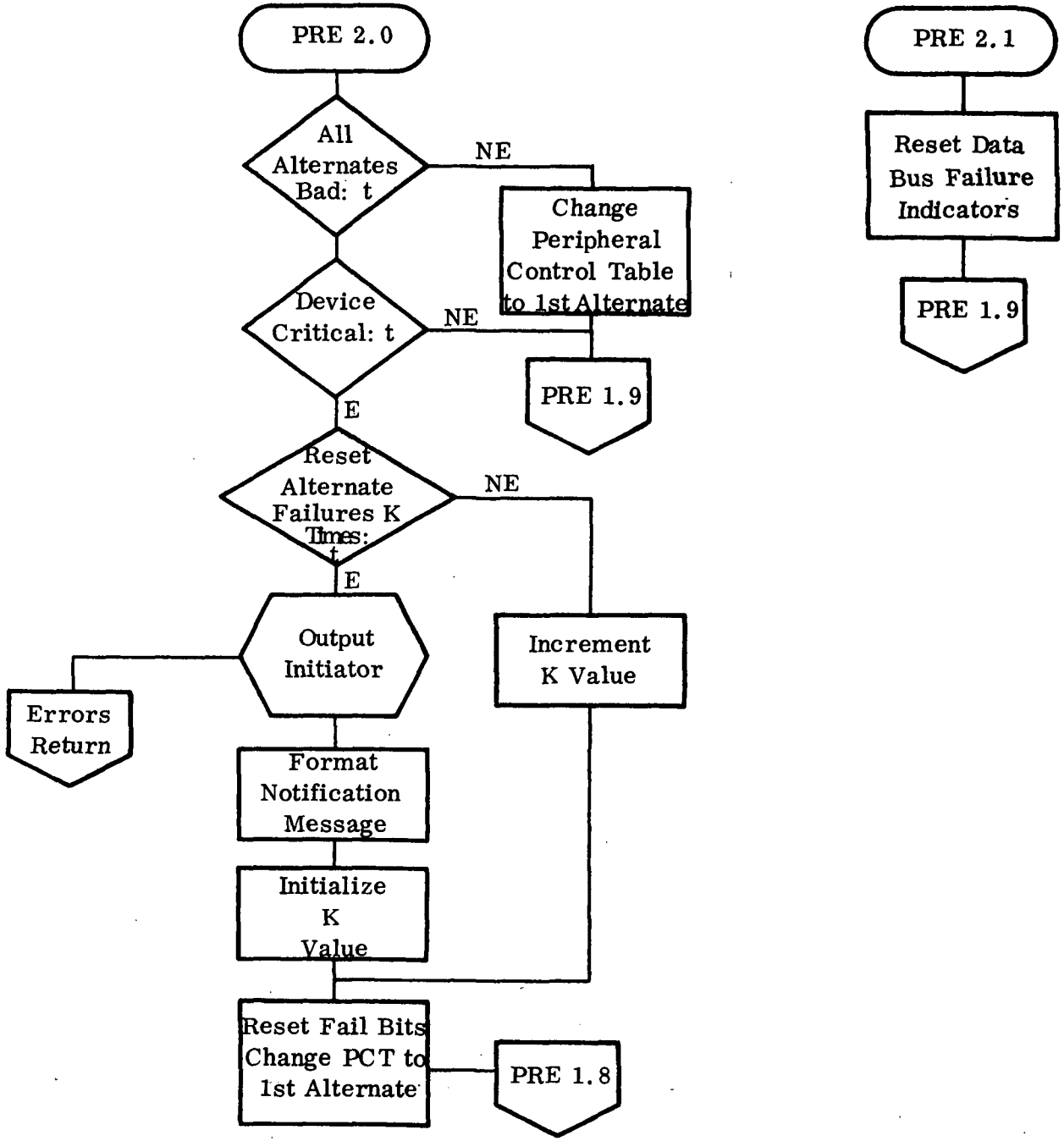


FIGURE C25. PERIPHERAL RECONFIGURATION (SHEET 3 OF 3)

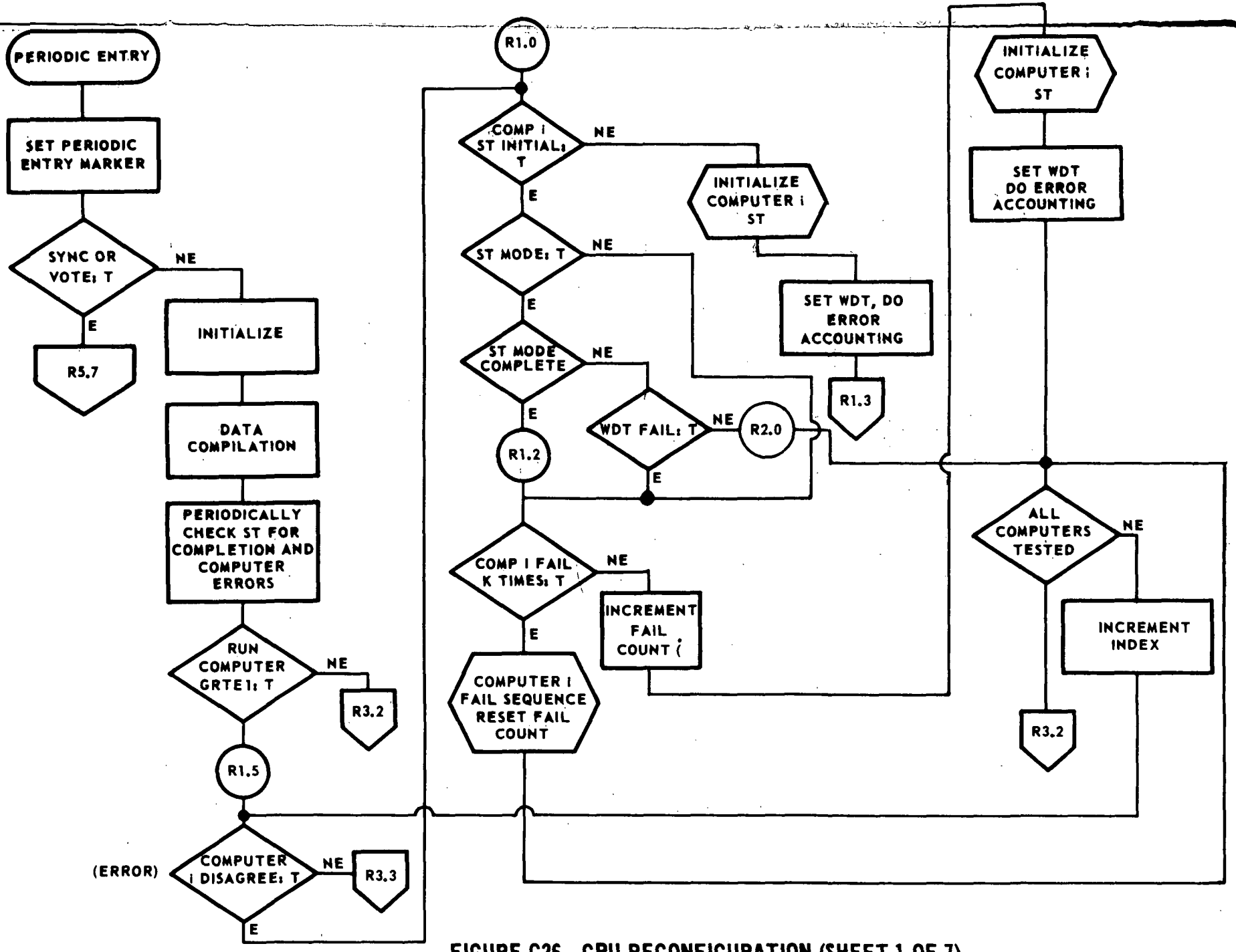


FIGURE C26. CPU RECONFIGURATION (SHEET 1 OF 7)

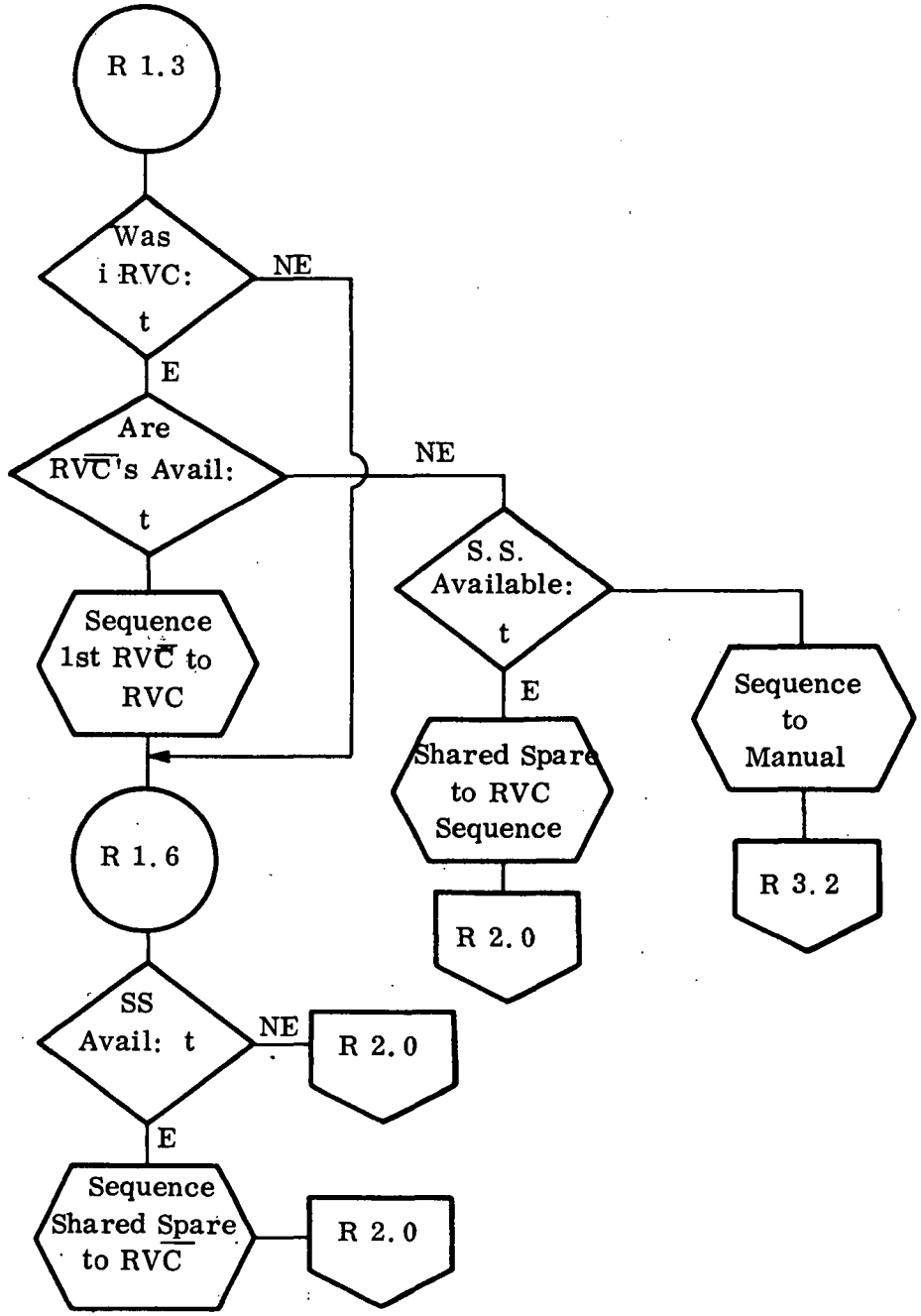


FIGURE C26. CPU RECONFIGURATION (SHEET 2 OF 7)

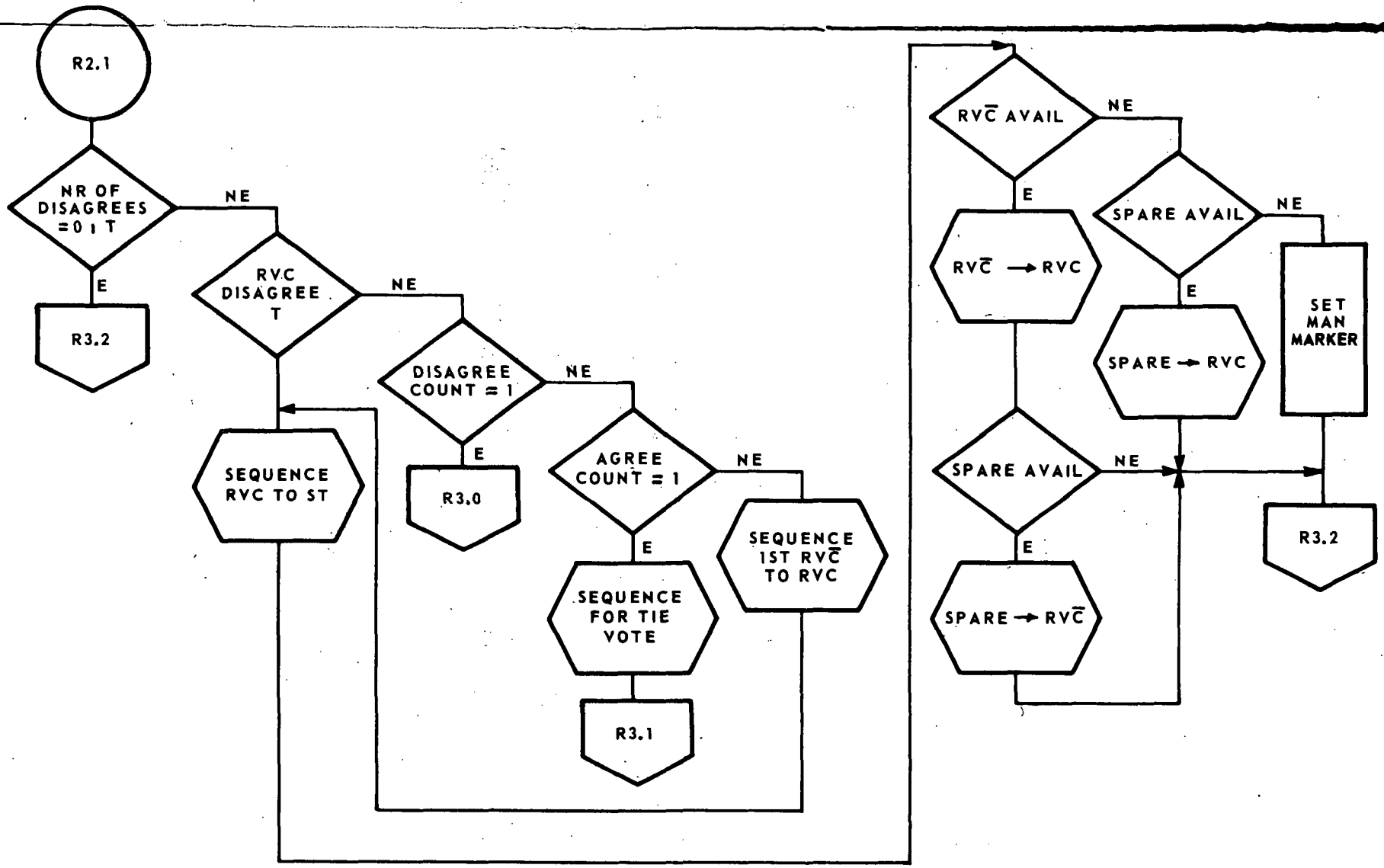


FIGURE C26. CPU RECONFIGURATION (SHEET 3 OF 7)

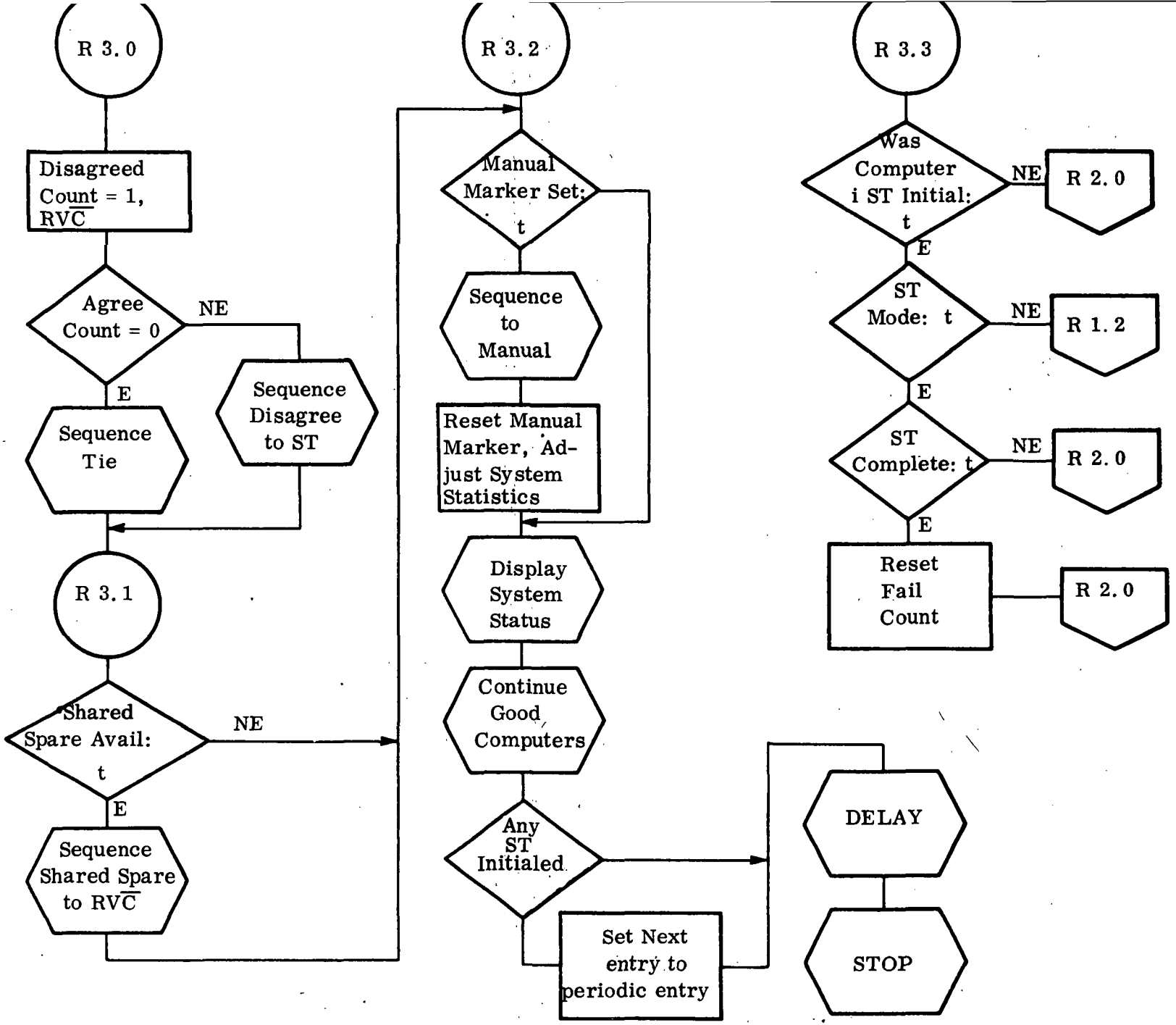


FIGURE C26. CPU RECONFIGURATION (SHEET 4 OF 7)

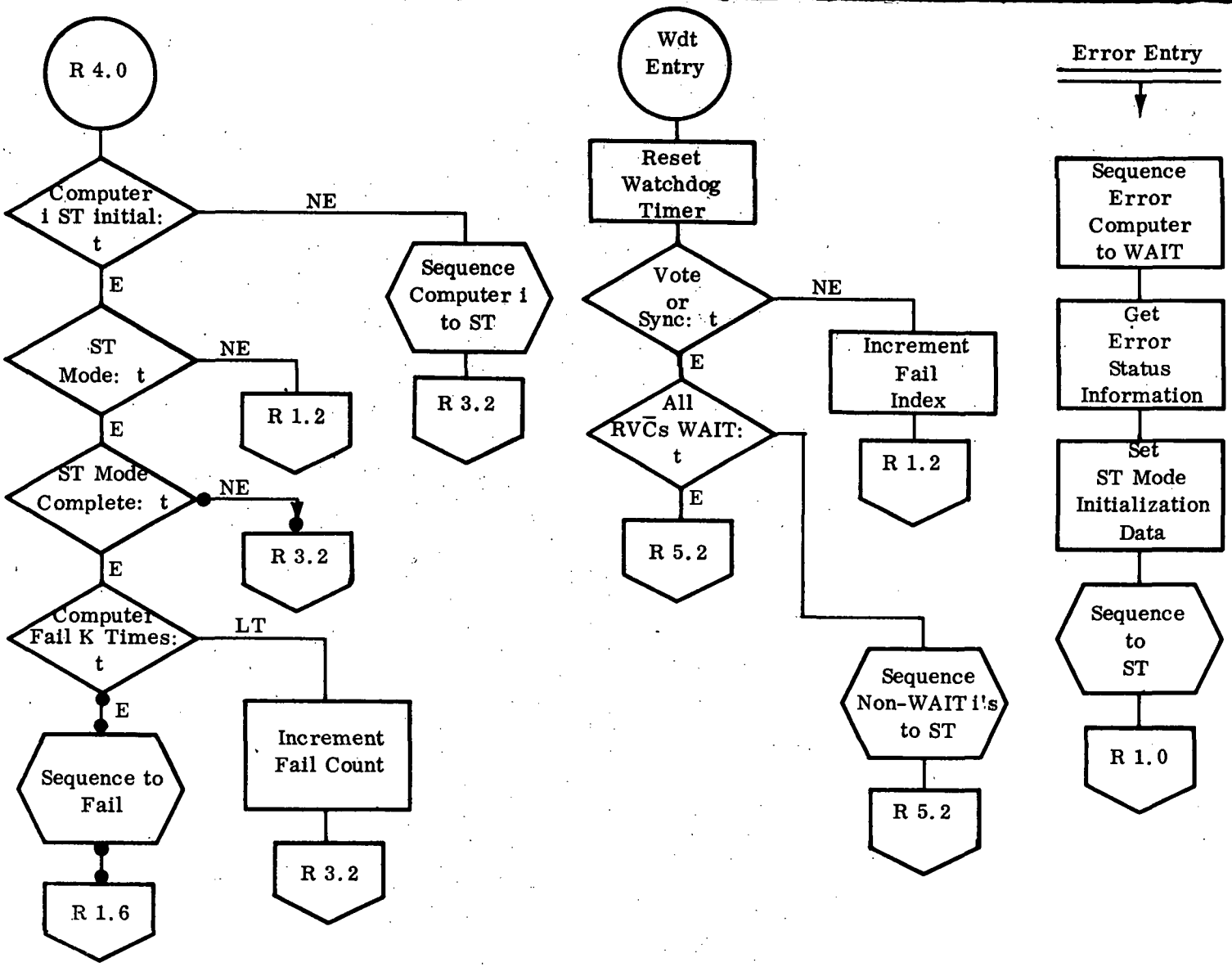


FIGURE C26. CPU RECONFIGURATION (SHEET 5 OF 7)

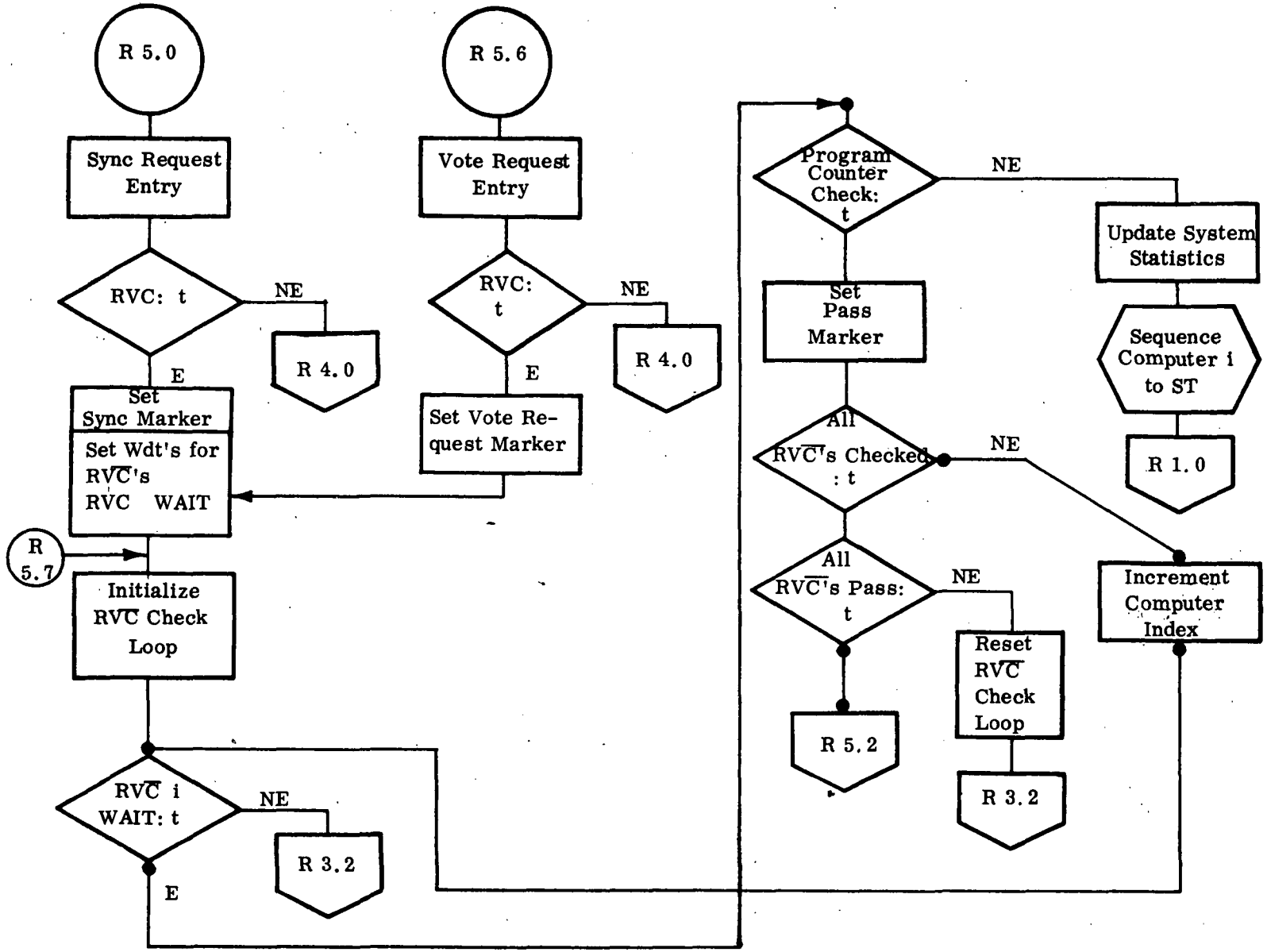


FIGURE C26. CPU RECONFIGURATION (SHEET 6 OF 7)

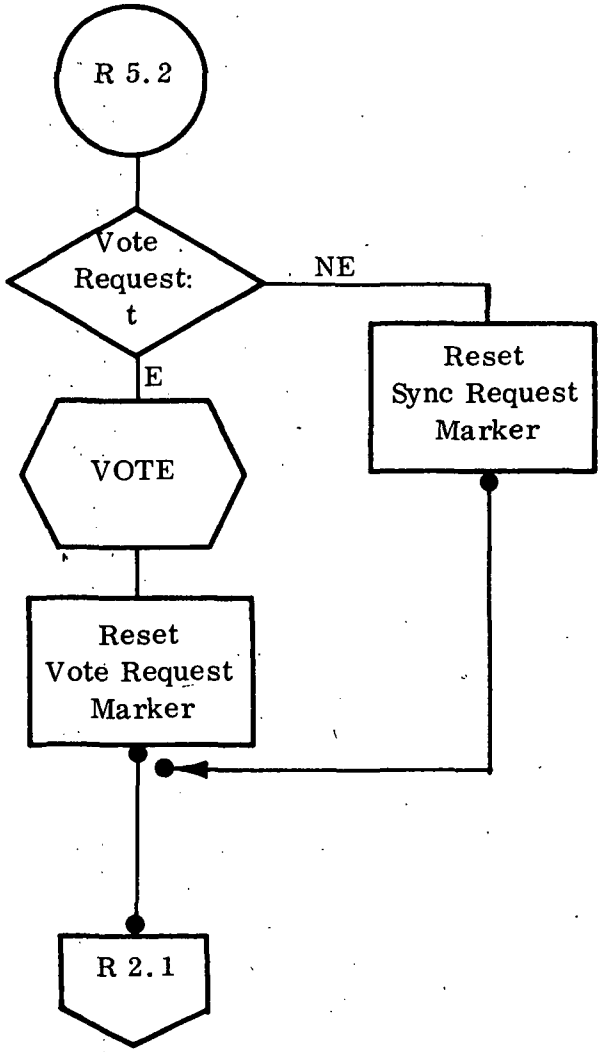


FIGURE C26. CPU RECONFIGURATION (SHEET 7 OF 7)

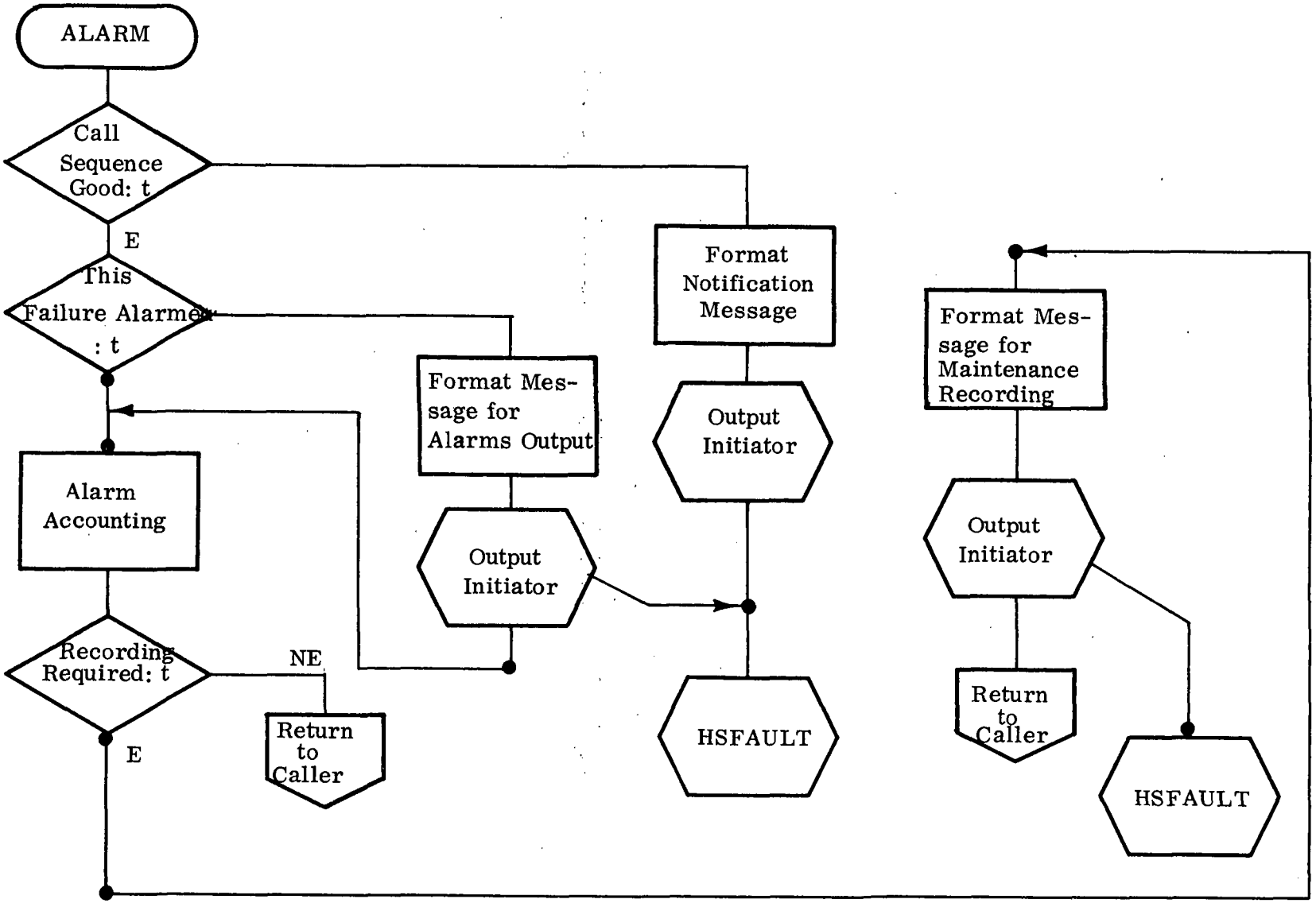


FIGURE C27. ALARM

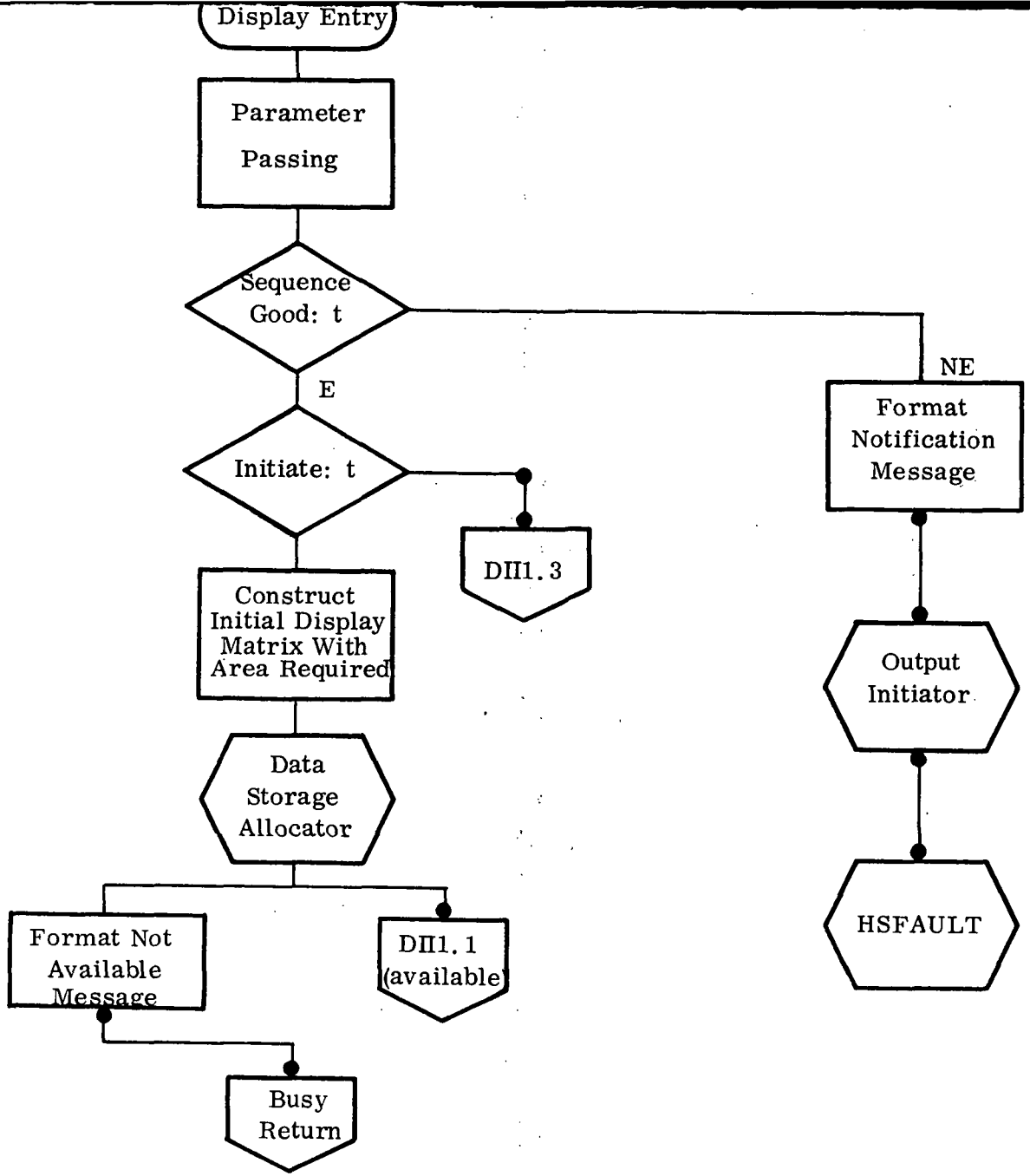


FIGURE C28. DISPLAY INITIATE / CANCEL (SHEET 1 OF 2)

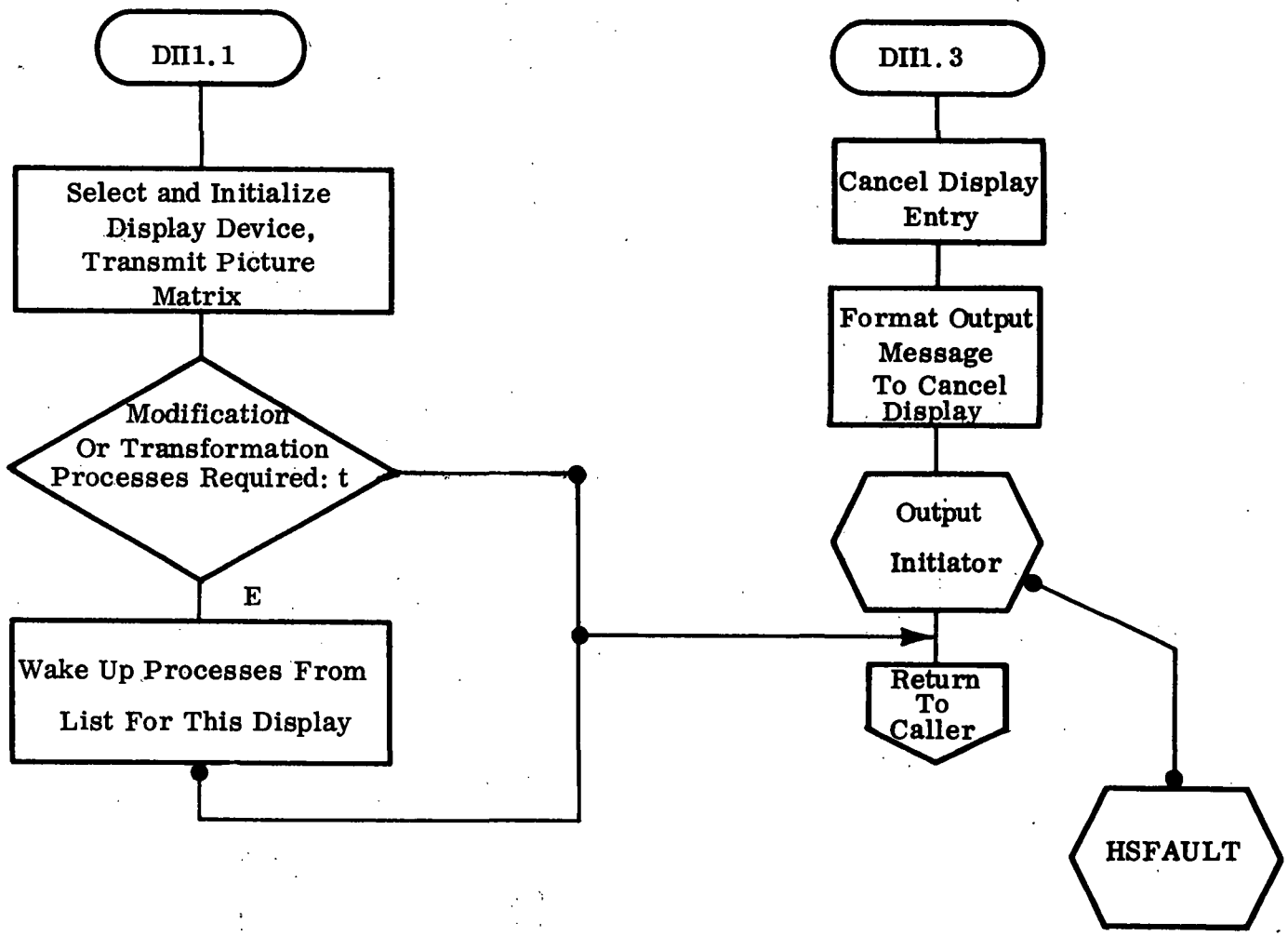


FIGURE C28. DISPLAY INITIATE/CANCEL (SHEET 2 OF 2)

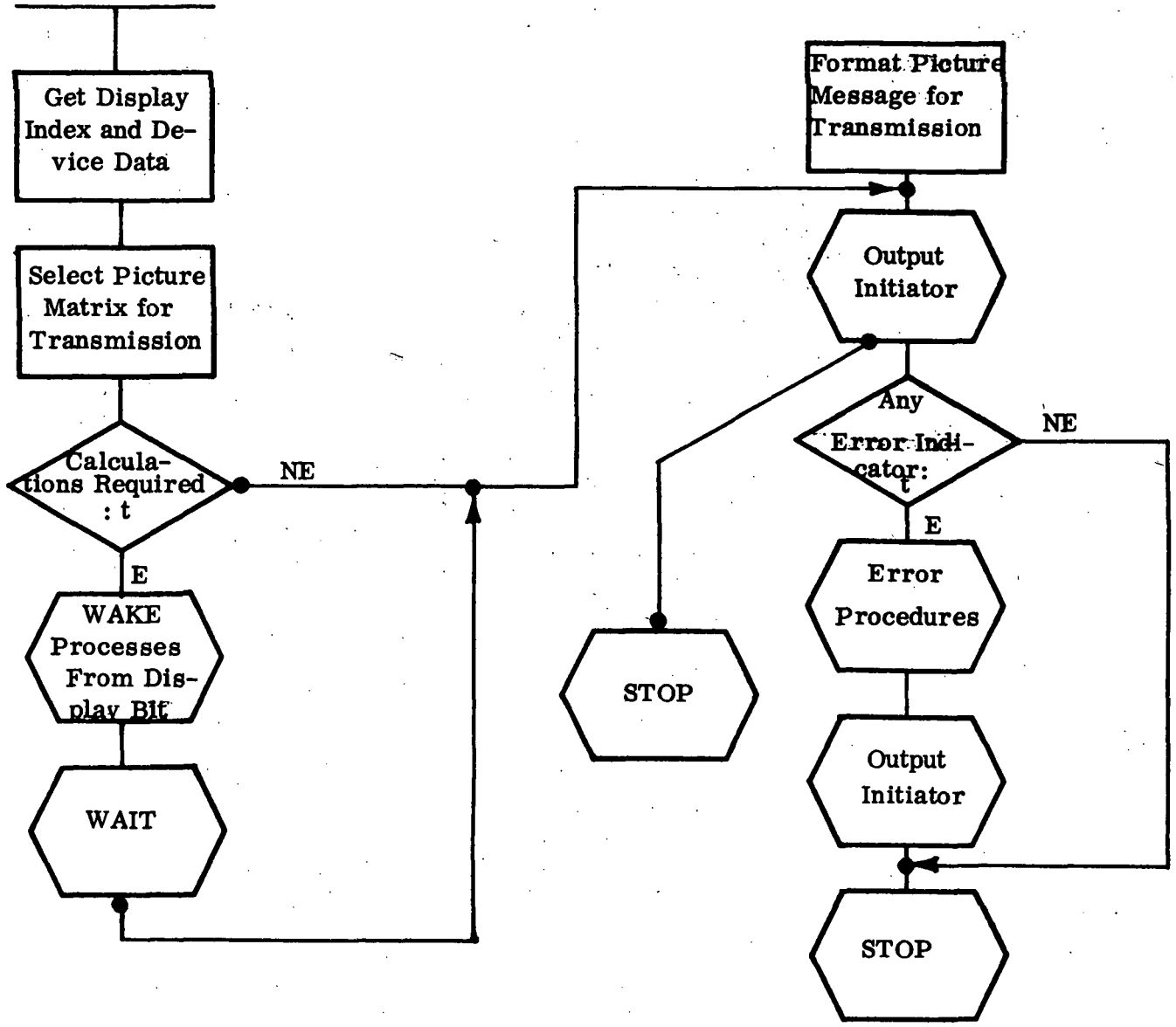
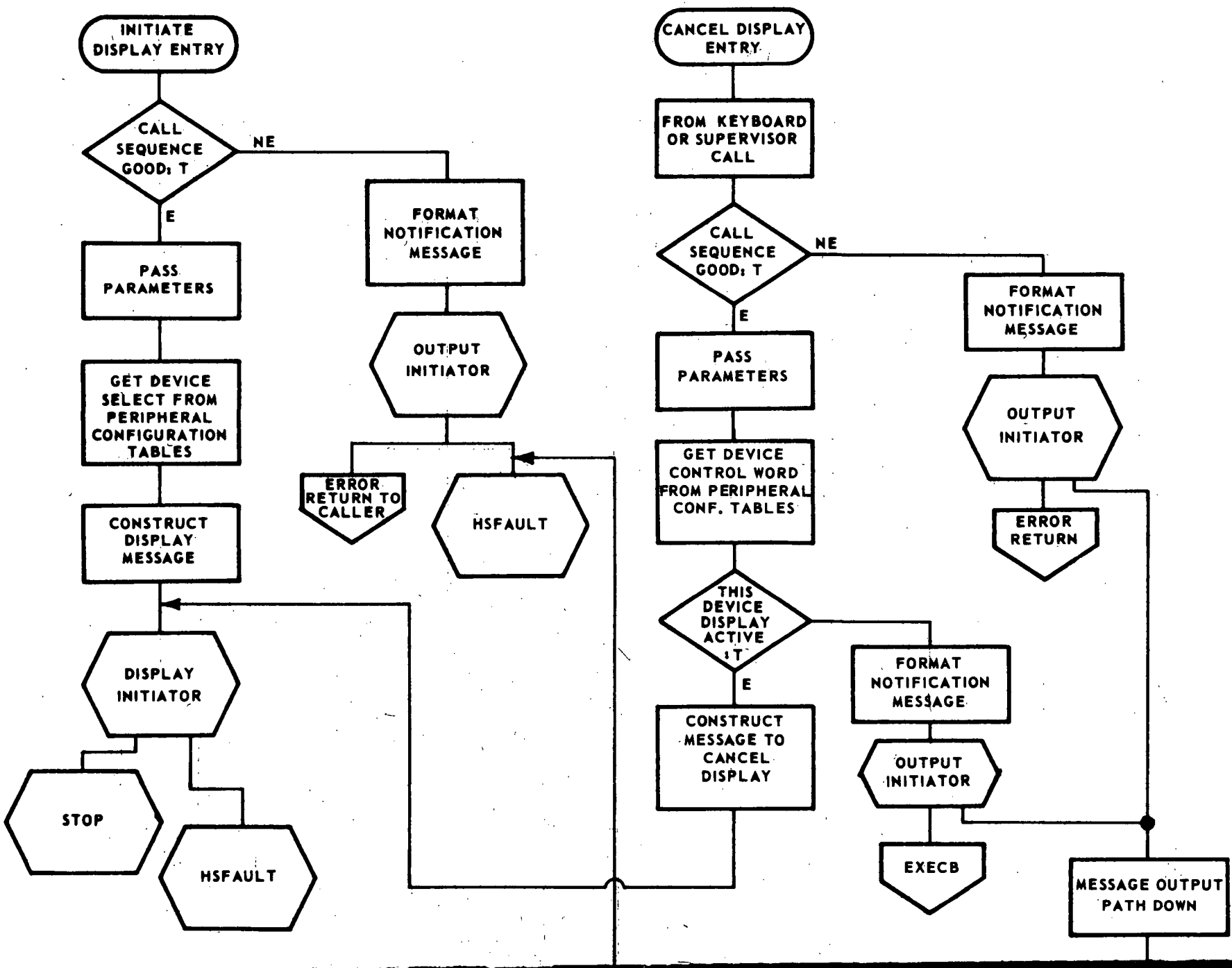


FIGURE C29. DISPLAY CONTINUATOR


INITIATE DISPLAY ENTRY
CALL SEQUENCE GOOD; T
NE
E
FORMAT NOTIFICATION MESSAGE
PASS PARAMETERS
OUTPUT INITIATOR
GET DEVICE SELECT FROM PERIPHERAL CONFIGURATION TABLES
ERROR RETURN TO CALLER
HSFAULT
CONSTRUCT DISPLAY MESSAGE
DISPLAY INITIATOR
STOP
HSFAULT
CANCEL DISPLAY ENTRY
FROM KEYBOARD OR SUPERVISOR CALL
CALL SEQUENCE GOOD; T
NE
E
FORMAT NOTIFICATION MESSAGE
PASS PARAMETERS
OUTPUT INITIATOR
GET DEVICE CONTROL WORD FROM PERIPHERAL CONF. TABLES
ERROR RETURN
THIS DEVICE DISPLAY ACTIVE ; T
E
FORMAT NOTIFICATION MESSAGE
CONSTRUCT MESSAGE TO CANCEL DISPLAY
OUTPUT INITIATOR
EXECB
MESSAGE OUTPUT PATH DOWN

NASA-Langley, 1971 – 08
Coml., Newport News, Va.

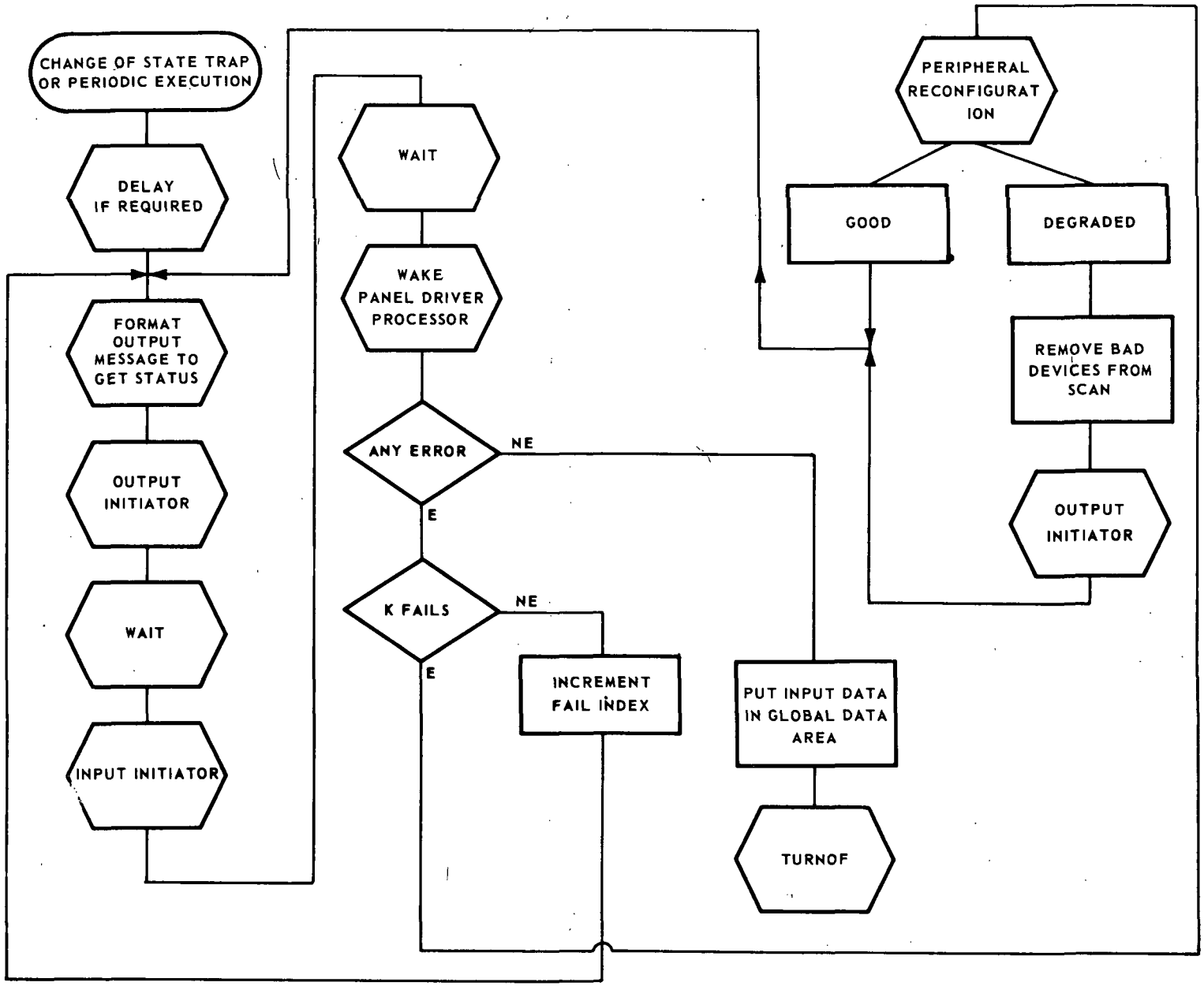


FIGURE C31. PANEL STATUS INPUT

"The aeronautical and space activities of the United States shall be conducted so as to contribute . . . to the expansion of human knowledge of phenomena in the atmosphere and space. The Administration shall provide for the widest practicable and appropriate dissemination of information concerning its activities and the results thereof."

—NATIONAL AERONAUTICS AND SPACE ACT OF 1958

NASA SCIENTIFIC AND TECHNICAL PUBLICATIONS

TECHNICAL REPORTS: Scientific and technical information considered important, complete, and a lasting contribution to existing knowledge.

TECHNICAL NOTES: Information less broad in scope but nevertheless of importance as a contribution to existing knowledge.

TECHNICAL MEMORANDUMS: Information receiving limited distribution because of preliminary data, security classification, or other reasons.

CONTRACTOR REPORTS: Scientific and technical information generated under a NASA contract or grant and considered an important contribution to existing knowledge.

TECHNICAL TRANSLATIONS: Information published in a foreign language considered to merit NASA distribution in English.

SPECIAL PUBLICATIONS: Information derived from or of value to NASA activities. Publications include conference proceedings, monographs, data compilations, handbooks, sourcebooks, and special bibliographies.

TECHNOLOGY UTILIZATION PUBLICATIONS: Information on technology used by NASA that may be of particular interest in commercial and other non-aerospace applications. Publications include Tech Briefs, Technology Utilization Reports and Technology Surveys.

Details on the availability of these publications may be obtained from:

SCIENTIFIC AND TECHNICAL INFORMATION OFFICE

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

Washington, D.C. 20546