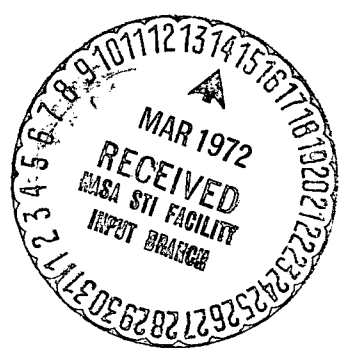


Report No. 72-0005
Contract NAS8-26990



FLIGHT PROGRAM LANGUAGE REQUIREMENTS

VOLUME III

APPENDICES

CR-123569

| | |
|--|--------------|
| (NASA-CR-123569) FLIGHT PROGRAM LANGUAGE REQUIREMENTS. VOLUME 3: APPENDICES (M&S Computing, Inc.) 7 Mar. 1972 262 p CSCL 09B | N72-21201 |
| | Unclas 15214 |
| | G3/08 |

March 7, 1972

Prepared for:

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
George C. Marshall Space Flight Center
Marshall Space Flight Center, Alabama 35812

M&S COMPUTING, INC.

Pages - 262

Reproduced by
NATIONAL TECHNICAL
INFORMATION SERVICE
U S Department of Commerce
Springfield VA 22151

CAT-08

PREFACE

This report summarizes the efforts and results of a study to establish requirements for a flight programming language for future onboard computer applications. This study was performed by M&S Computing under contract NAS8-26990 from the Marshall Space Flight Center of NASA. The technical monitor was Mr. Richard Jenke, S&E-CSE-LI.

Several government-sponsored study and development efforts have been directed toward design and implementation of high level programming languages suitable for future aerospace applications. As a result, several different languages were available as potential candidates for future NASA flight programming efforts. The study centered around an evaluation of the four most pertinent existing aerospace languages. Evaluation criteria were established and selected kernels from the current Saturn V and Skylab Flight Programs were used as benchmark problems for sample coding. An independent review of the language specifications incorporated anticipated future programming requirements into the evaluation. A set of detailed language requirements was synthesized from these activities.

This report is the final report of the study and is provided in three volumes. This third volume contains the report appendices, which describe the benchmark problems coded and provide listings of the benchmark coding.

Distribution of this report is provided in the interest of information exchange and should not be construed as endorsement by NASA of the material presented. Responsibility for the contents resides with the organization that prepared it.

Participating personnel were:

T. T. Schansman
R. E. Thurber
L. C. Keller
W. M. Rogers

Approved by:

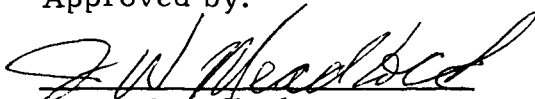

J. W. Meadlock

TABLE OF CONTENTS

| <u>Appendix</u> | | <u>Page No.</u> |
|-----------------|------------------------------------|-----------------|
| A | Flight Program Kernel Descriptions | 1 |
| B | Flight Program Kernel Coding | 81 |

APPENDIX A

FLIGHT PROGRAM KERNEL DESCRIPTIONS

This appendix contains flowcharts and narrative descriptions of the flight program kernels which were coded. The descriptions also discuss certain assumptions made during coding of the kernels and the unique language requirements imposed by each kernel. The actual coding of the kernels is found in Appendix B.

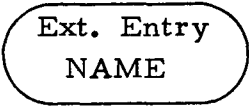
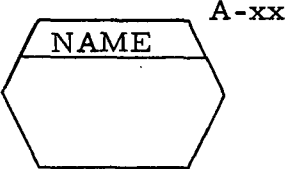

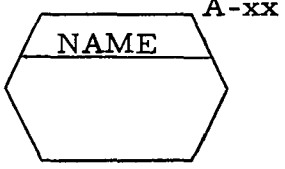

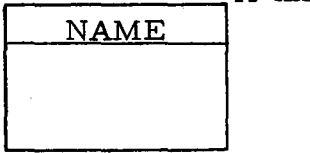
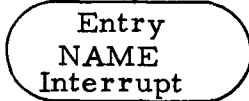
Each kernel description is a separate paragraph of this appendix, and a kernel flowchart is included as a figure at the end of the paragraph. Kernel names and associated paragraph and flowchart figure numbers are listed below:

| <u>Paragraph</u> | <u>Kernel Name</u> | <u>Flowchart Figure</u> |
|------------------|---------------------------|-------------------------|
| A. 1 | Initialization | A-1 (a-b) |
| A. 2 | Interrupt Processor | A-2 (a-d) |
| A. 3 | Non-Interrupt Sequencer | A-3 |
| A. 4 | Periodic Processor | A-4 |
| A. 5 | Events Processor | A-5 |
| A. 6 | Iterative Guidance Mode | A-6 (a-d) |
| A. 7 | Digital Command System | A-7 (a-c) |
| A. 8 | Accelerometer Processing | A-8 (a-d) |
| A. 9 | Minor Loop | A-9 (a-d) |
| A. 10 | Switch Selector Processor | A-10 (a-n) |
| A. 11 | ATM Task Keying | A-11 |

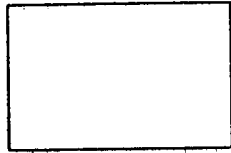
Separate pages of multiple-page flowcharts are designated by lower case letters appended to the figure numbers. These are indicated above.

As a documentation aid, paragraph A. 12 contains glossaries of the names used in the program listings of Appendix B. The glossaries include brief explanations of each name.

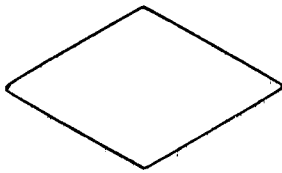
Special flowchart symbology has been used to identify and cross-reference program kernels and the various types of partitioning within kernels. The following depicts and explains this symbology. The "Entry Point" column shows the symbol used for entry into each type of program block, and the corresponding "Calling Symbol" indicates how that type of program block is called from some other flowchart. The label "A-xx" references the flowchart where the "called" program block is described. If there is no label, the program block was not coded and no flowchart is provided.

| <u>Entry Point</u> | <u>Calling Symbol</u> | <u>Type of Program Block</u> |
|---|---|--|
|  |  | External entry point to a program kernel. Called from some other kernel. |
|  |  | Internal entry point to a subprogram within a kernel. Called only from within the kernel. |
|  |  | Indicates a program block which is coded in-line on the coding sheets but is shown on a separate flowchart solely for clarity of documentation. It is not a separate subprogram. |
|  | (None) | Entry to logic which is executed on occurrence of the interrupt NAME. |

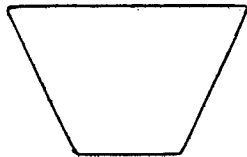
Flowchart symbols internal to a program block have conventional interpretations as follows:



Process



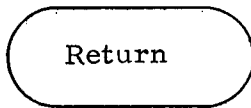
Decision



Input/Output



On-Page Connector



Return to calling
program at point of
call

The term "Note x" on a flowchart identifies a note at the end of the kernel descriptions.

A.1 Initialization

A.1.1 Description of Operation

A certain amount of initialization must be performed for any type of computing system. For a flight program, initialization involves setting up both program data storage areas and hardware registers. For example, data variables for an integration scheme must be assigned initial values and program switches must be setup to properly control program execution. Certain hardware registers such as accelerometers and the real time clock must be read to obtain initial values while others such as program controlled timers must be loaded with an initial value.

While it is true that program data storage areas could be initialized at program generation time, it is usually desirable to perform the initialization in real time under program control to eliminate the need for reloading the program each time it is to be restarted. In addition, a certain amount of reinitialization must be performed dynamically as the transition is made from one mission phase to another.

Two entry points exist for the Initialization kernel. The first is used when the program is entered from Prepare-to-Launch and performs overall system initialization. The second is used at the end of each mission phase to perform the initialization for the next phase.

A.1.2 Unique Language Characteristics Required

The manner in which initialization is performed depends greatly upon the organization of the data base. Data which is defined as "local" and is contained within an application module would require an initialization pass to be made through the module unless special techniques were provided by the language to enable such data to be externally referenced by a centralized initialization program. A separate initialization pass through each module forces an undesirable decentralization of the function, so the best choice within the capabilities of the selected languages is to put all data which must be initialized into a common data pool (Compool), so it can be accessed by the Initialization module. However, since almost all of the Saturn flight program data gets initialized, this design would leave very little data local to the modules and would reduce the opportunities to describe local and global data in the languages. Therefore, some of this data

remains local to the module and the details of the application module data initialization were not coded. This decision was influenced by the fact that the detailed coding is primarily restricted to a set of assignment statements, and data item assignment capabilities in the languages are well exercised in other kernels.

A. 1. 3 Flowchart Notes

Note 1

For HAL and CLASP the phase control logic beginning at GP002 had to be made a separate program module since it was common to both EGP0 and MPA00. This was necessitated by language restrictions which limit a program module to a single entry point.

INITIALIZATION

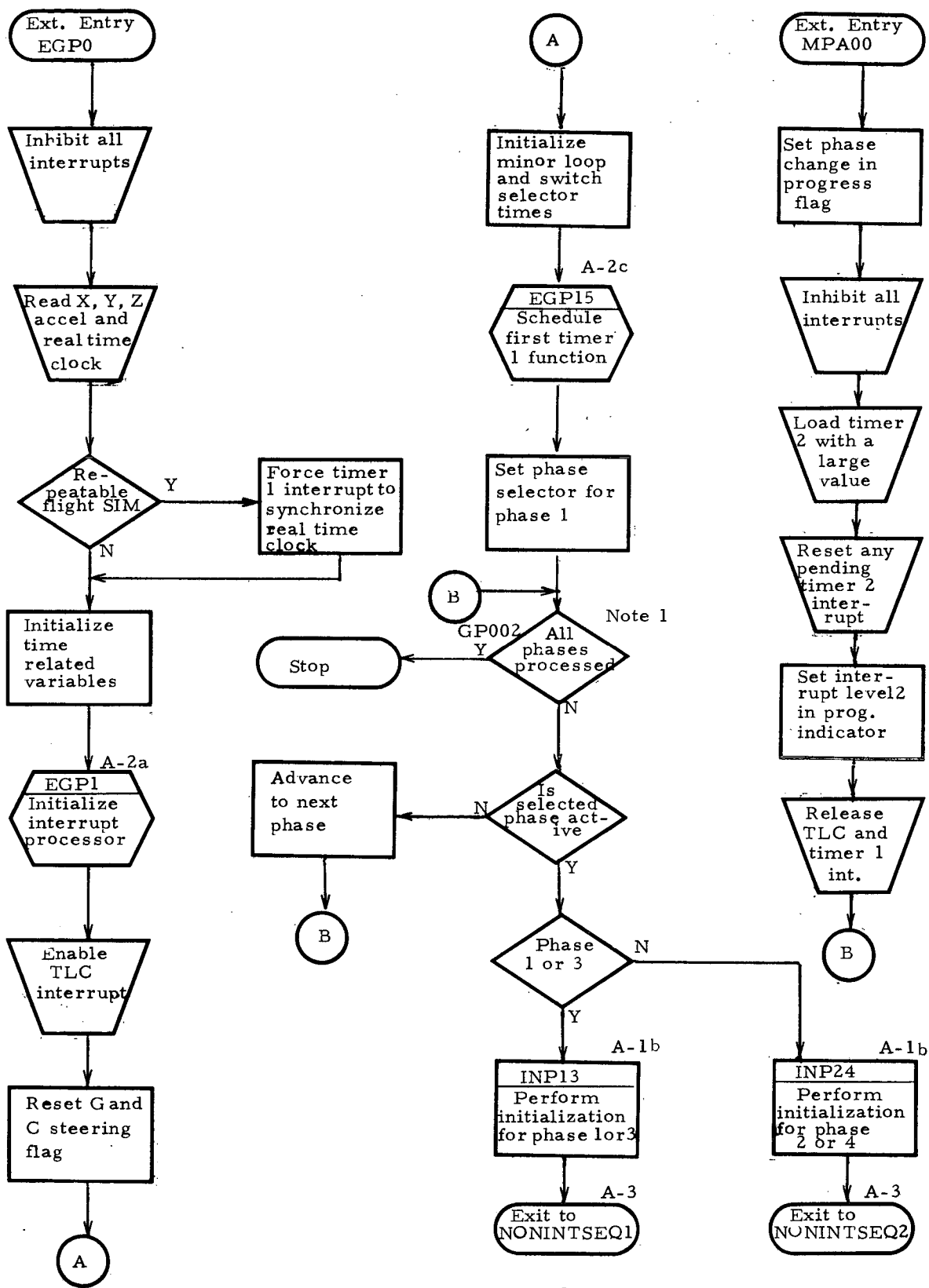


Figure A-1a

INITIALIZATION
(continued)

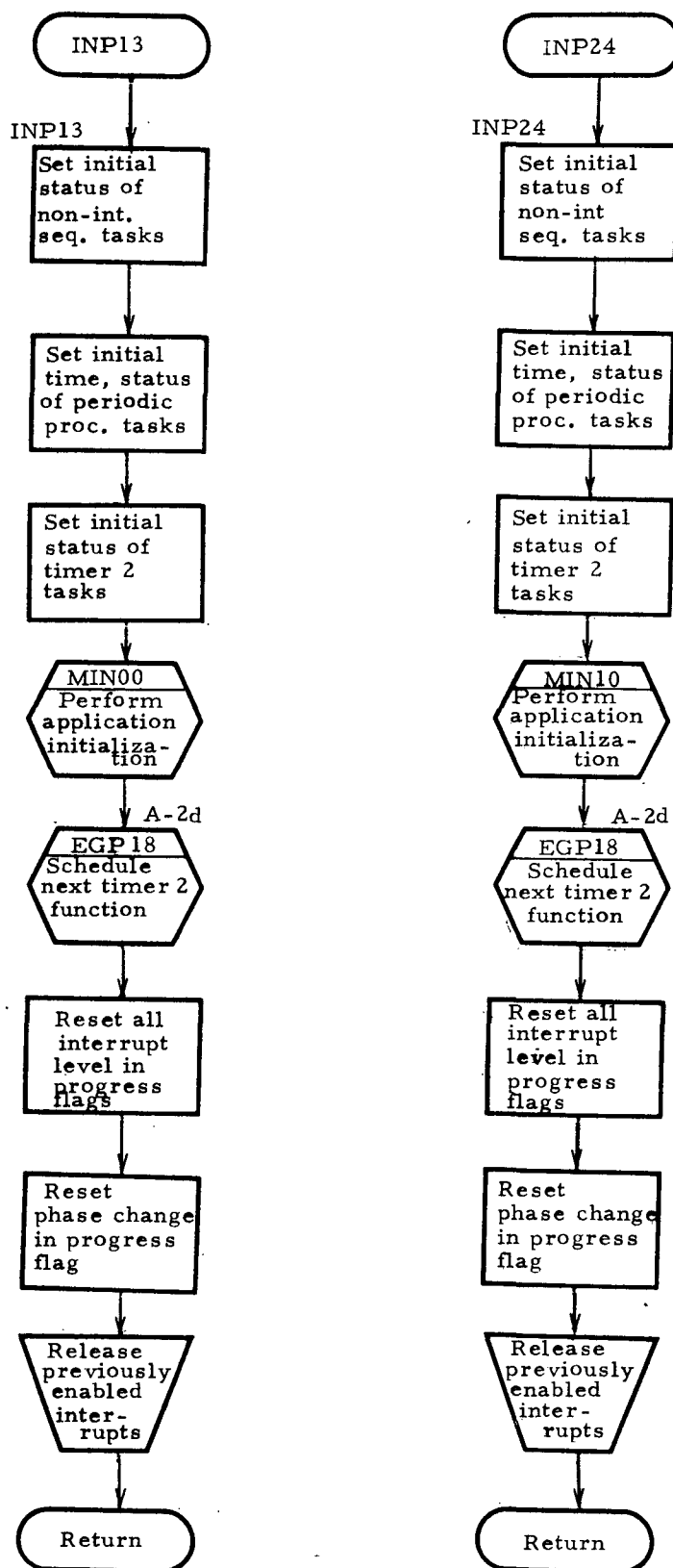


Figure A-1b

A.2 Interrupt Processor

A.2.1 Description of Operation

In most present-day computing systems and, in all likelihood, those of the future, hardware interrupts are used to signal both the occurrence of external events and/or the expiration of a program-specified time period. Direct handling of interrupts is performed by a task called the Interrupt Processor which is usually a part of an operating system. The Interrupt Processor determines the cause of the interrupt and makes provision for initiating the task associated with the interrupt. In a system where tasks are invoked according to priority, the task to be executed in response to the interrupt may or may not be executed before control is returned to the interrupted task, depending on relative priority of the two tasks. In non-priority systems, the interrupt task is executed before control is returned to the interrupted task.

The Saturn Flight Program has provision for five effective levels of priority. Listed in order of priority they are:

- Level 4 - Computer memory failure (TLC)
- Level 3 - Timer 1
- Level 2 - External interrupts
- Level 1 - Timer 2
- Level 0 - Background (non-interrupt level)

The two timers are program loadable and are used internally for scheduling of time dependent tasks.

Included in the Interrupt Processor kernel are the Timer Scheduler subroutines to illustrate capabilities for scheduling program controllable interrupts. Timer 1 Scheduler is dedicated to the Minor Loop and Switch Selector Tasks and schedules whichever is due next by loading the time-to-go into Timer 1. The Timer 2 Scheduler is assigned all remaining tasks which must be activated via a time-dependent interrupt. All Timer 2 tasks can be enabled or disabled under program control, and, when enabled, must have an activation time specified. The Timer 2 Scheduler selects the enabled tasks next due for execution and loads Timer 2 with the required time-to-go. Since Timer 2 can hold only a maximum of four seconds, the Timer 2 Scheduler schedules itself if no other task is due within the next four seconds.

Also included in the kernel is the system time update subroutine which maintains mission elapsed time by accumulating readings from a hardware real time clock.

A.2.2 Unique Language Characteristics Required

The Interrupt Processor requires facilities for responding to hardware interrupts and for controlling (inhibiting/enabling) them. Part of this control includes knowing which interrupts have been inhibited by other modules and, therefore, should not be enabled by this module. Since this capability was not readily available, comments were appended to the logic to indicate that only "previously enabled interrupts" are being enabled.

Interrupt control capabilities are often considered privileged functions which should be relegated to the operating system. In the Saturn Flight Program, however, application programs occasionally require direct interface with external hardware. For protection from other activities, they need control of interrupts, making it desirable to be able to perform such control in a high-level language. Interrupt control requirements are also demonstrated by several other kernels. Accelerometer Processing (Paragraph A.8.1) is a good example.

The Interrupt Processor also requires the ability to select the proper task (subprogram) for execution in response to a given interrupt since the task assignment varies in real time for the timer interrupts. Timing efficiency is highly important for selection and transfer of control.

A.2.3 Assumptions Made During Coding

It was assumed that certain functions were performed automatically by compiler-generated code or by the system under which the object programs execute. In particular, the saving and restoring of program status for the interrupted task as well as resetting the hardware interrupt indication were assumed to be automatic.

Symbolic names were assumed for each of the hardware interrupts of the Saturn Launch Vehicle Digital Computer. These names were then used in any kernel where direct reference was made to interrupts. Paragraph A.12 contains a glossary of these names.

A.2.4 Flowchart Notes

Note 1

The program entry point EGPI is utilized to activate the interrupt handling routines for SPL and CLASP. The statements within it are not executed during the activation process but are merely armed (readied) for execution in response to the associated interrupts. For HAL the entry is used to schedule the interrupt handling tasks.

Note 2

The Timer 1 interrupt handler for CLASP and HAL does not determine which of the Switch Selector modules is to receive control. Since these languages restrict a program module to a single entry point, control is passed to a common entry point of the switch Selector Processor which then internally decides which function is to be performed.

INTERRUPT PROCESSOR

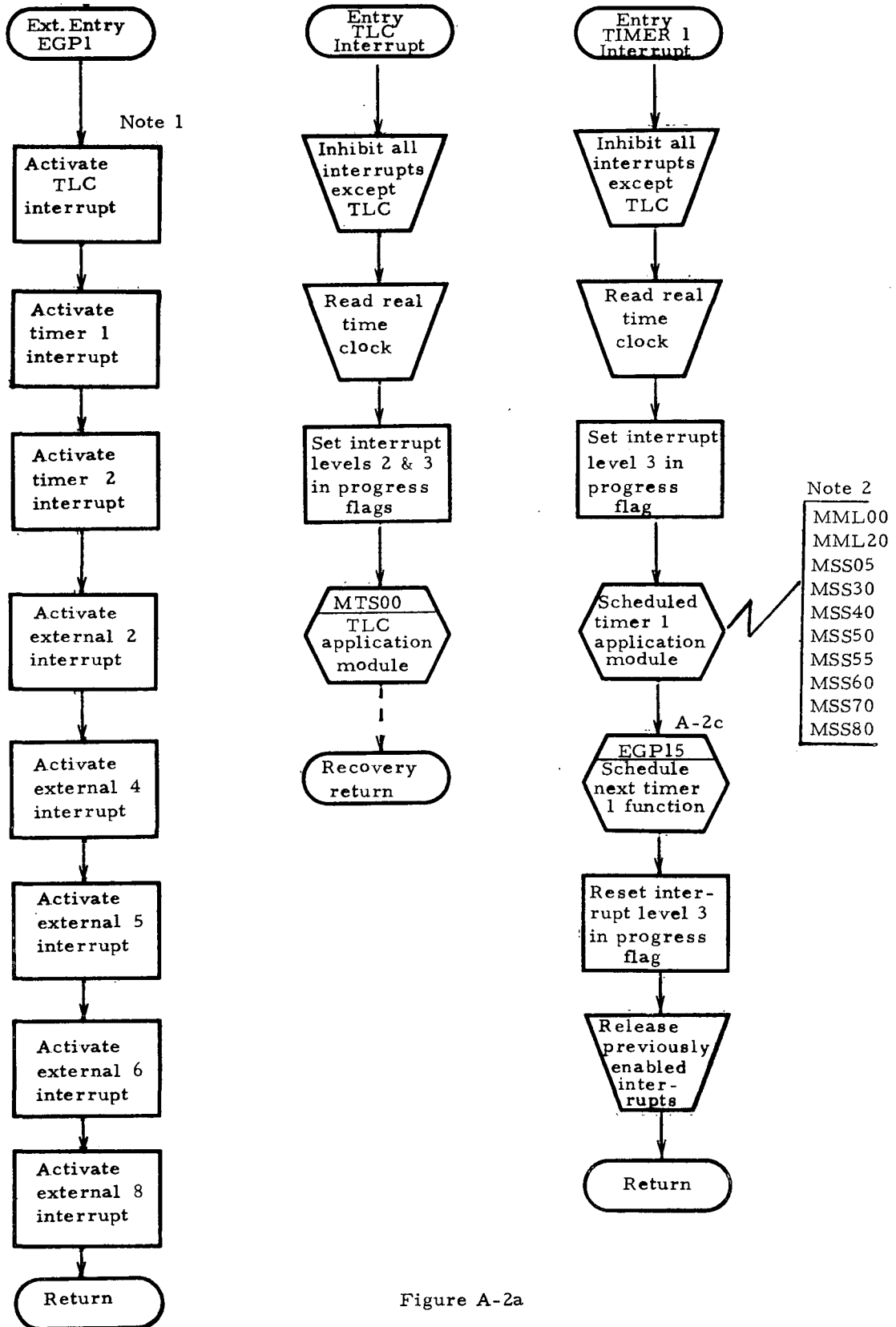


Figure A-2a

INTERRUPT PROCESSOR
(continued)

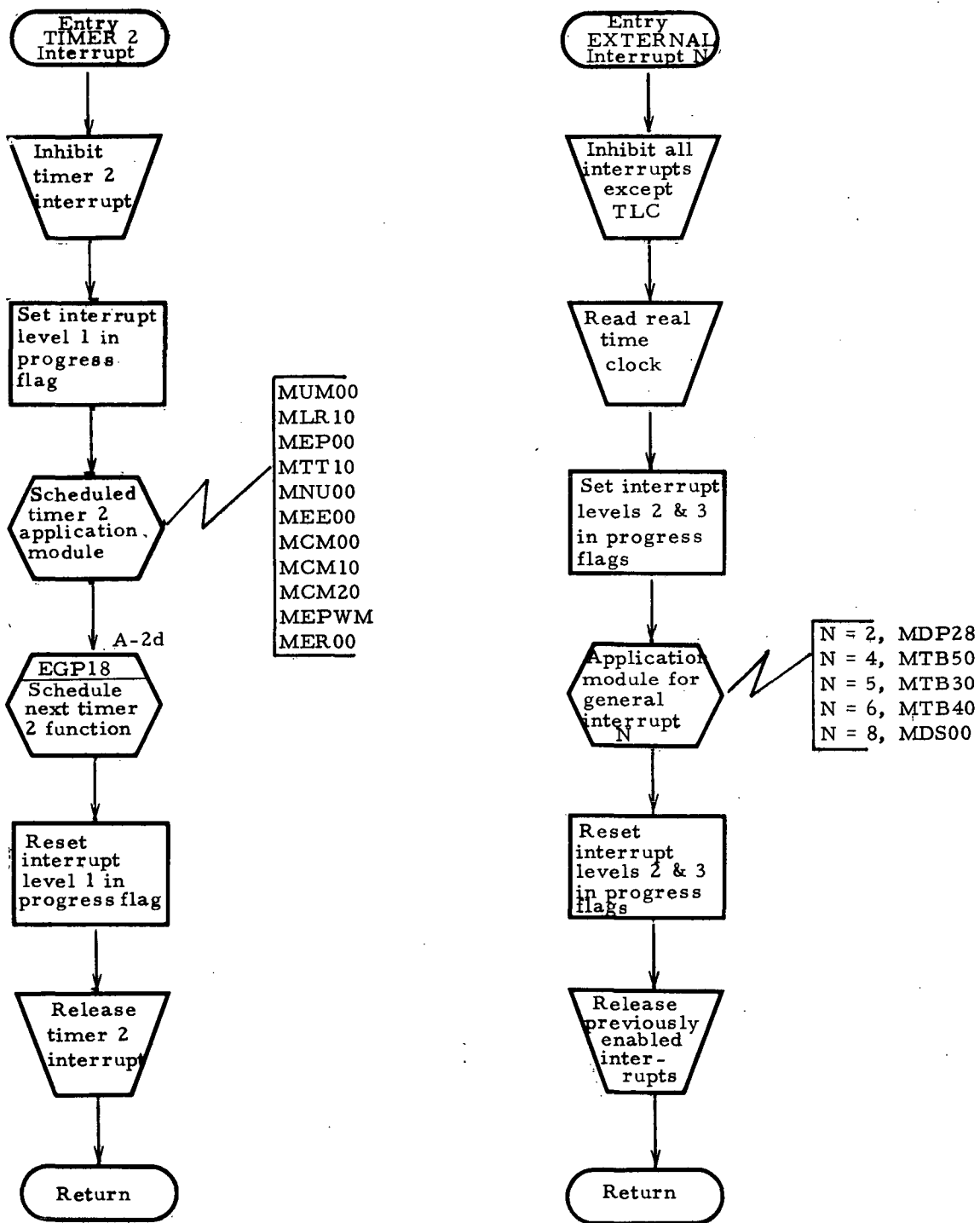


Figure A-2b

INTERRUPT PROCESSOR
(continued)

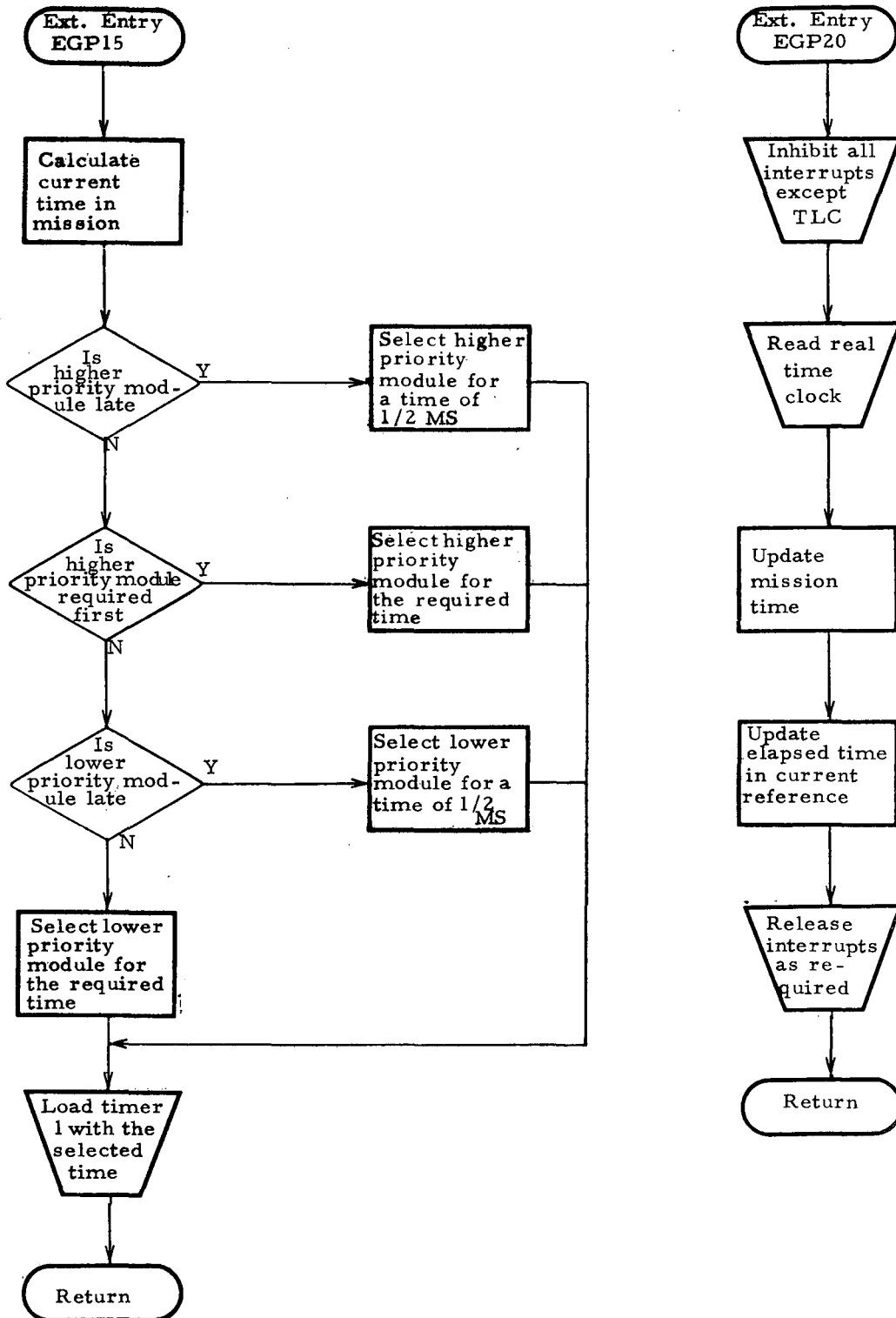


Figure A-2c

INTERRUPT PROCESSOR
(continued)

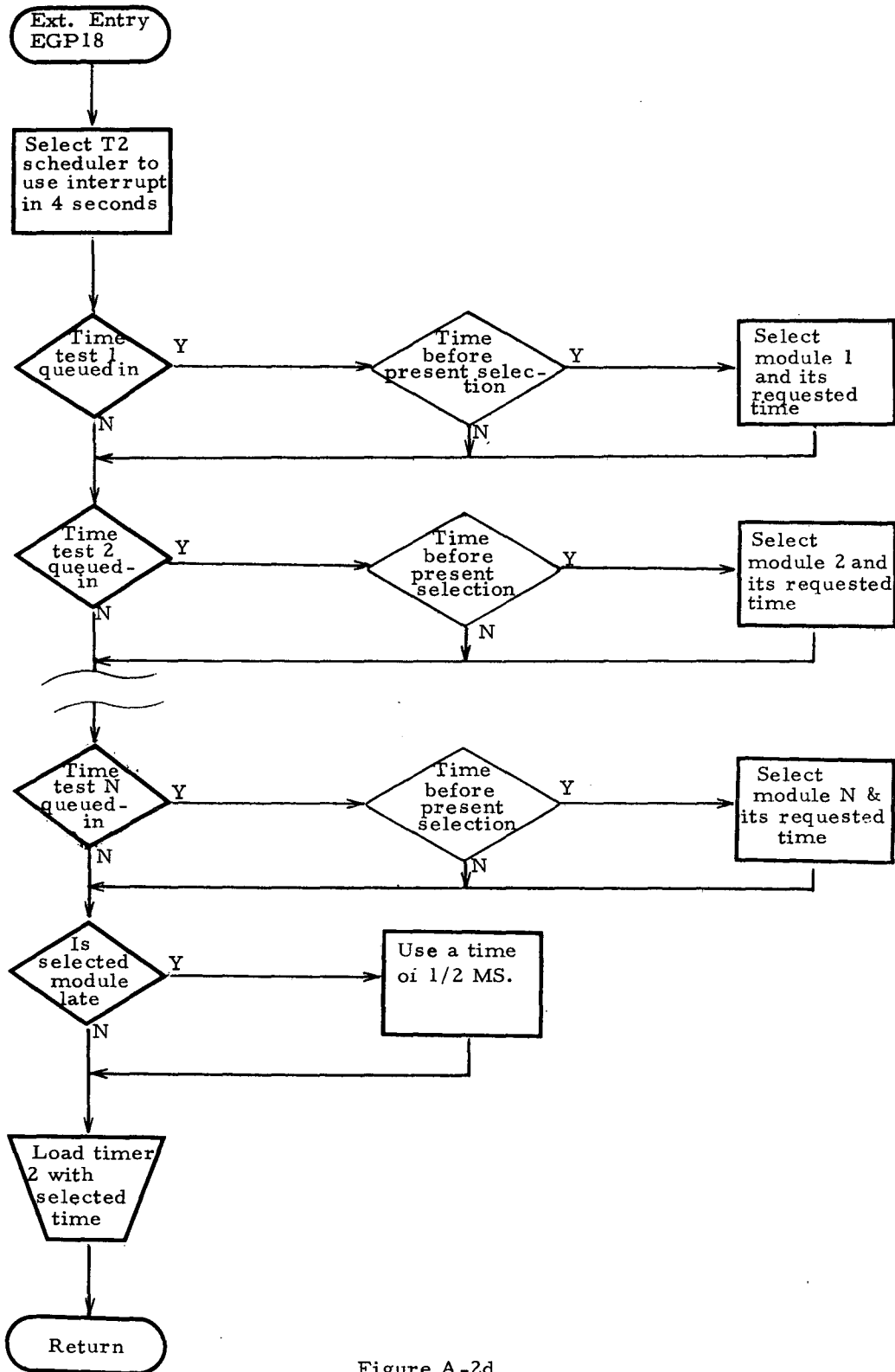


Figure A-2d

A. 3 Non-Interrupt Sequencer

A. 3.1 Description of Operation

The bulk of the Saturn Flight Program computations are performed on a non-interrupt basis. That is, the basic mode of execution consists of cycling a series of computational tasks on the lowest system priority level (lower than all of the interrupt levels). This is performed by the Non-Interrupt Sequencer which is a part of the operating system. Actually there are two Non-Interrupt Sequencers, one for the powered phases of a mission and one for the coast phases. Two sequencers are used because the computations performed differ considerably between the two phase types and require different groups of application tasks.

Tasks to be executed by the Non-Interrupt Sequencer have associated status indicators which can be used to enable or disable each individual task. During system initialization for a given mission phase, the status indicators for the tasks to be cycled during that phase are set to a predefined state. After initialization is completed, control is transferred to the appropriate sequencer.

The Non-Interrupt Sequencer for a given phase examines the status indicators assigned to it in the order in which the associated tasks are to be executed. If an indicator is enabled, the task is invoked. Otherwise the next indicator in the sequence is tested. When control is returned from an enabled application task, the sequencer calls the Periodic Processor (paragraph A.4) before stepping to the next indicator. After all indicators have been tested, the Non-Interrupt Sequencer returns to the first indicator in the group and repeats the cycle continuously until the end of the phase.

The status indicators are set as required by application tasks in response to the occurrence of external events (interrupts or discretes), on the basis of elapsed time, or as a result of internally programmed decisions. In this manner, the basic sequence of computations for a given mission phase can be modified as required.

A. 3.2 Unique Language Characteristics Required

The Non-Interrupt Sequencer existed in the Saturn Flight Program as executable tables consisting of modifiable instructions which were used to invoke enabled application tasks and to bypass disabled tasks. Rather than using status indicators to enable/disable, the

instructions in the sequencer control tables were simply modified as required.

Since programming in a higher level language makes it impractical, if not impossible, to "execute" a table or to modify instructions, the sample coding of the Non-Interrupt Sequencer was implemented through testing of status indicators as described in the preceding paragraph (A. 3. 1).

A. 3. 3 Flowchart Notes

Note 1

The Non-Interrupt Sequencer flowchart is general in the sense that it applies to any mission phase. Actually the kernel, as coded, contains two separate programs for the boost and coast mission phases.

NON-INTERRUPT SEQUENCER

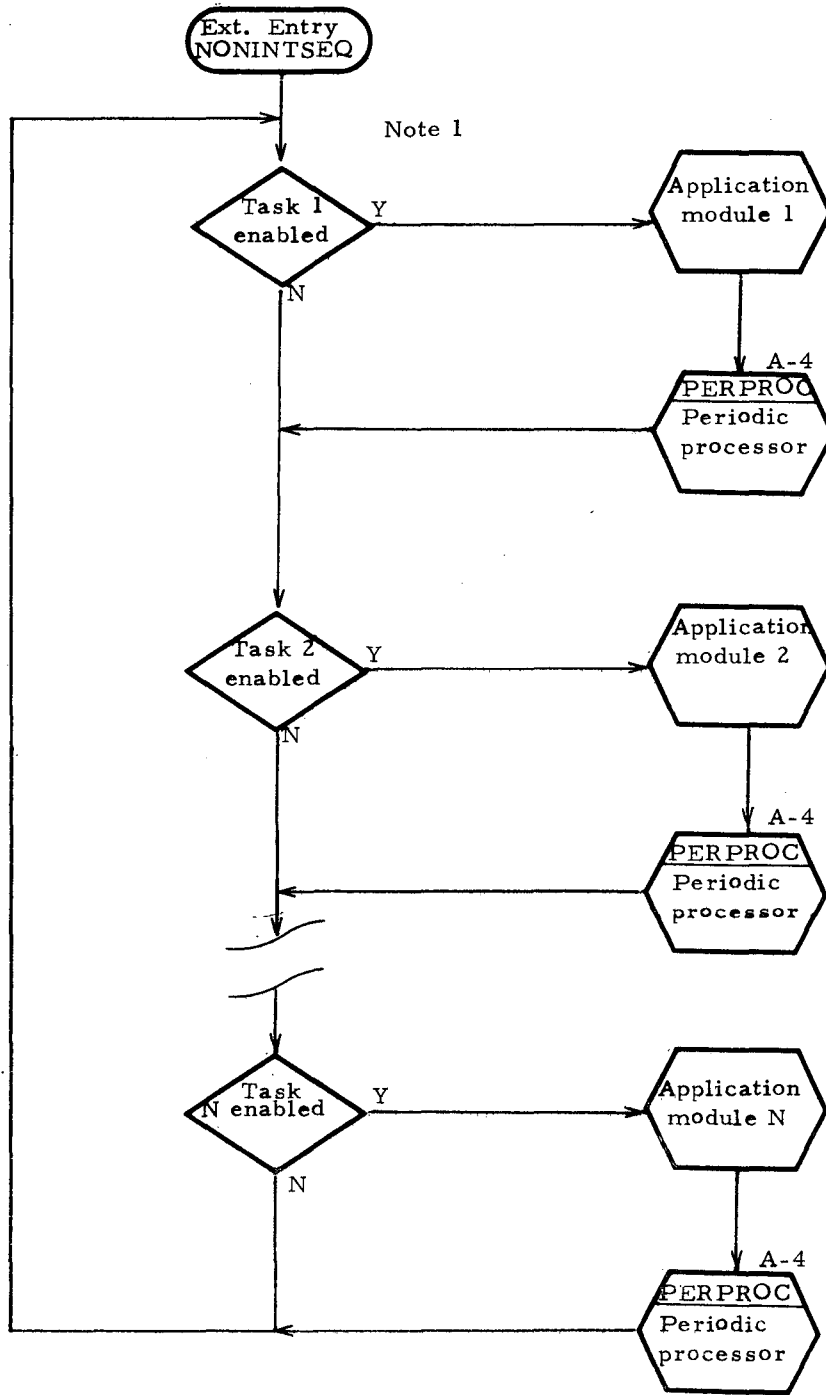


Figure A-3

A.4 Periodic Processor

A.4.1 Description of Operation

Certain tasks in the Saturn Flight Program must be executed repetitively at a fixed frequency but require neither stringent timing accuracy nor synchronization with other tasks. An example is a task which compresses data as a function of time. The scheduling of such tasks is performed by the Periodic Processor as a function of the operating system.

The Periodic Processor is invoked by the Non-Interrupt Sequencer following the execution of each enabled application task. Consequently, the timing accuracy with which it is capable of scheduling tasks is no better than the execution time required by the longest Non-Interrupt Sequencer subtask. Since this time resolution is relatively low, tasks with execution frequencies exceeding five times per second or with stringent timing accuracy requirements should be scheduled by the Interrupt Processor, through the Timer 1 and Timer 2 schedulers.

The Periodic Processor utilizes control tables containing timing information for each periodic application task and status indicators similar to those of the Non-Interrupt Sequencer (paragraph A.3). The Periodic Processor first examines the status indicator for an entry and then, if the task is enabled, it compares the task execution interval with the time elapsed since its last execution. If the task is enabled and is due to execute, it is invoked by the Periodic Processor. When the task completes execution and returns control, or when the task for a given entry is not invoked, the Periodic Processor continues on to the next table entry. Upon reaching the end of the table, control is returned to the Non-Interrupt Sequencer.

A.4.2 Unique Language Characteristics Required

The Periodic Processor requires the capability to access data from control tables.

PERIODIC PROCESSOR

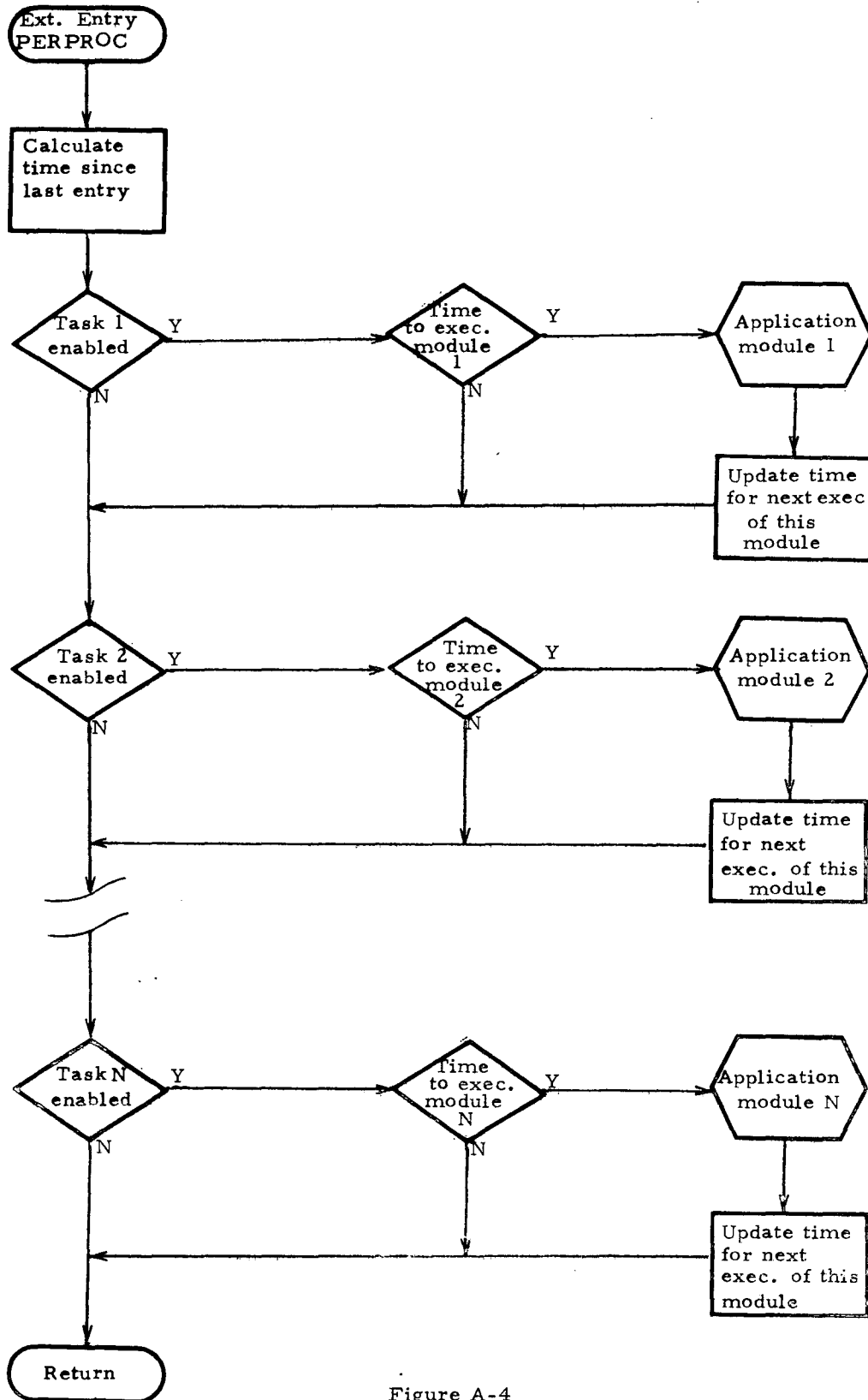


Figure A-4

A.5 Events Processor

A.5.1 Description of Operation

Non-repetitive tasks to be executed at a given time are scheduled for execution by the Events Processor in coordination with the Interrupt Processor. The Events Processor utilizes a predefined table of task identifiers with associated execution times. An example of such a task is one which sets accelerometer reasonableness test constants at a given time during the mission.

The Events Processor selects one entry at a time from the table and schedules the entries in the sequence in which they exist in the table. The execution time for a given table entry is used by the Events Processor to reschedule itself via the low priority timer of the Interrupt Processor. Then when it is reactivated at the specified time, it executes the associated task and selects the next entry from the table. When it reaches the end of a table, it disables itself and remains dormant until it is re-enabled at a later time.

Two special entry points are required in addition to the normal entry from the Interrupt Processor. The first is used at the start of each mission time base (time reference frame) to initialize pointers to the beginning of the corresponding Event Table. The second entry is utilized to enable and reschedule the Events Processor as required following periods when it has been disabled.

A.5.2 Unique Language Characteristics Required

The Events Processor is responsible for invoking a relatively large group of tasks (one at a time) using the identifiers obtained from the Events Processor Table. Language capabilities permitting a call to one of several tasks depending on the value of the identifier would significantly improve efficiency. Lacking such capabilities the programmer is forced to code a call for each task and then use the identifier as an index for a "computed GOTO" in order to pass control to the tasks.

As implied in the preceding discussions, the Events Processor also requires means for accessing data tables.

A. 5. 3 Flowchart Notes

Note 1

Since CLASP and HAL do not permit multiple entry points for a module, the MEP05 module must call the MEP10 module for these languages rather than transfer control to it as shown in the flowchart.

EVENTS PROCESSOR

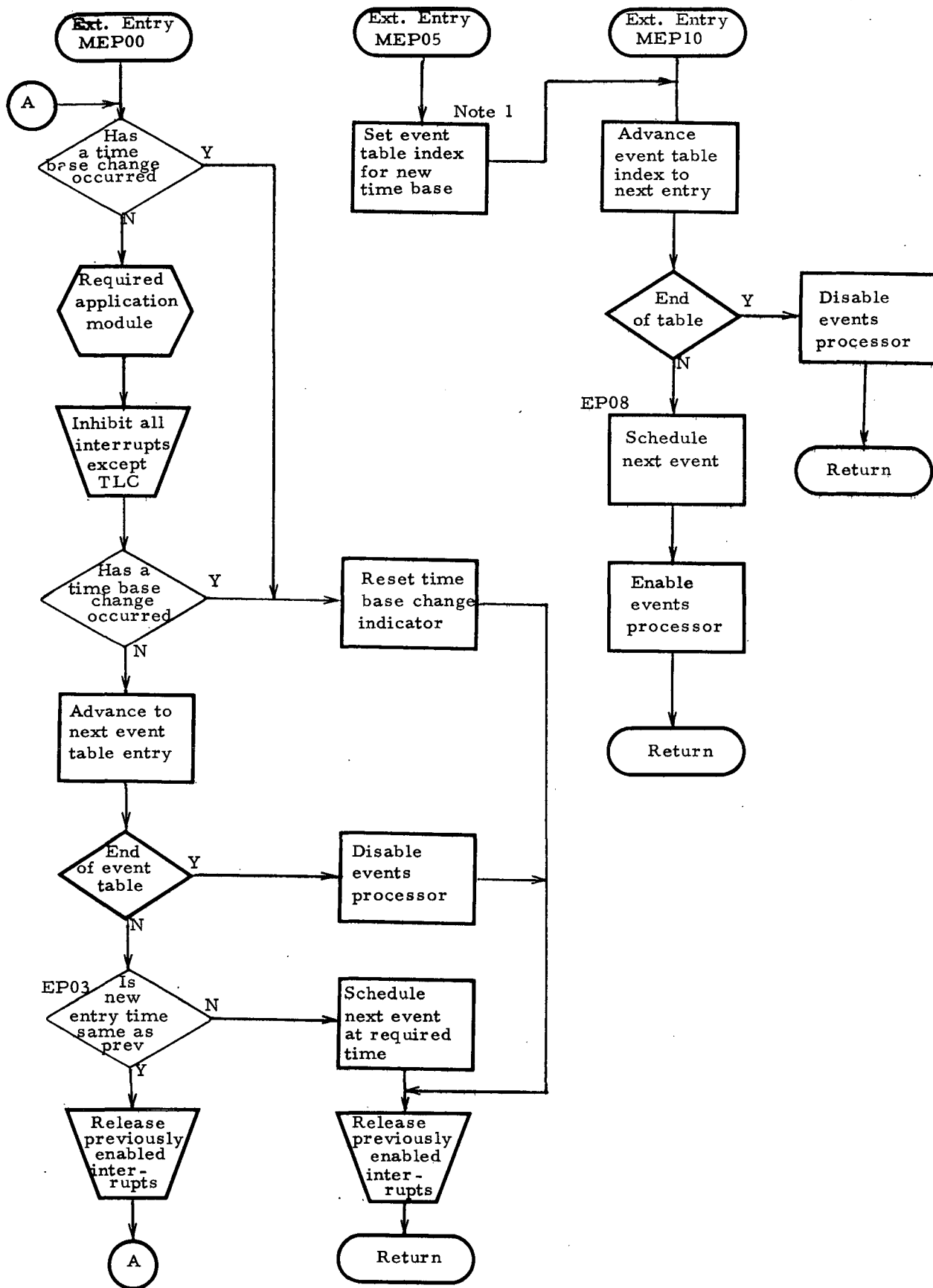


Figure A-5

A.6 Iterative Guidance Mode

A.6.1 Description of Operation

Iterative Guidance Mode (IGM) is a path-adaptive guidance program which steers along a nearly optimum trajectory toward a predefined target. It is path-adaptive in the sense that it is designed to adjust to perturbations to nominal vehicle performance. For example, if one of the upper stage engines fails to develop full thrust, IGM will adapt the steering computations to still achieve terminal position and velocity with sufficient accuracy. The steering program is based on the calculus of variations and is derived from a simplified set of differential equations of motion. It is designed for powered flight in a vacuum with multiple distinct thrust levels and short coasting periods.

IGM is executed once each iteration of the flight program background loop (Non-Interrupt Sequencer, paragraph A.3) during the periods when it is active. It performs two basic functions:

- o Guidance computations
- o Phasing

Guidance computations generate vehicle steering commands (desired attitude angles) using navigation data, vehicle performance data, time, and desired terminal conditions. Calculations are performed in the target plane and injection coordinate systems and then rotated into the plumblane coordinate system for attitude control.

Phasing evaluates vehicle performance data and estimates the times to go until the expected thrust level changes occur. For the Saturn V vehicle and missions, there are two distinct thrust level changes for the translunar injection boost period.

Due to the large size of IGM, it is neither informative nor practical to code all of it in each of the languages. Therefore, only the portion containing the guidance computations is coded. The operations performed by the phasing segment are similar to those contained in other kernels so coding them would be redundant.

A.6.2 Unique Language Characteristics Required

The IGM kernel contains the majority of the numerical computations performed by the selected kernels. In addition to the common

mathematical expressions including built-in functions (LOG, SQRT, SIN, ATAN, etc.), it also demonstrates vector and matrix operations. It requires capabilities for coding vector expressions and for performing such functions as dot product and vector rotation.

A.6.3 Flowchart Notes

Note 1

The dashed connector from the entry point to the first block indicates the omission of the phasing portion of IGM which was not coded.

ITERATIVE GUIDANCE MODE

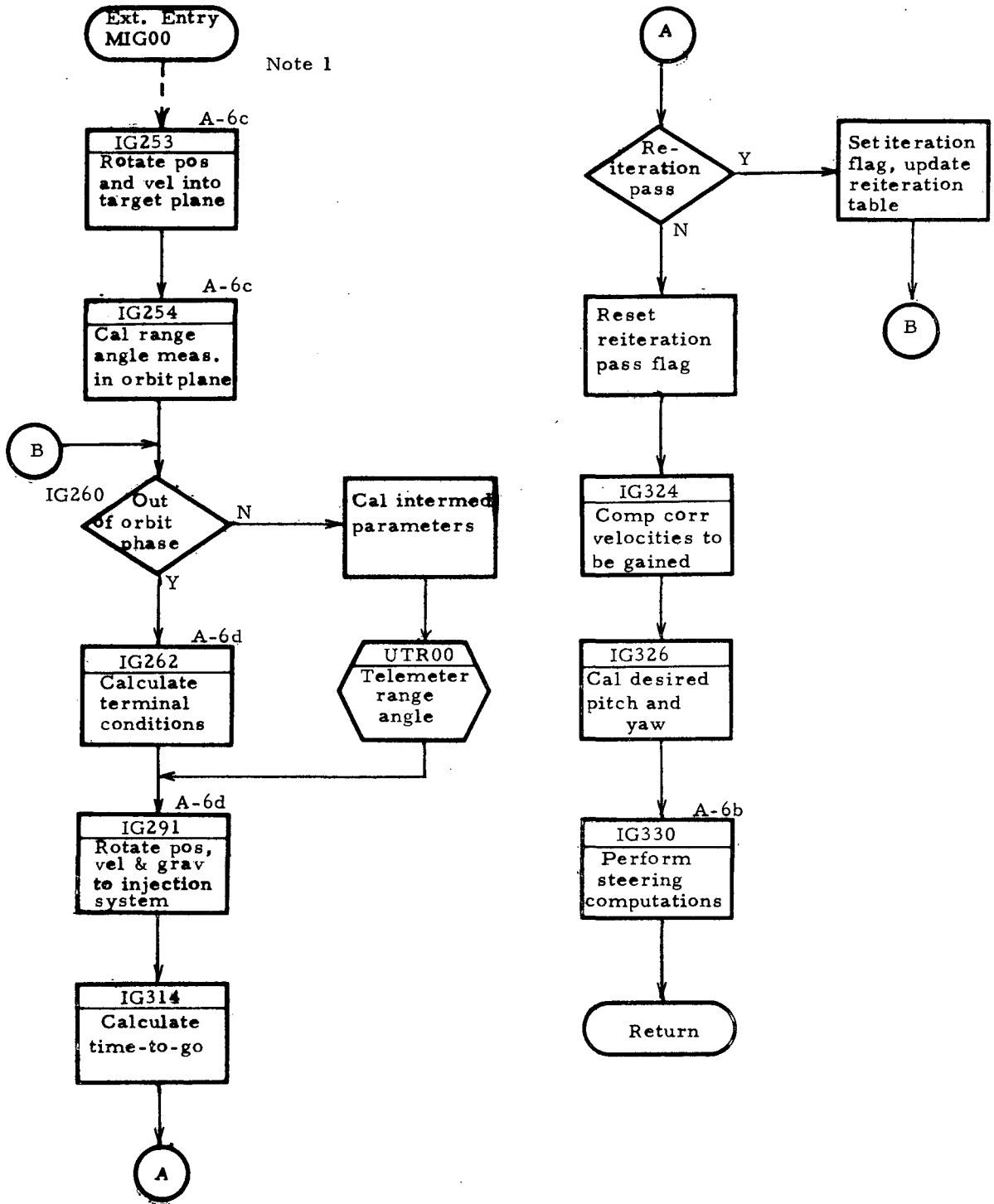


Figure A-6a

ITERATIVE GUIDANCE MODE
(continued)

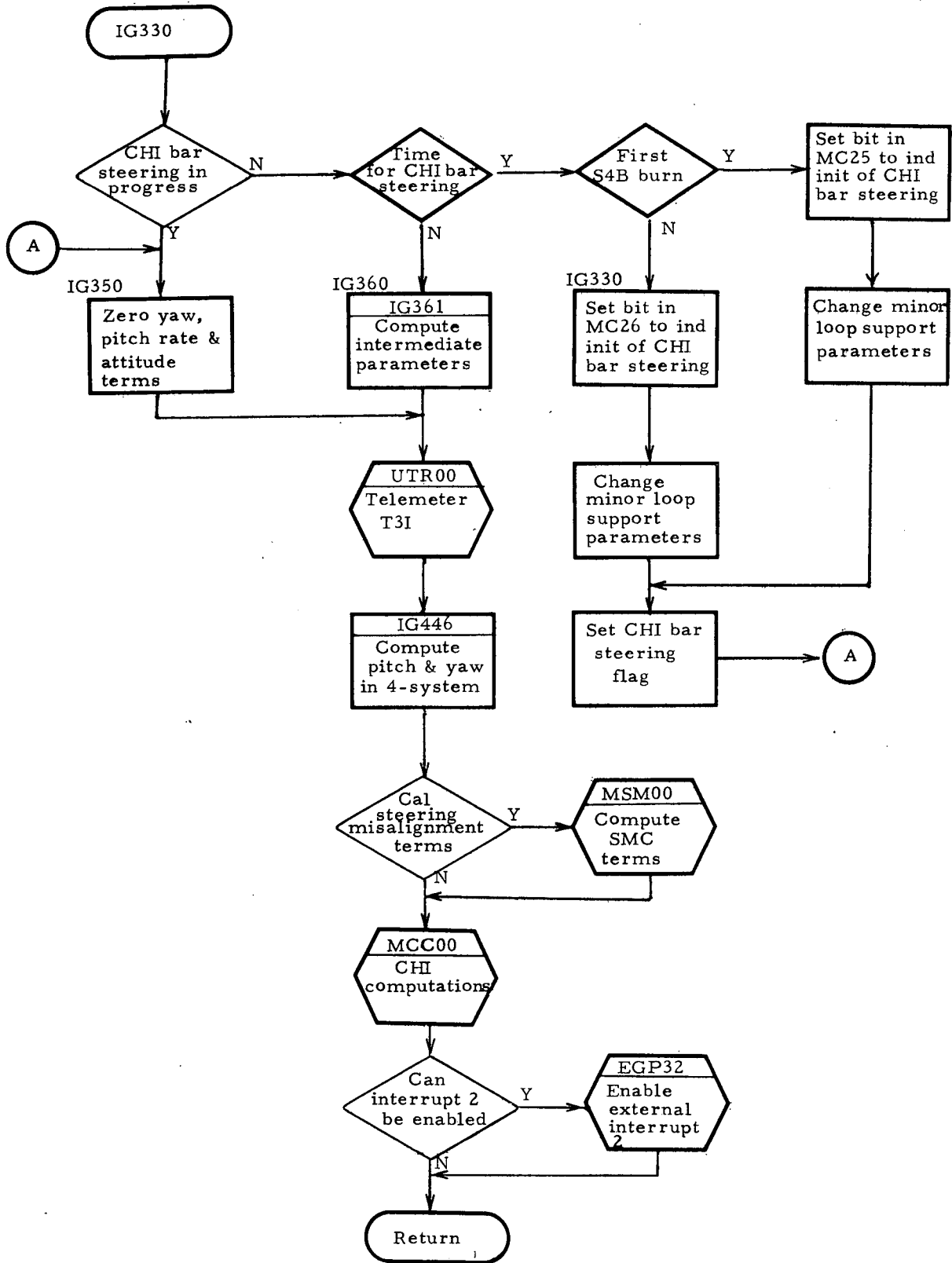


Figure A-6b

ITERATIVE GUIDANCE MODE
(continued)

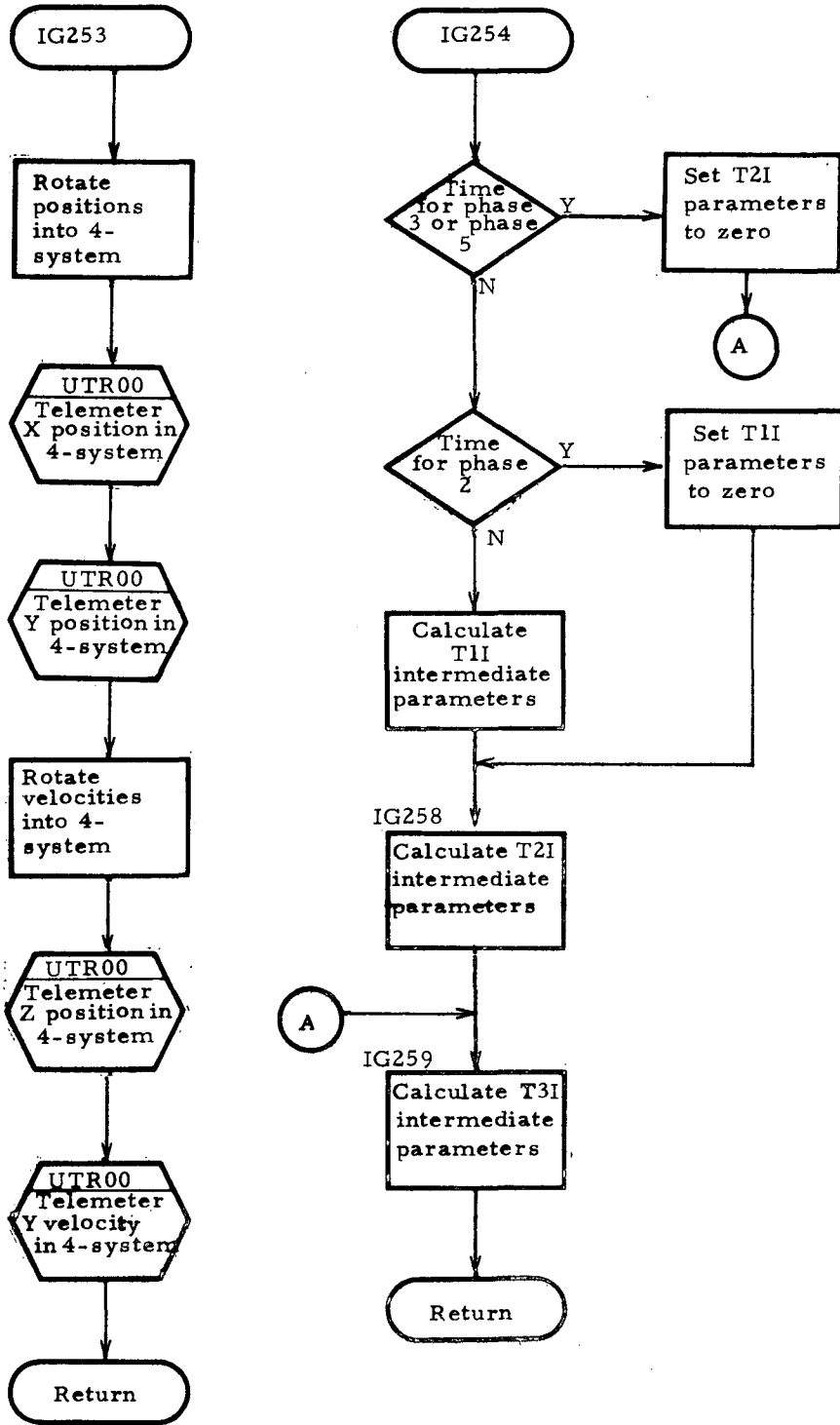


Figure A-6c

ITERATIVE GUIDANCE MODE
(continued)

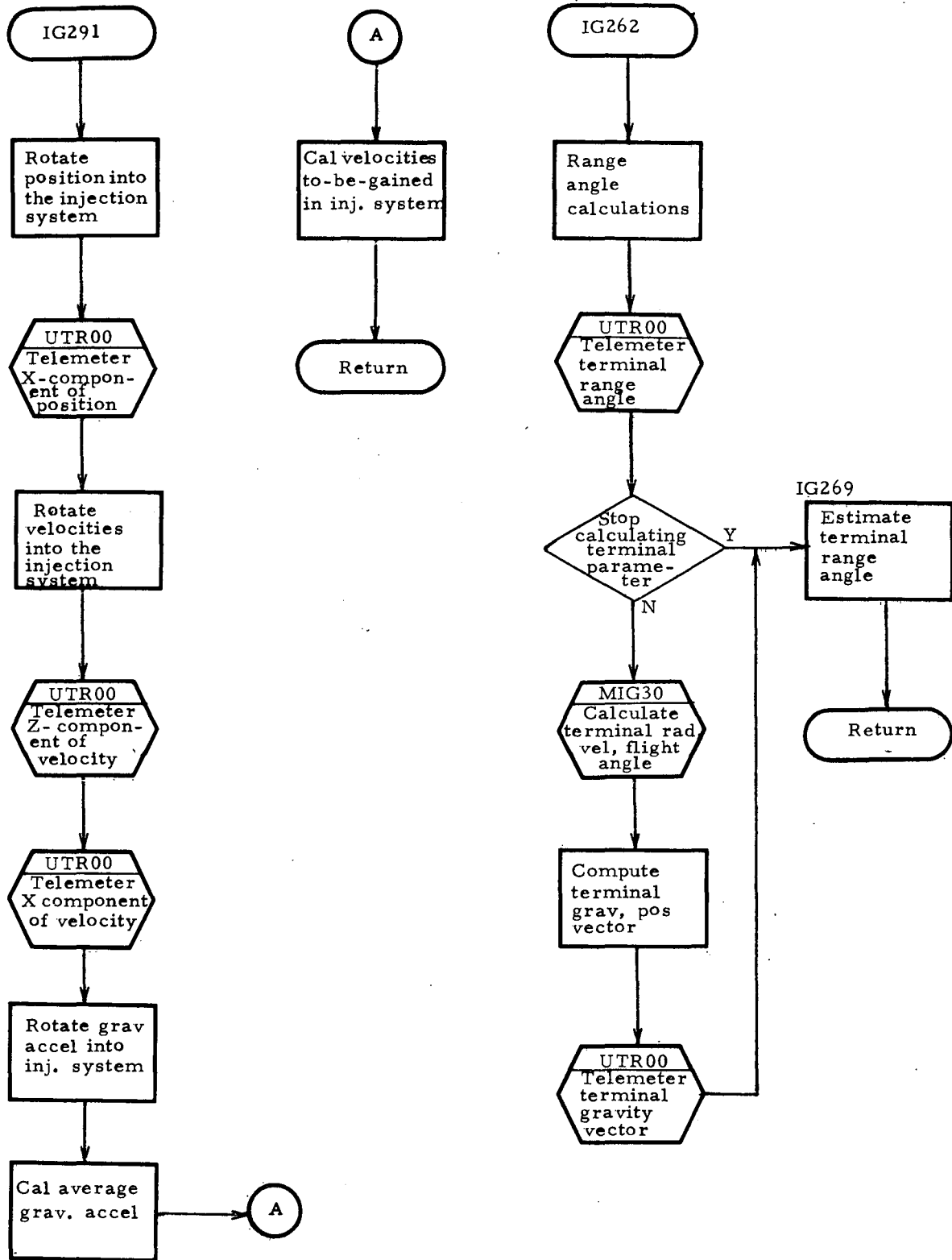


Figure A-6d

A.7 Digital Command System (DCS)

A.7.1 Description of Operation

The Digital Command System provides communication facilities for receiving commands and data transmitted from ground stations. Capabilities exist for controlling flight program timing, navigation, guidance, targeting, and sequencing functions from the ground and for requesting specific program data to be telemetered to the ground.

Each DCS function, as received by the DCS software task, consists of a mode command to identify the function, followed by a variable number of data commands depending on the requirements of each function. The DCS task is initiated by the Interrupt Processor in response to the hardware indication that input data has been received. When a mode command is received it is tested for validity and legality and then analyzed to determine whether or not data words are required to perform the associated function. If data is required, the DCS task returns control to the operating system and is reinitiated as each data command is received. Each data word is also tested for validity and legality as it is received. When all data for a given function has been received, or if a function does not require data, the appropriate module is activated to process the function. Upon the detection of errors in the DCS inputs, error messages are formulated and transmitted back to the ground and the function is not activated.

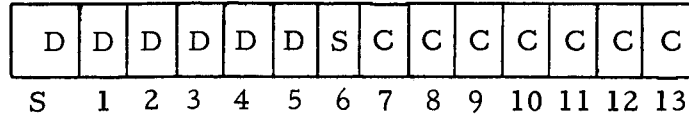
The coded kernel does not include the various application modules which are invoked to perform the requested functions. Only the central, coordinating portion of the overall DCS is demonstrated.

The format of DCS input data is shown in Table A-1 along with a list of functions in Table A-2 and error codes in Table A-3.

A.7.2 Unique Language Characteristics Required

The Digital Command System has requirements to perform real time I/O. It reads the DCS Input Register to obtain the incoming data and the Discrete Input Register to examine the bit which stipulates whether the DCS input data is a mode command or a data word for a previous mode command. It also writes to the Discrete Output Register to issue the command reset pulse for the Command Receiver.

DCS INPUT FORMAT



LVDC
Bit Position

Significance

S - 5
 6
 7 - 13
 14 - 25

DCS mode or data command
 Sequence bit
 Complement of bits S-6
 Unused

Table A-1

DCS FUNCTIONS

| Mode Comm. | Status Code | No. Data Words | Function |
|---------------|----------------|-------------------|--|
| 05 | 05 | 0 | Maneuver inhibit |
| 10 | 10 | 1 | Time base update |
| 11 | 11 | 35 | Navigation update |
| 12 | 12 | 2 | Generalized switch selector |
| 13 | 13 | 2 | Sector dump |
| 14 | 14 | 3 | Telemeter single memory location |
| 17 | 17 | 0 | Time base 8 enable |
| 20 | 20 | 0 | Terminate |
| 22 | 22 | 1 | Maneuver update |
| 25 | 25 | 0 | Execute alternate sequence 6D |
| 31 | 31 | 35 | Target update |
| 33 | 33 | 0 | Execute communication maneuver |
| 34 | 34 | 0 | Execute evasive maneuver |
| 45 | 45 | 0 | Inhibit coolant control valve |
| 52 | 52 | 6 | S-IVB/IU lunar impact |
| 53 | 77 | 0 | Switch CCS antenna system to omni |
| 54 | 77 | 0 | Switch CCS antenna system to low gain |
| 55 | 77 | 0 | Switch CCS antenna system to high gain |
| 60 | 60 | 0 | Transposition, docking, and extrac- tion enable |

Note: The Mode Command comes from bits S-5 of the input command. The Status Code is the telemetered status word. Both are represented in octal.

Table A-2

DCS ERROR CODES

| Error Code (Octal) | Description |
|-----------------------|---|
| 04 | Orbital Mode/Data bit is invalid; data command was received when a mode command was expected. |
| 10 | True complement test failed for mode command; information bits 7-13 are not the complement of bits S-6. |
| 14 | Mode command invalid; the mode command received is not defined for this mission. |
| 20 | Orbital Mode/Data bit is invalid; mode command was received when expecting a data command. |
| 24 | Mode command sequence bit incorrect; the sequence bit received was 1 instead of 0. |
| 34 | Unable to issue generalized switch selector command function at this time; the last requested generalized switch selector command function has not been issued. |
| 44 | True complement test failed for data command; information bits 7-13 are not the complement of bits S-6. |
| 54 | The time of implementation of a navigation update or target update command is less than 10 seconds in the future. |
| 60 | Data command sequence bit incorrect; the sequence bit must begin with 1 and alternate from 1 to 0 in each sequential data command of a set. |
| 64 | A DCS program is in progress at this time; however, no more data is required; only a terminate mode command can be processed at this time. |
| 74 | The mode command received is defined for this mission but is not acceptable in the present time frame. |

Table A-3

The kernel also requires capabilities for unpacking the input data and performing validity and legality tests on the data. When an error is detected, the data must be formatted for an error message. Table accessing facilities are also required since information concerning each mode command is stored in tables. The information includes:

- o Number of data words required
- o Command activity status (enabled/disabled)
- o Status code (for telemetry)

Since a variety of functions must be invoked by DCS, a variable call facility as discussed in paragraph A.5.2 would be useful.

A.7.3 Assumptions Made During Coding

The DCS kernel was not coded as it exists in the Saturn Flight Program. It was reorganized to simplify program logic while retaining all of the necessary functions. Reorganization primarily involved the centralization of certain functions within the DCS kernel which, in the original flight program, were performed in the various DCS application sub-task modules. In particular, each application module previously was required to determine whether or not it was active, to issue status telemetry, and to make provision for obtaining any needed input data. In the coding of the DCS kernel these functions were performed in the DCS task itself to eliminate duplication and greatly simplify the operation of the application modules.

DIGITAL COMMAND SYSTEM

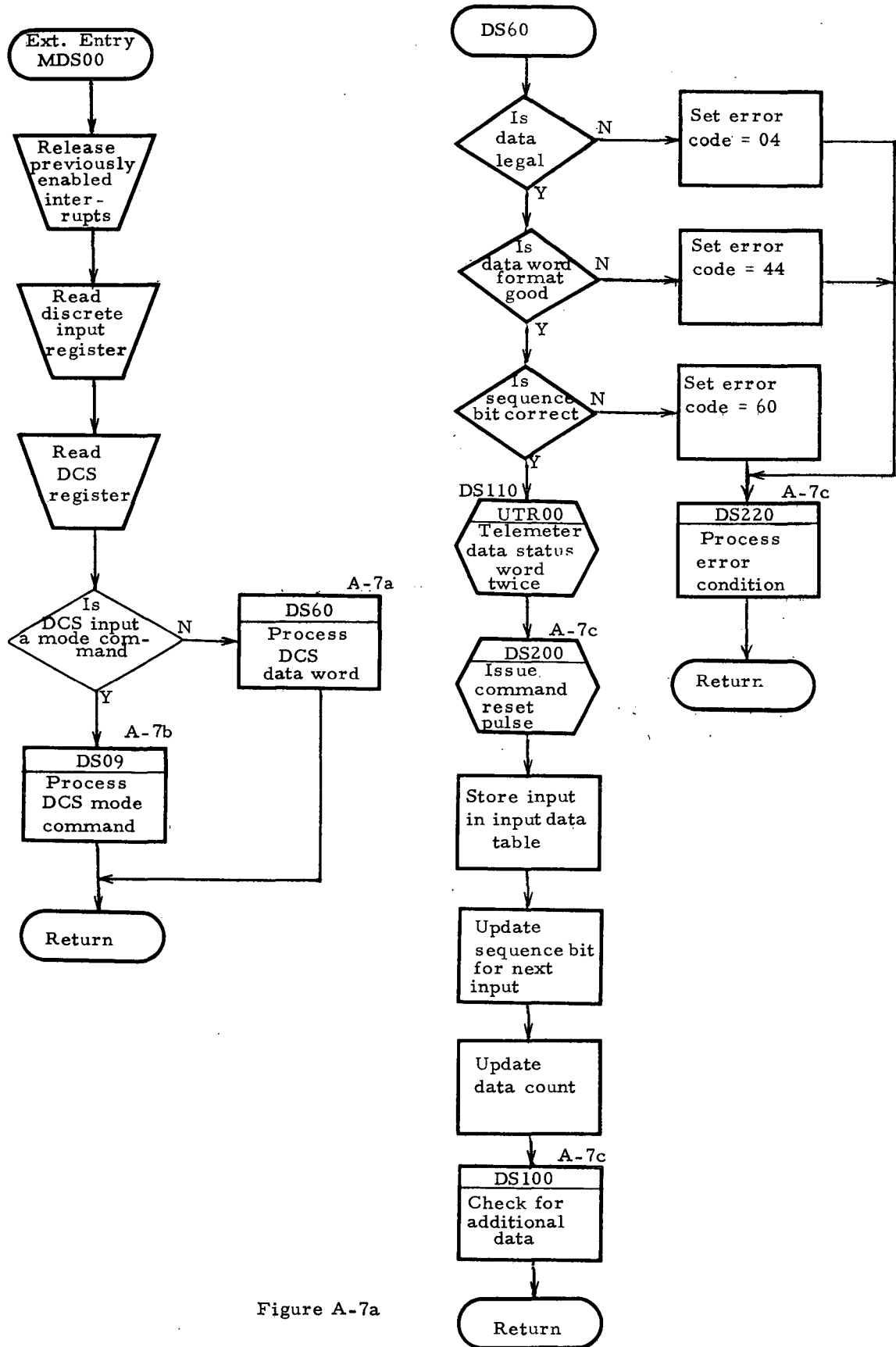


Figure A-7a

DIGITAL COMMAND SYSTEM
(continued)

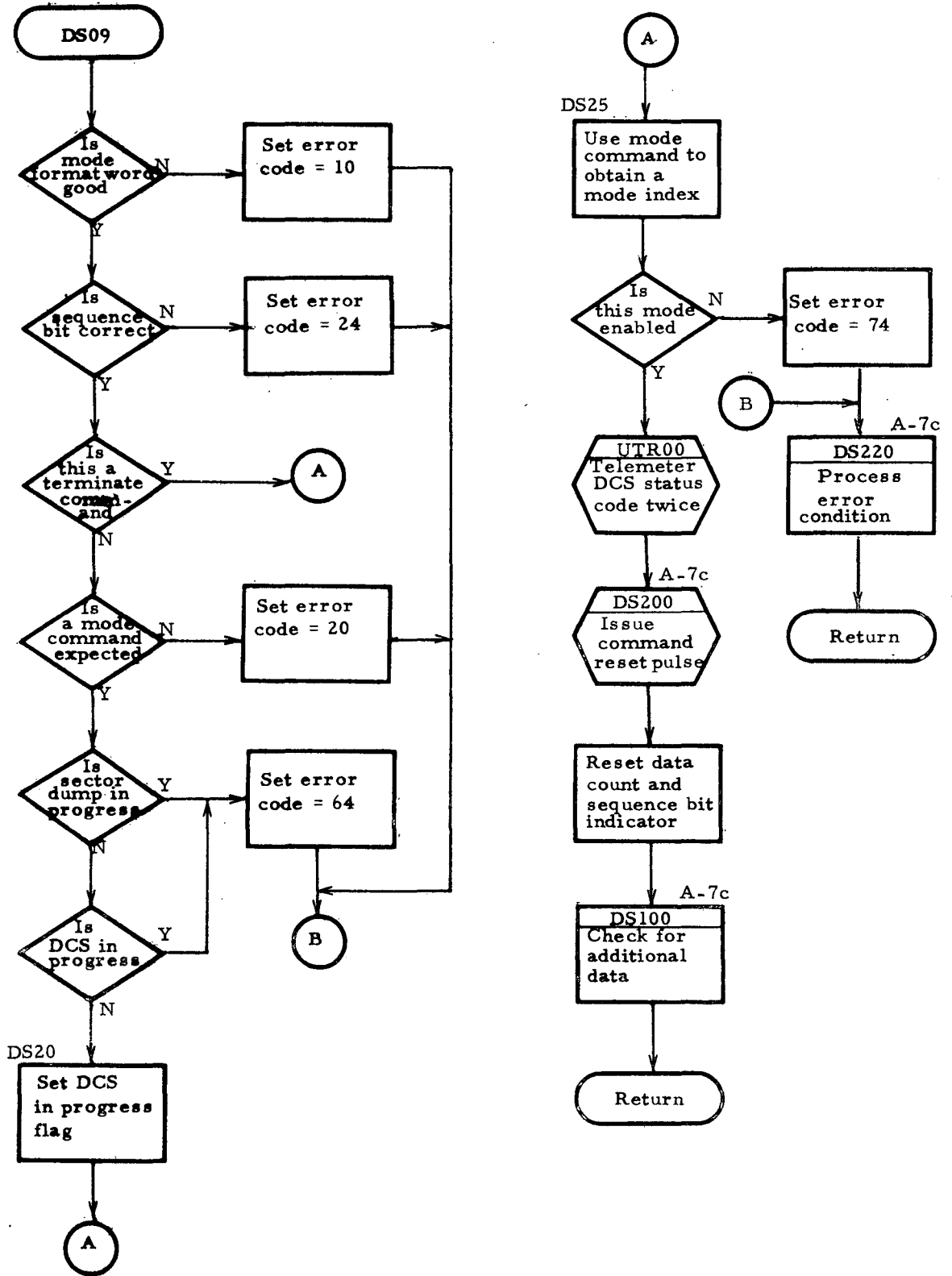


Figure A-7b

DIGITAL COMMAND SYSTEM
(continued)

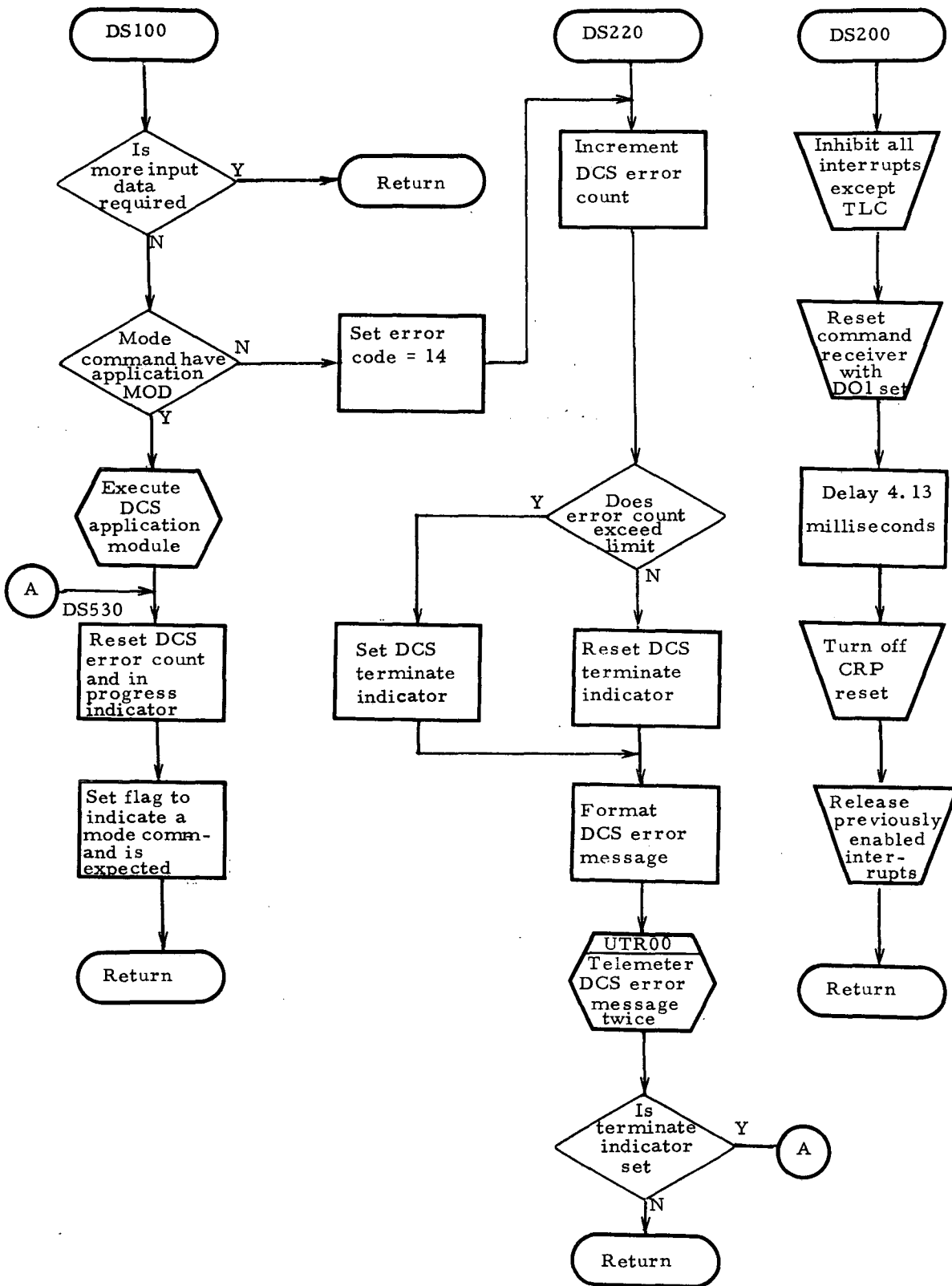


Figure A-7c

A.8 Accelerometer Processing

A.8.1 Descriptions of Operation

The accelerometers attached to the inertial platform of the vehicle provide data which serve as the basis for performing navigation during boost phases of a mission. Accelerometer Processing, as its name implies, reads the accelerometers and refines the data into a form suitable for updating vehicle position and velocity.

During periods when it is enabled, Accelerometer Processing is executed once each iteration of the flight program background loop by the Non-Interrupt Sequencer (paragraph A.3). It first inhibits interrupts, reads the accelerometers for all three platform axes, reads the real time clock, and then releases the interrupts. Interrupts are inhibited to insure that the input data are all obtained at a given point in time rather than separated in time by the execution of an interrupt-driven task.

Before the input data can be used for navigation, each accelerometer reading is subjected to three tests. Each reading provides two pulse counts for redundancy. These pulse counts are subtracted from the pulse counts of the previous computation cycle to obtain two delta readings which represent the change in vehicle velocity along that axis during the previous computation cycle. The two delta readings are then compared and if they disagree by more than two pulses, an error indication is set. The delta closest to a predicted value is selected for further processing.

A zero test is performed next to detect an unchanging accelerometer. Finally, a reasonableness test is performed in which the actual delta is required to fall within a band of plus or minus fifty percent of the predicted value enlarged by a reasonableness constant. If a reading does not pass the reasonableness test, it is replaced by a backup value derived from an internally calculated acceleration profile. Error indications are set to indicate failure to pass any of the tests.

After the tests are performed, the readings are used to calculate vehicle acceleration and to update vehicle velocity.

An additional function performed by the Accelerometer Processing kernel is the calculation of mission time at the time the accelerometers are read.

A. 8.2 Unique Language Characteristics Required

Accelerometer Processing requires facilities for reading real time data (acceleration and time) and for converting the data to an internally usable form. The need also exists for controlling interrupts via a momentary inhibit as discussed in paragraph A. 8. 1.

A. 8.3 Flowchart Notes

Note 1

The computation of the average CHIs for the SMC calculations (see A-8b) is shown in the flowcharts as coded for SPL and CLASP, where PIRADS were used. In HAL, PIRADS were not used so the special test shown for the averaging of the pitch commanded CHI was unnecessary.

Note 2

Likewise, for the computation of the expected velocity changes (see A-8b), usage of the special SIN/COS routine (USC00) for PIRADS was replaced by the usage of the built-in SIN/COS functions in HAL.

ACCELEROMETER PROCESSING

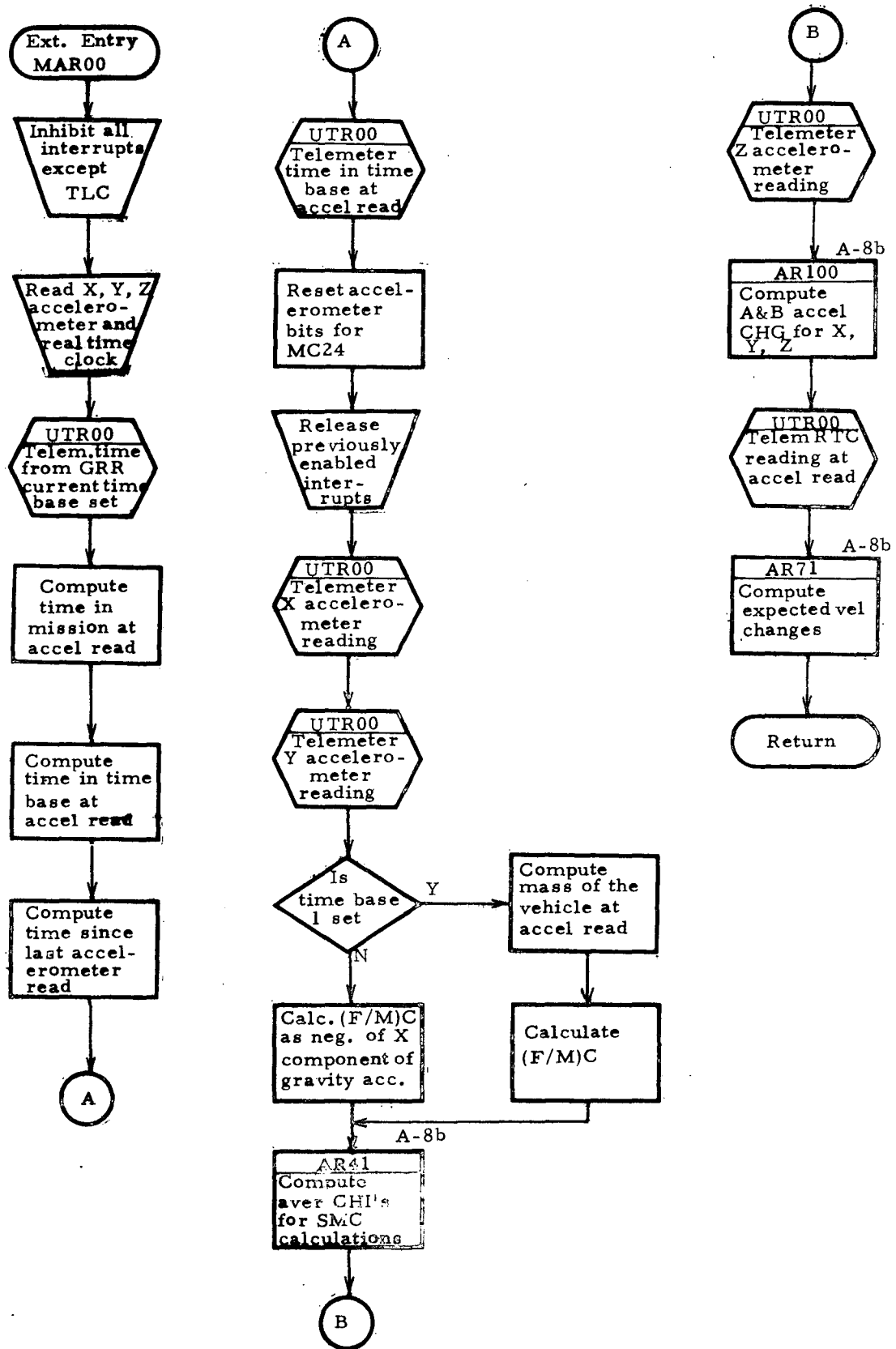


Figure A-8a

ACCELEROMETER PROCESSING
(continued)

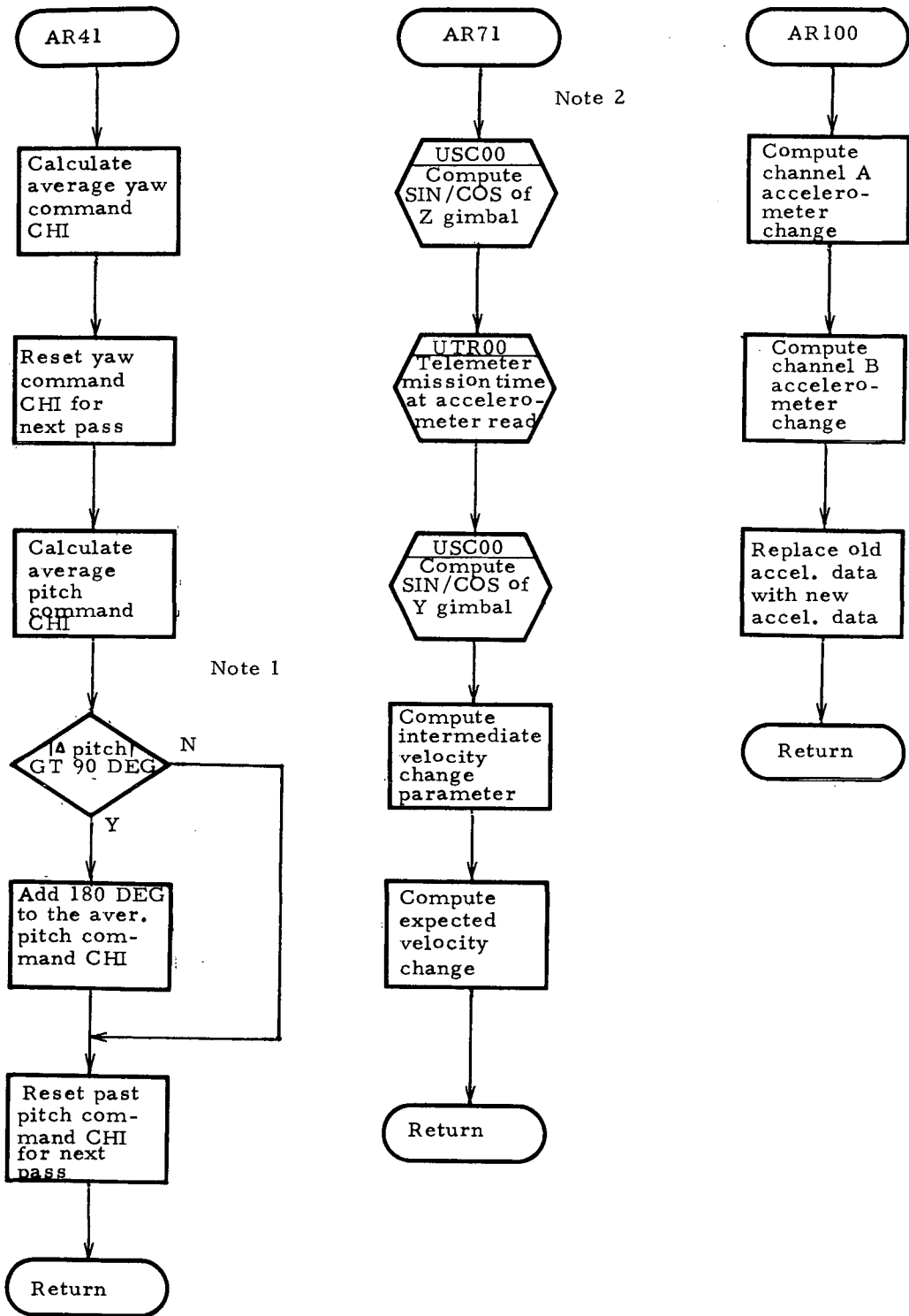


Figure A-8b

ACCELEROMETER PROCESSING
(continued)

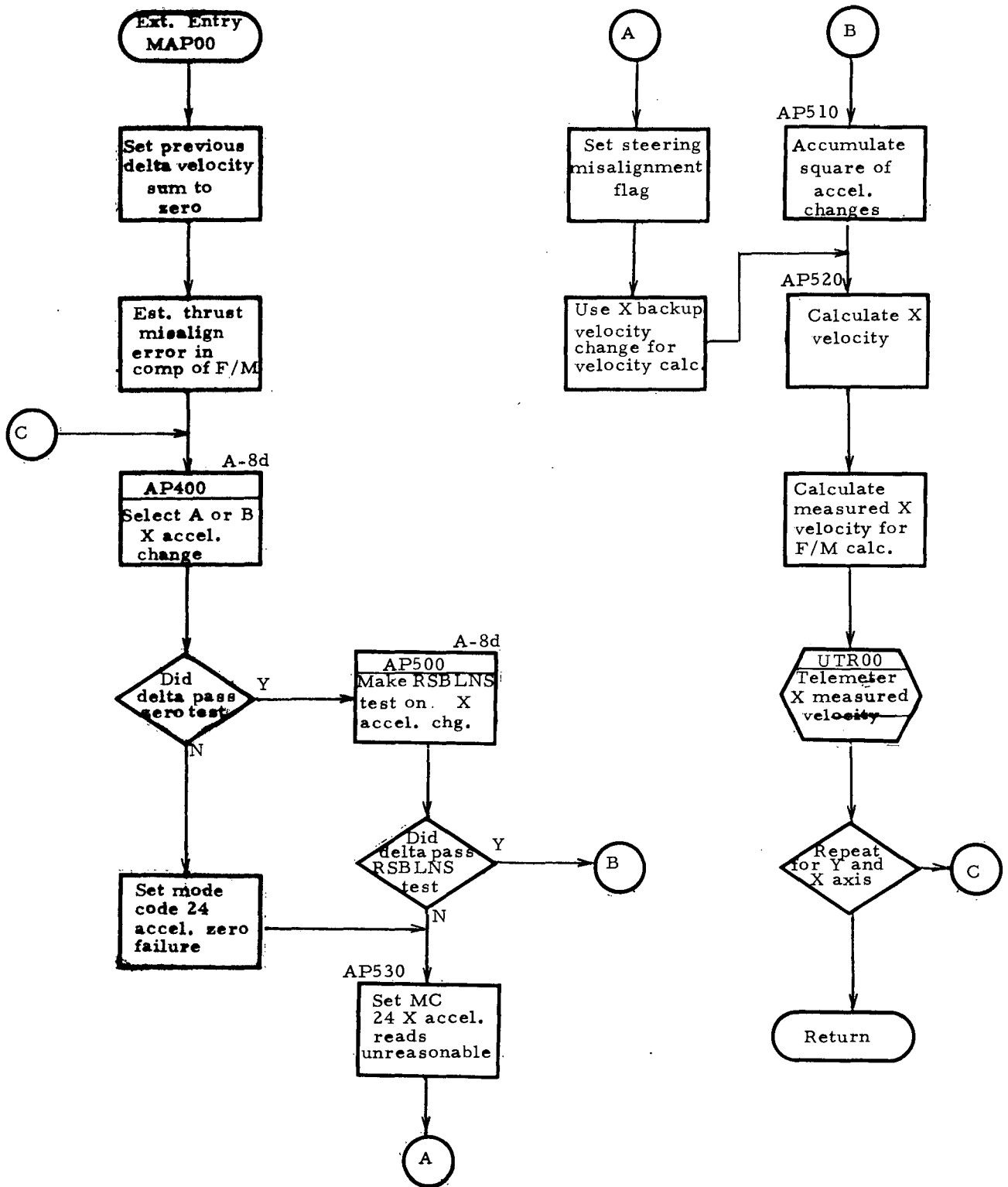


Figure A-8c

ACCELEROMETER PROCESSING
(continued)

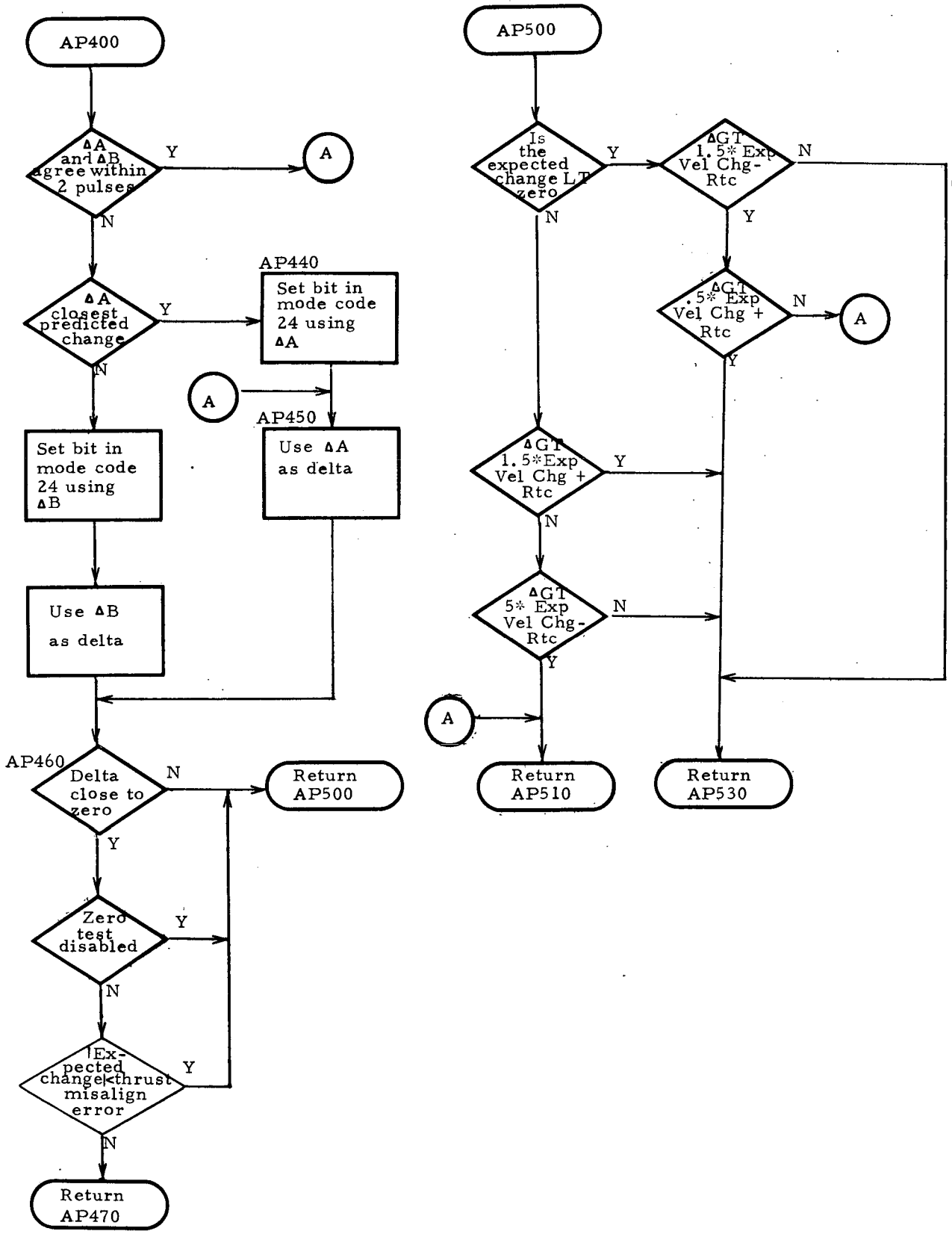


Figure A-8d

A.9 Minor Loop

A.9.1 Description of Operation

Vehicle attitude control is performed by the Minor Loop. In general terms, attitude control consists of determining actual attitude as indicated by vehicle sensors, calculating the attitude correction required to achieve the desired attitude specified by a guidance task, limiting the correction command, and issuing properly formatted attitude control commands to the vehicle control system.

To maintain vehicle stability, the Minor Loop is executed twenty-five times per second during boost phases and ten times per second in orbit. These high frequencies require the Minor Loop to be scheduled via the high-priority timer of the Interrupt Processor.

Vehicle attitude angles for yaw, pitch, and roll are sensed by inertial platform resolvers which measure the angles between the platform gimbals and the mounting frame. A fine and a coarse (backup) resolver are provided for each gimbal. The fine resolvers are selected until repeated errors cause a switch to be made to the backup resolvers. Each resolver contains redundant counter readings and a disagreement indicator which are used by the Minor Loop for validity checking.

After reading a resolver, the Minor Loop performs disagreement processing to select the proper counter. Reasonableness tests are then performed to detect invalid zero readings or an unreasonably large change from the previous reading. In the event that both counters of a resolver are bad, or if the selected counter fails the reasonableness tests, the corresponding vehicle attitude angle is not updated and the previous attitude control command is reissued. Error indicators are set to identify the type of failure. If the occurrence of resolver failures exceeds predefined frequencies, a switch is made to the corresponding backup resolver. Backup failures result in guidance reference failure indications and the last valid attitude command is issued repeatedly for the remainder of the mission.

Resolver readings which have been determined to be valid are converted to internal units and used to determine actual vehicle attitude. The actual attitude is then compared with the desired attitude and the difference is used to calculate attitude error commands to be issued to the attitude control system. Before the commands

are issued, however, they are limited to not exceed rate and magnitude tolerances.

A special entry point in the Minor Loop is provided for flight simulation tests so that ladder profiles may be generated.

A. 9.2 Unique Language Characteristics Required

While not specifically required, an indirect I/O capability would be useful. In the Minor Loop it may be desirable to read either fine or backup gimbals depending upon whether or not previous gimbal failures have occurred, or it may be desirable to not issue a read command at all as in a repeatable flight simulation test run. An indirect I/O capability is not mandatory since tests could be made to determine the type of I/O required. However, in a program such as the Minor Loop where time is of utmost importance, such tests would impose timing penalties.

Techniques are required for insuring that a given amount of time has elapsed between the issuance of gimbal read commands. Since the programmer loses sight of execution time in a high level language, the language should provide a means for determining such delta times and for specifying required time delays.

The relatively high execution frequencies of the Minor Loop (25/second in boost and 10/second in orbit) make minimizing execution time particularly desirable for this kernel. The ability to direct the compiler to minimize execution time, even at some cost in increased memory requirements, would be useful if the flight computer processing time capacity was near saturation.

A. 9.3 Flowchart Notes

Note 1

The Flight Simulation entry to the Minor Loop (MML00) is coded as a separate subroutine in CLASP, CMS-2, and HAL since these languages restrict a program module to a single entry point. It then calls the normal Minor Loop (MML20) rather than transferring control to a point within it as shown in the flowchart.

MINOR LOOP

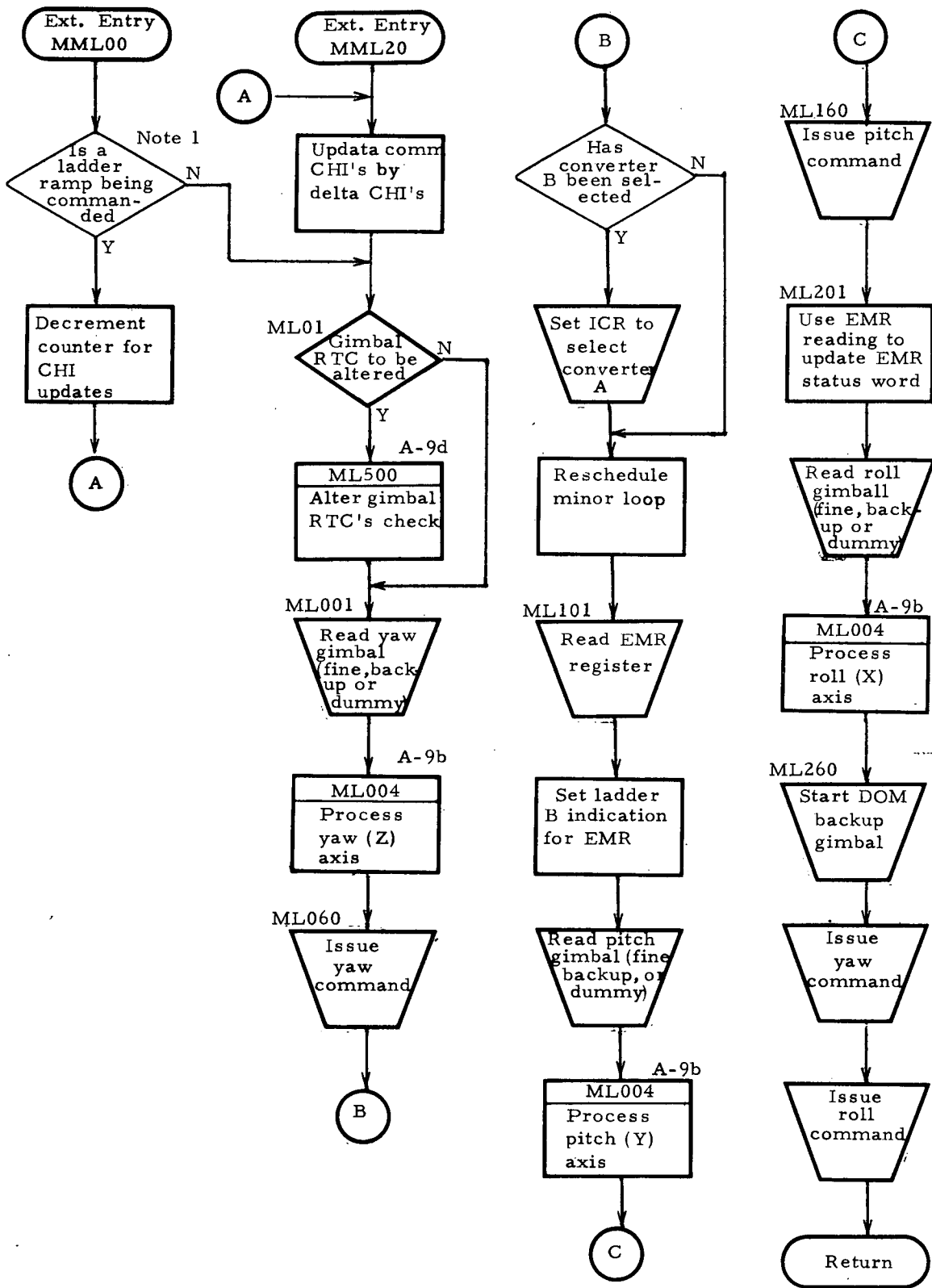


Figure A-9a

MINOR LOOP
(continued)

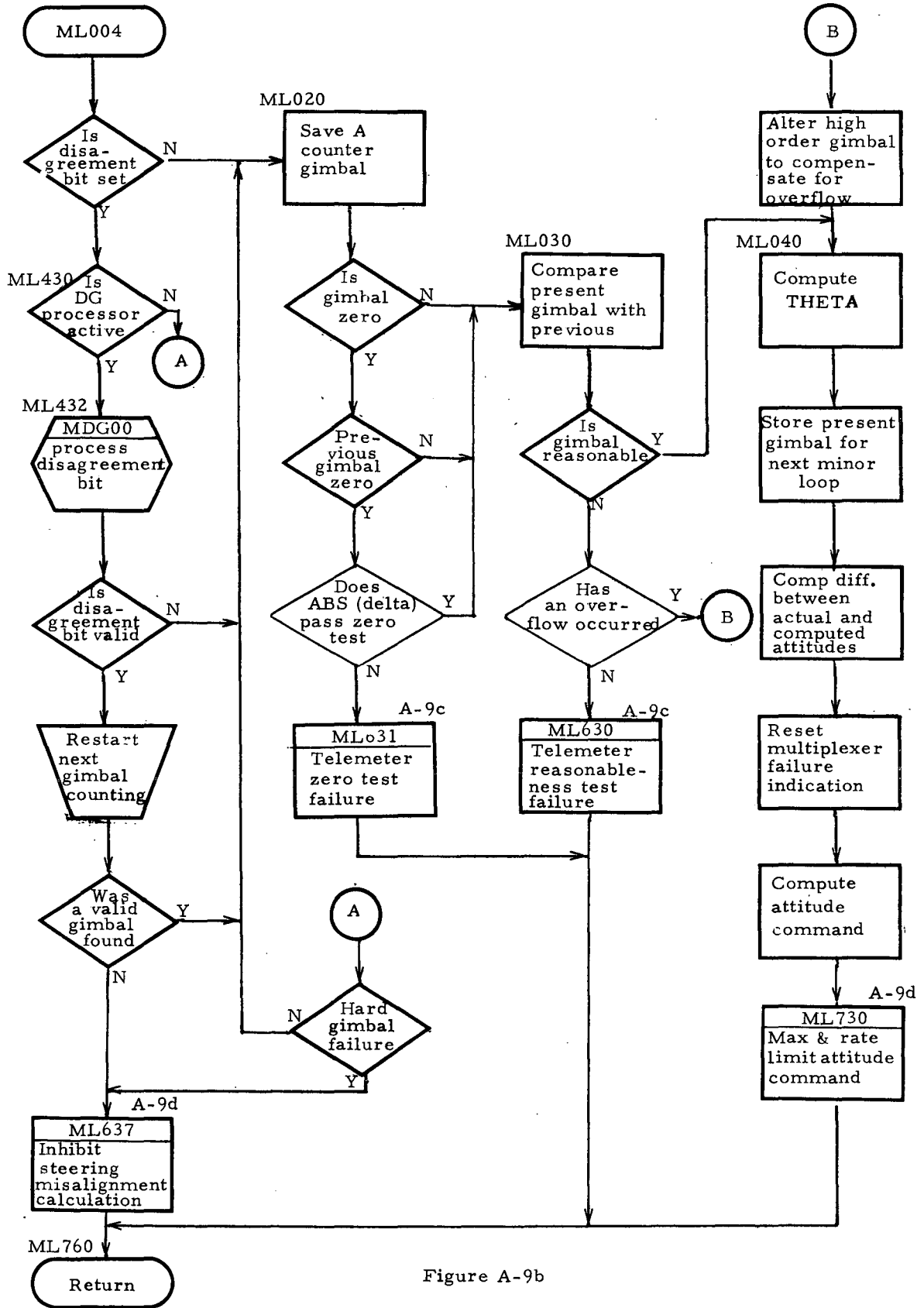


Figure A-9b

MINOR LOOP
(continued)

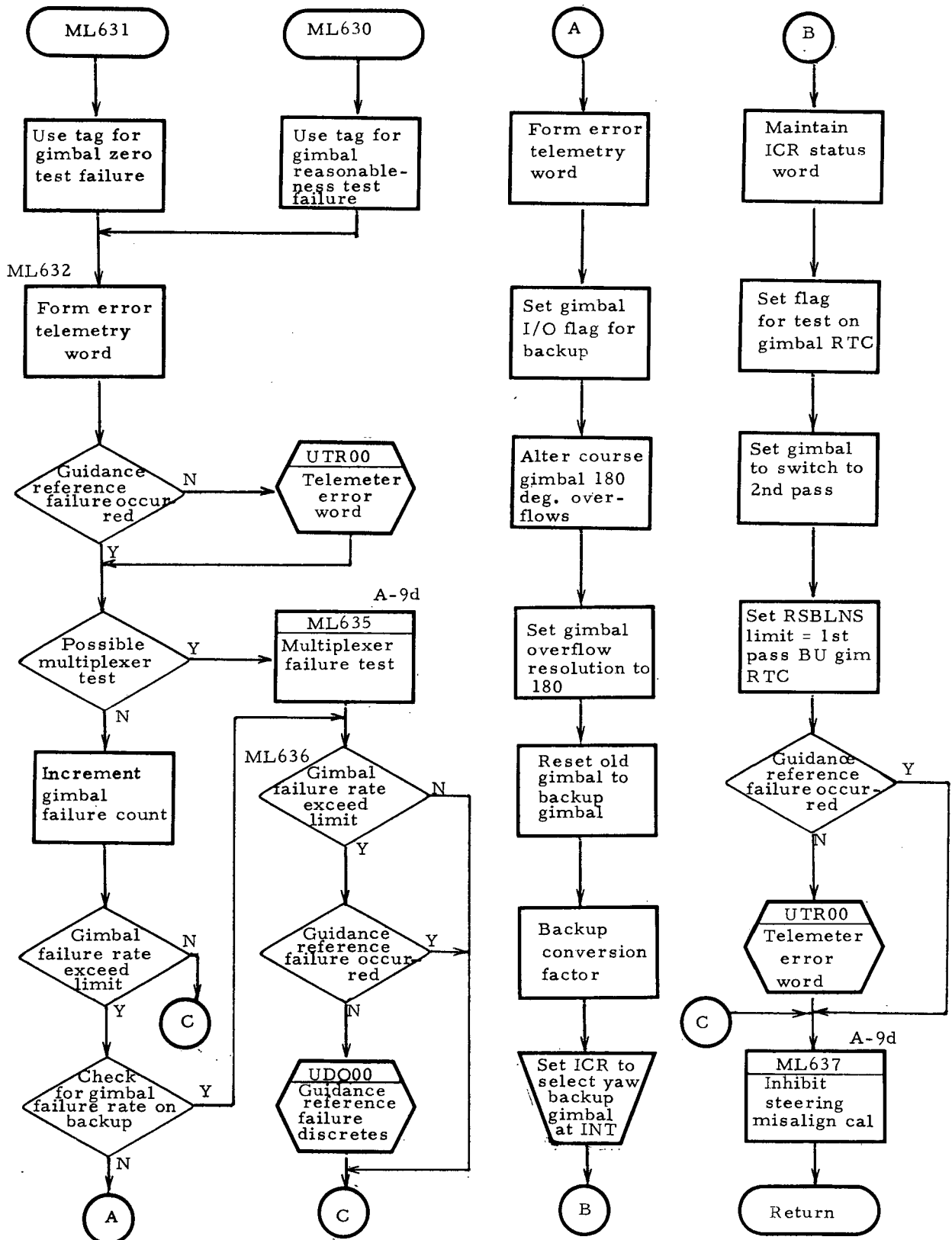


Figure A-9c

MINOR LOOP
(continued)

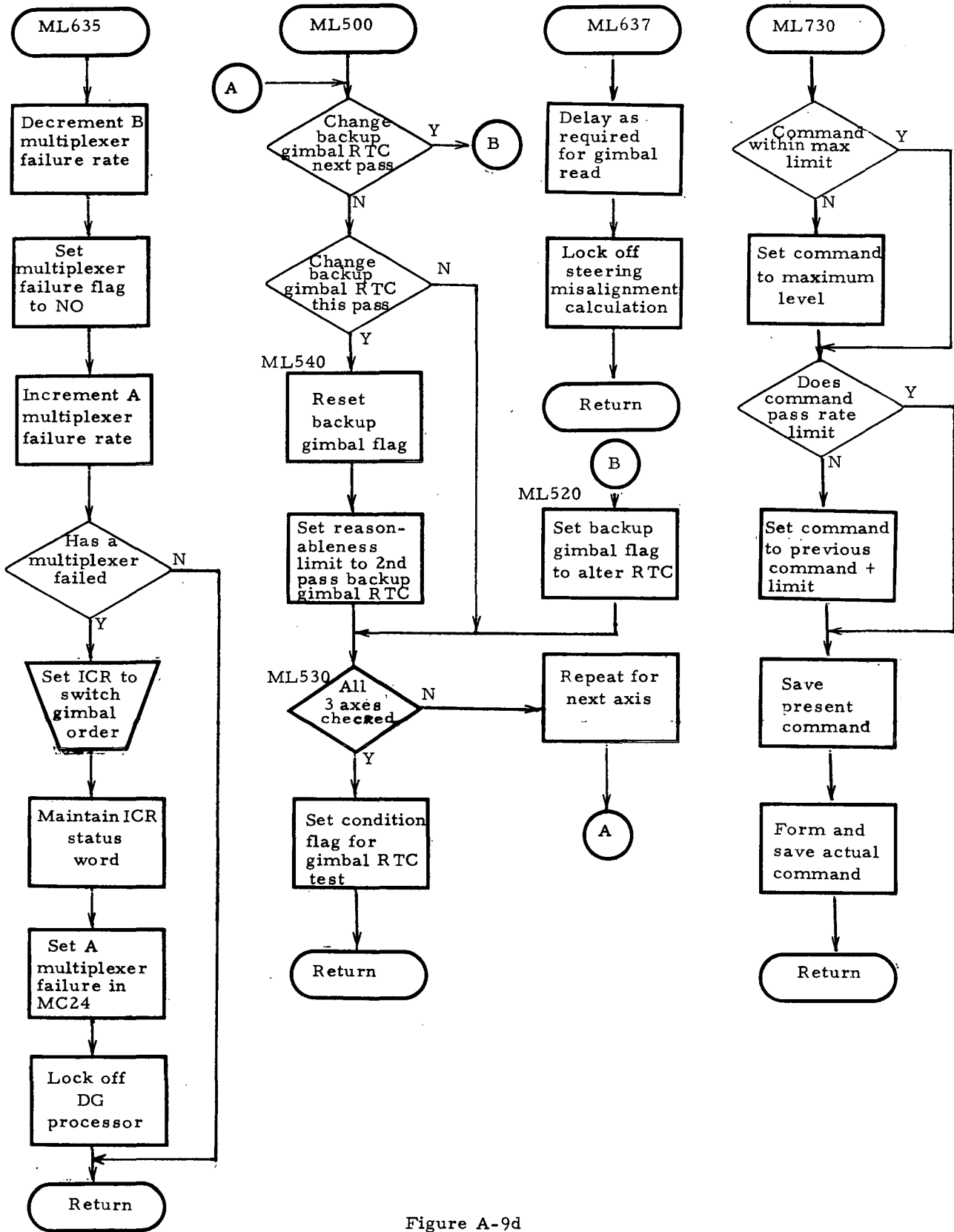


Figure A-9d

A.10 Switch Selector Processor

A.10.1 Description of Operation

Certain hardware functions of the Saturn vehicle are activated by the flight program via the issuance of switch selector output commands. The Switch Selector Processing task functions in much the same way as the Events Processor, in that it utilizes a predefined table containing the switch selector commands and their associated times for issuance. The time of activation for a given entry is used to schedule the Switch Selector Processor via the high-priority timer of the Interrupt Processor.

However, the process of issuing a switch selector is more involved than the function of initiating tasks performed by the Events Processor. The issuance of a single switch selector function requires at least five I/O operations to be performed:

- o Hung stage test
- o Issue stage and address
- o Verify address
- o Issue read command
- o Reset read command

In addition, if the hung stage test fails, a forced reset must be issued before the stage and address is issued. Also, if an address verification fails, a forced reset must be issued followed by the issuance of the stage and complemented address. Depending on the type of verification error, the system may be reconfigured to issue future switch selectors via different circuitry.

Hardware restrictions require timing delays between command issuance. Since the switch selector delays are on the order of ten to twenty-five milliseconds, the Switch Selector Processor re-schedules itself for execution at the proper time and returns control to the operating system. The interval is too long to be accomplished through an in-line delay.

In addition to the nominal sequence of switch selector functions,

provision is made for alternate sequences which can be activated as specified by other application tasks. Depending on the type of alternate sequence, the alternate switch selector functions will be issued instead of, or intermixed with, those of the nominal sequence.

Numerous entry points exist for the Switch Selector Processing kernel. Most of the entire are used for scheduling its various functions via the Timer 1 Scheduler as discussed above. Three additional entries are used to request an alternate sequence, to issue a forced reset, or to initialize Switch Selector Table pointers for a new time base.

A.10.2 Unique Language Characteristics Required

Requirements for an indirect I/O capability and for measuring short time periods are similar to those discussed for the Minor Loop (see paragraph A.9.3).

Although the decision-making statements are a common feature of nearly all programming languages, special emphasis on them here is warranted due to the unusually large number of decisions made in the Switch Selector Processor. This kernel places a premium on language capabilities which enable the programmer to express complex decision sequences in a logical and concise manner. The extent to which a language provides such capabilities contributes directly to the elimination of program logic errors and to an improvement in readability. Decision tables are particularly useful in this environment.

Another relatively common characteristic of the Switch Selector Processor is the manipulation of data at the bit level. It utilizes features for setting, resetting, and testing bits in status/control words and also for formatting and analyzing I/O data words. These functions require a language to provide bit-string handling facilities.

Also required is the ability to access data from tables. A subroutine is utilized to select the next switch selector command to be issued from one of a number of tables. Since alternate sequences can be interspersed and/or interleaved with switch selectors from the nominal sequence, the subroutine must be able to jump from table to table based on sequence decisions made by other programs.

A.10.3 Flowchart Notes

Note 1

Switch Selector Processing consists of a number of interrelated functions, some of which use common program logic. In order to minimize program duplication, the functions were all combined into a single program module and invoked via multiple entry points as shown in the flowcharts. The kernel was actually coded that way in SPL. However, for the other three languages, where multiple entry points are not allowed, a common entry point (MSS00) was utilized wherein control was transferred to the appropriate function. The decision logic for this transfer of control is not shown on the flowcharts but in each of the three languages (CLASP, CMS-2, and HAL) it consisted of a computed GOTO.

SWITCH SELECTOR PROCESSOR

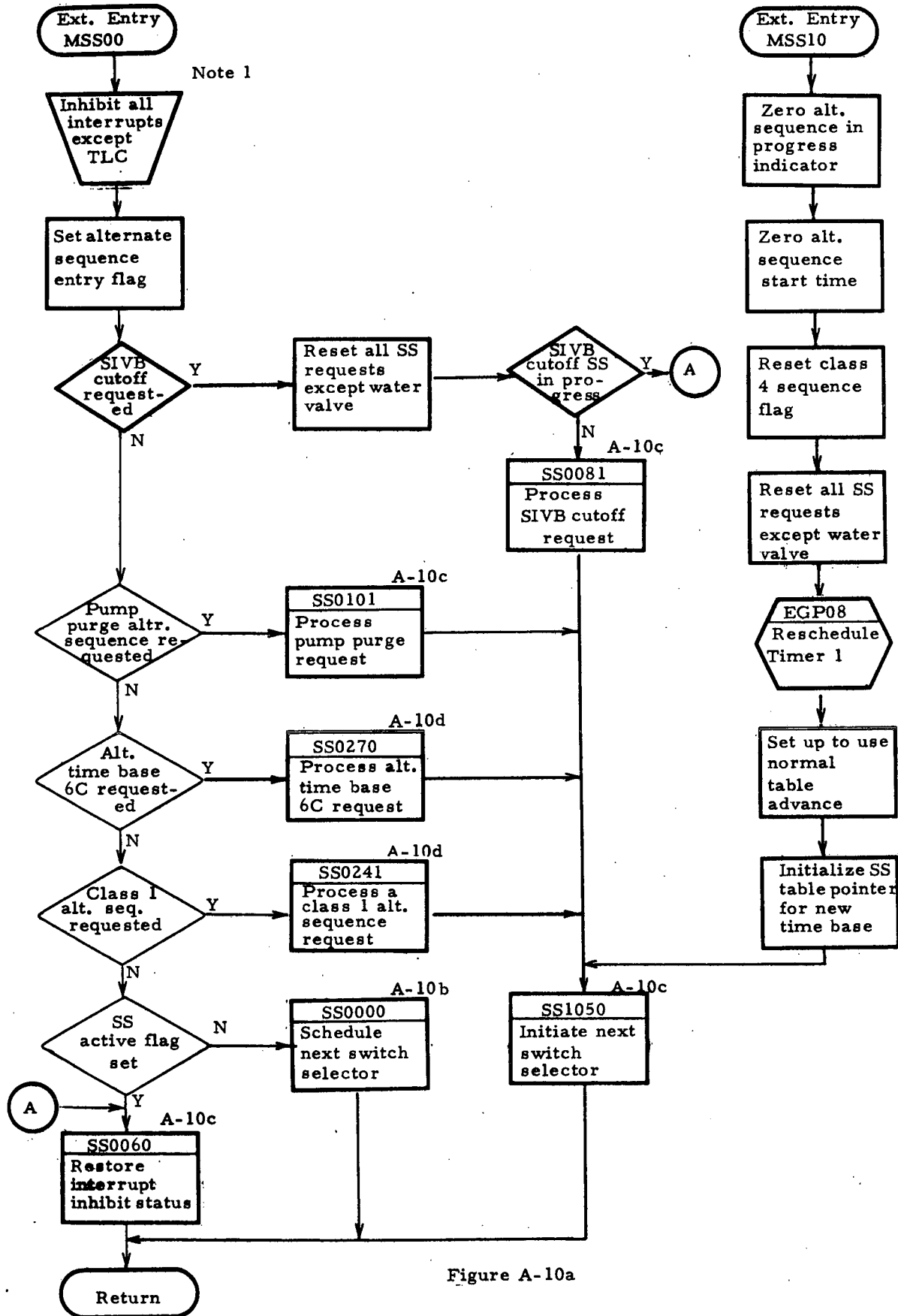


Figure A-10a

SWITCH SELECTOR PROCFSSOR
(continued)

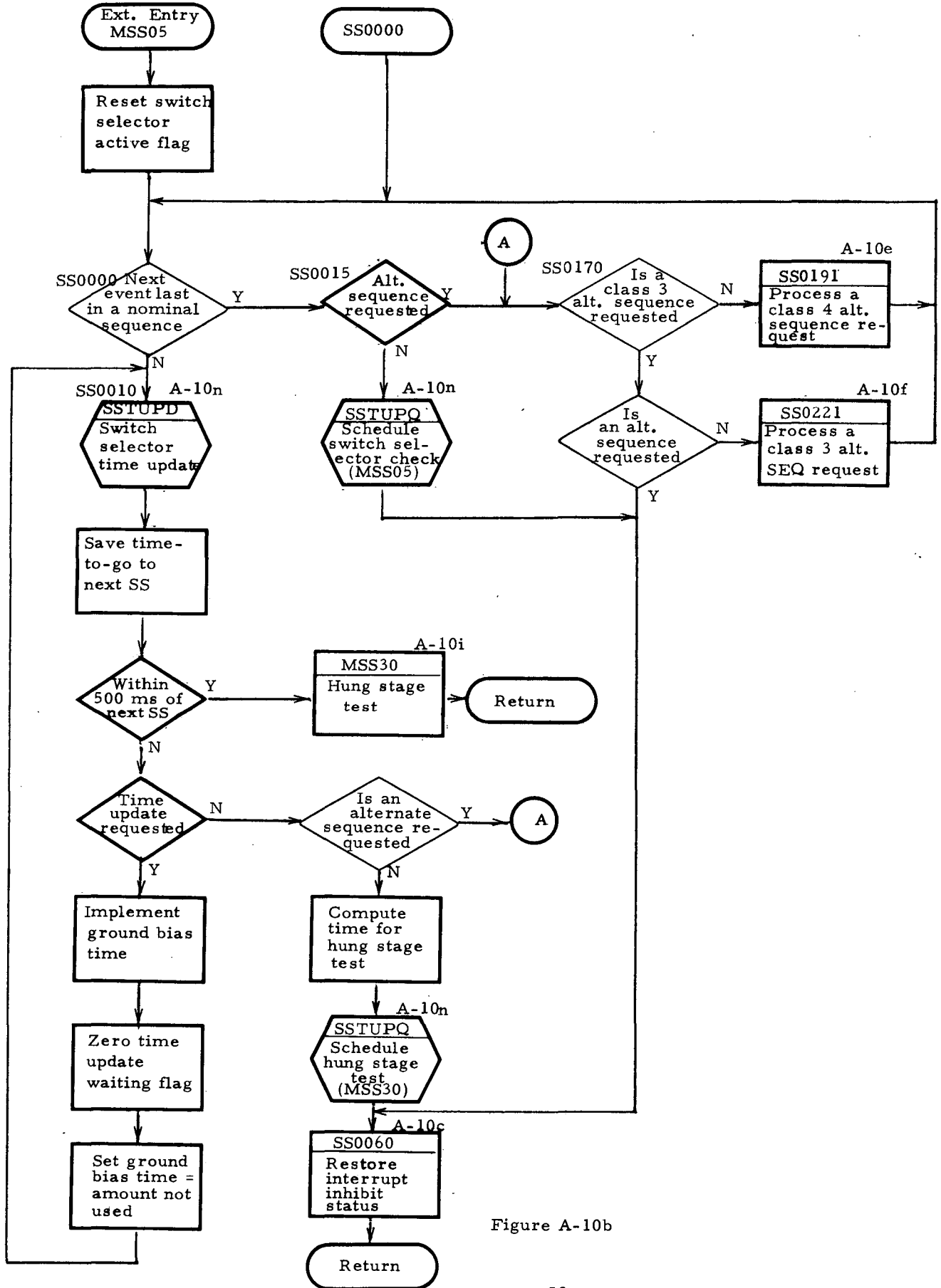


Figure A-10b

SWITCH SELECTOR PROCESSOR
(continued)

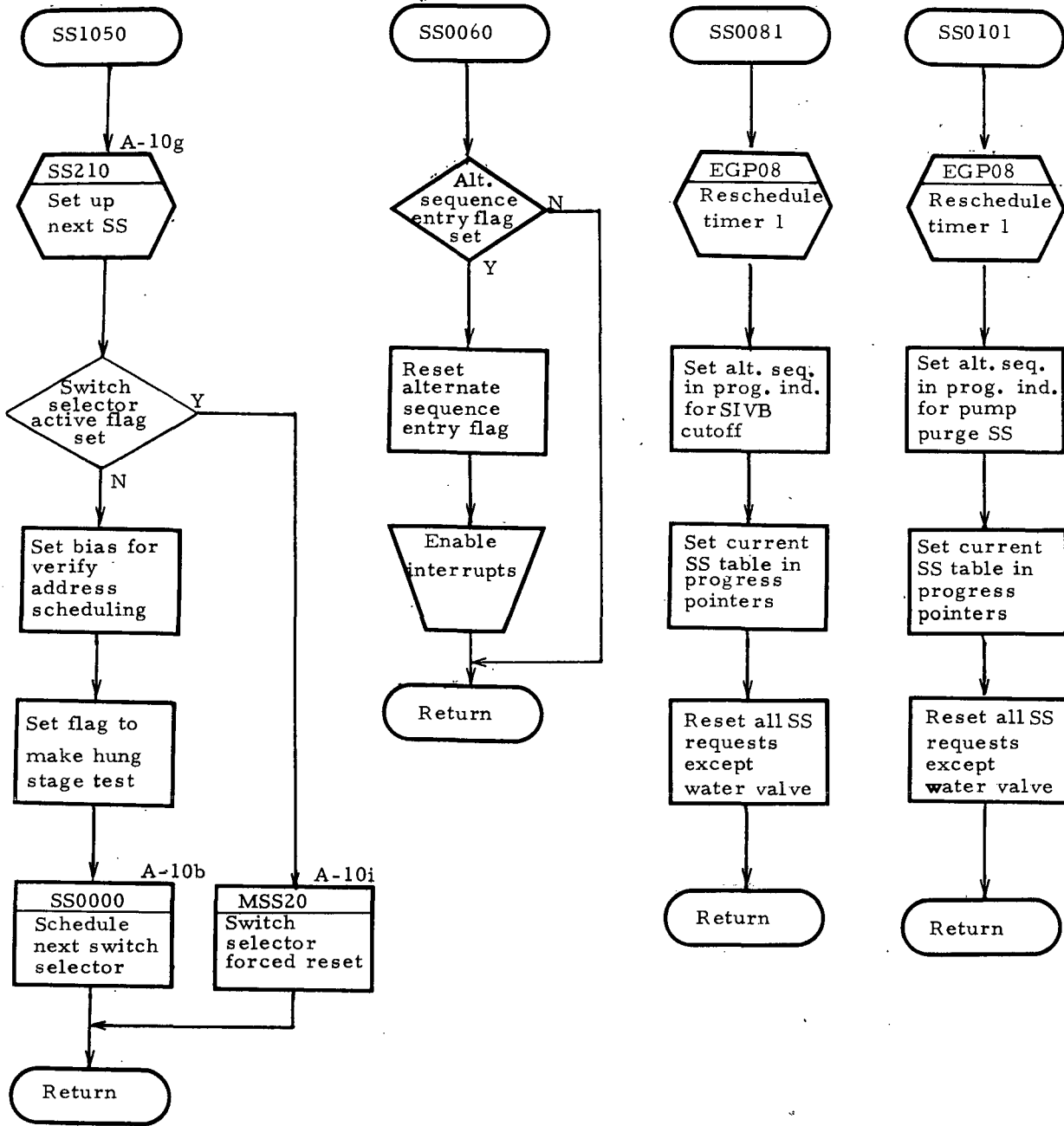


Figure A-10c

SWITCH SELECTOR PROCESSOR
(continued)

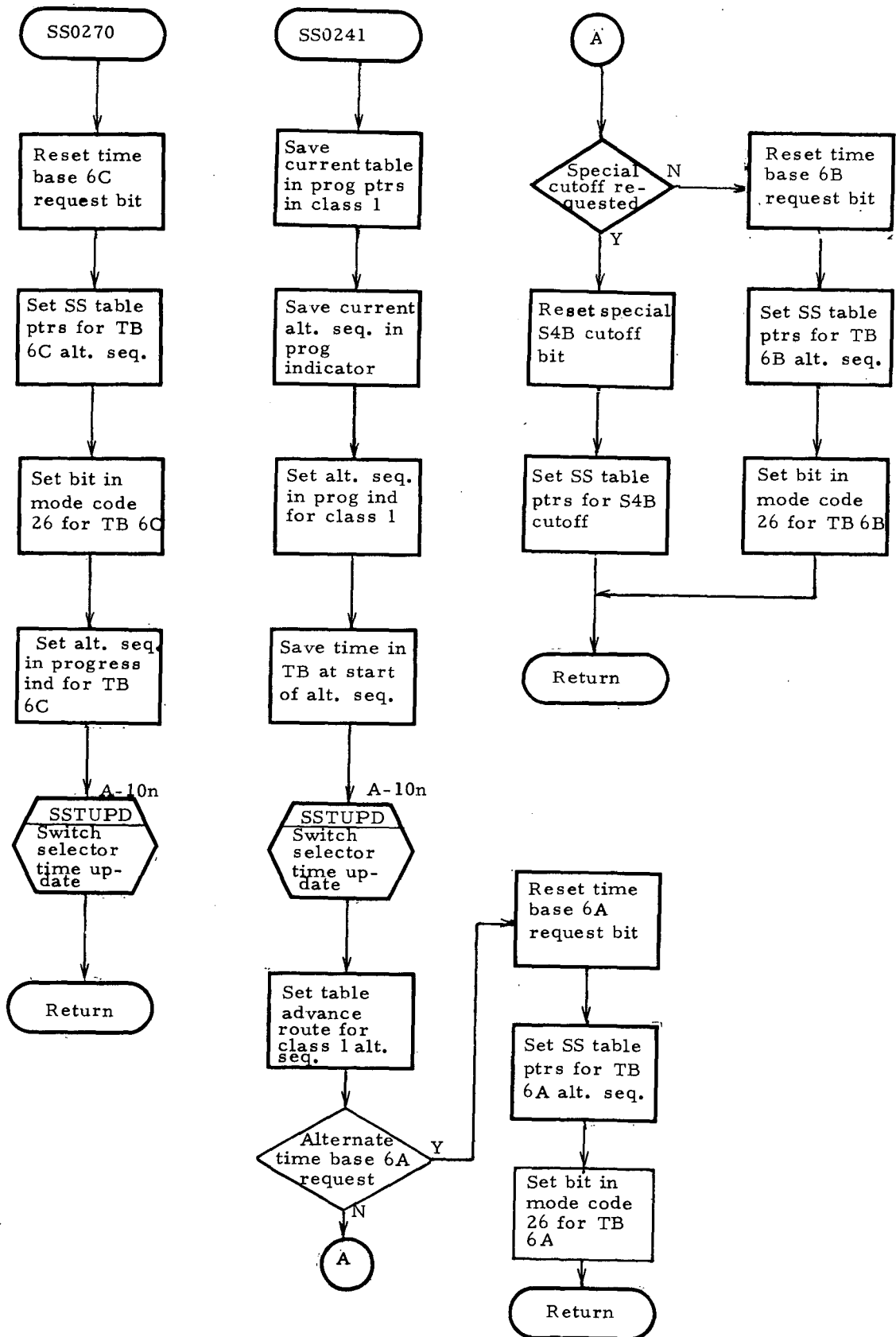


Figure A-10d

SWITCH SELECTOR PROCESSOR
(continued)

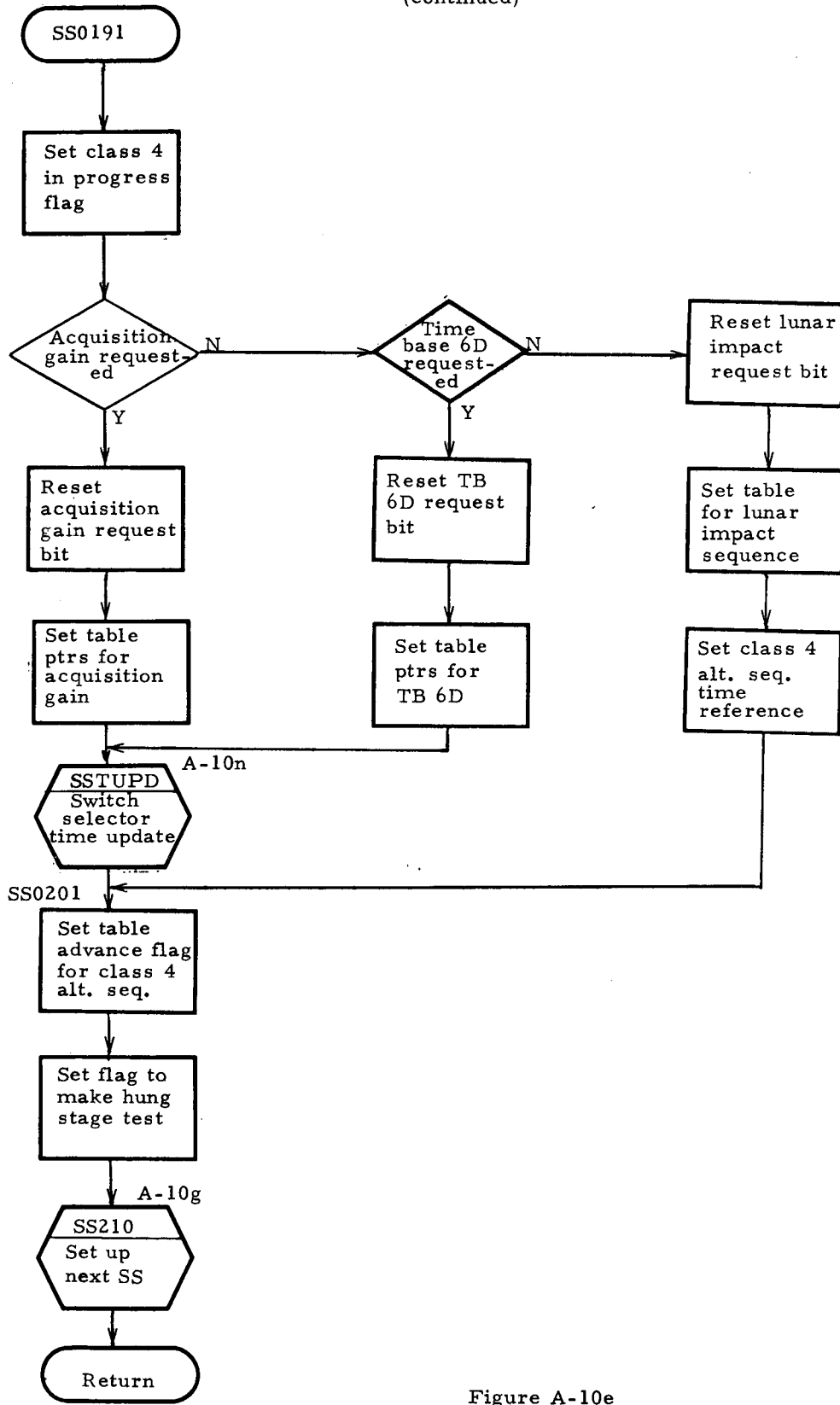


Figure A-10e

SWITCH SELECTOR PROCFSSOR
(continued)

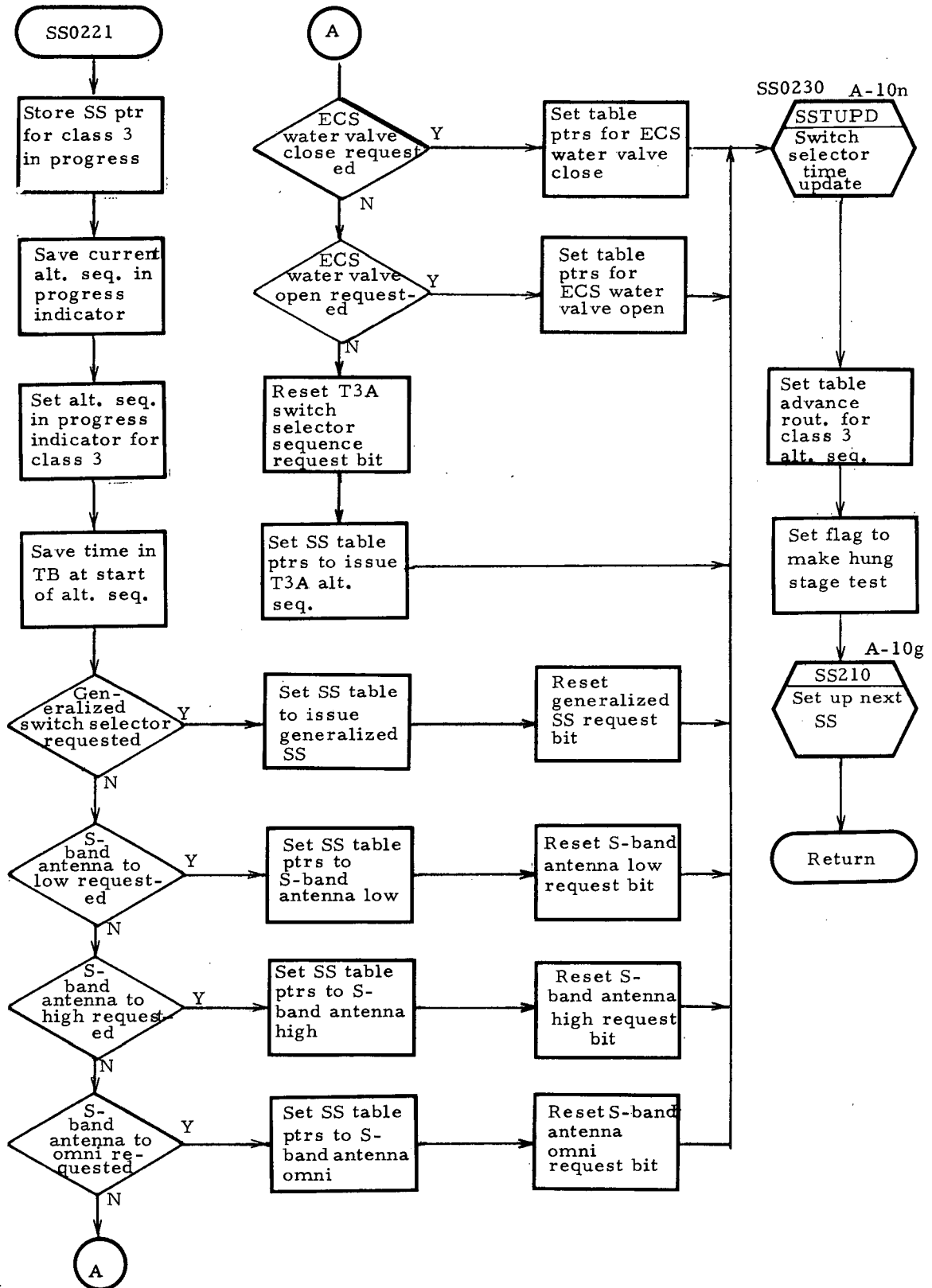


Figure A-10f

SWITCH SELECTOR PROCESSOR
(continued)

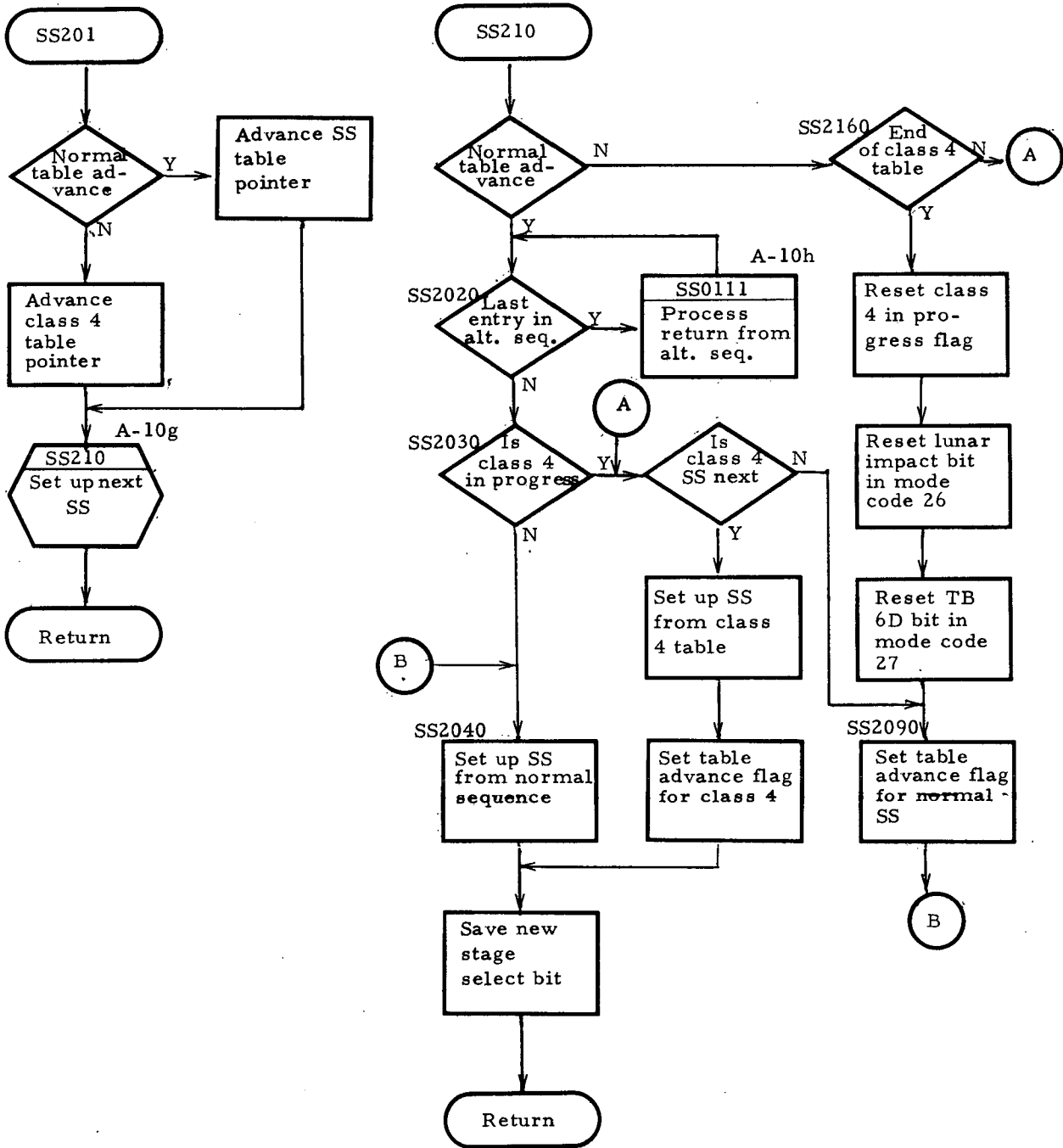


Figure A-10g

SWITCH SELECTOR PROCESSOR
(continued)

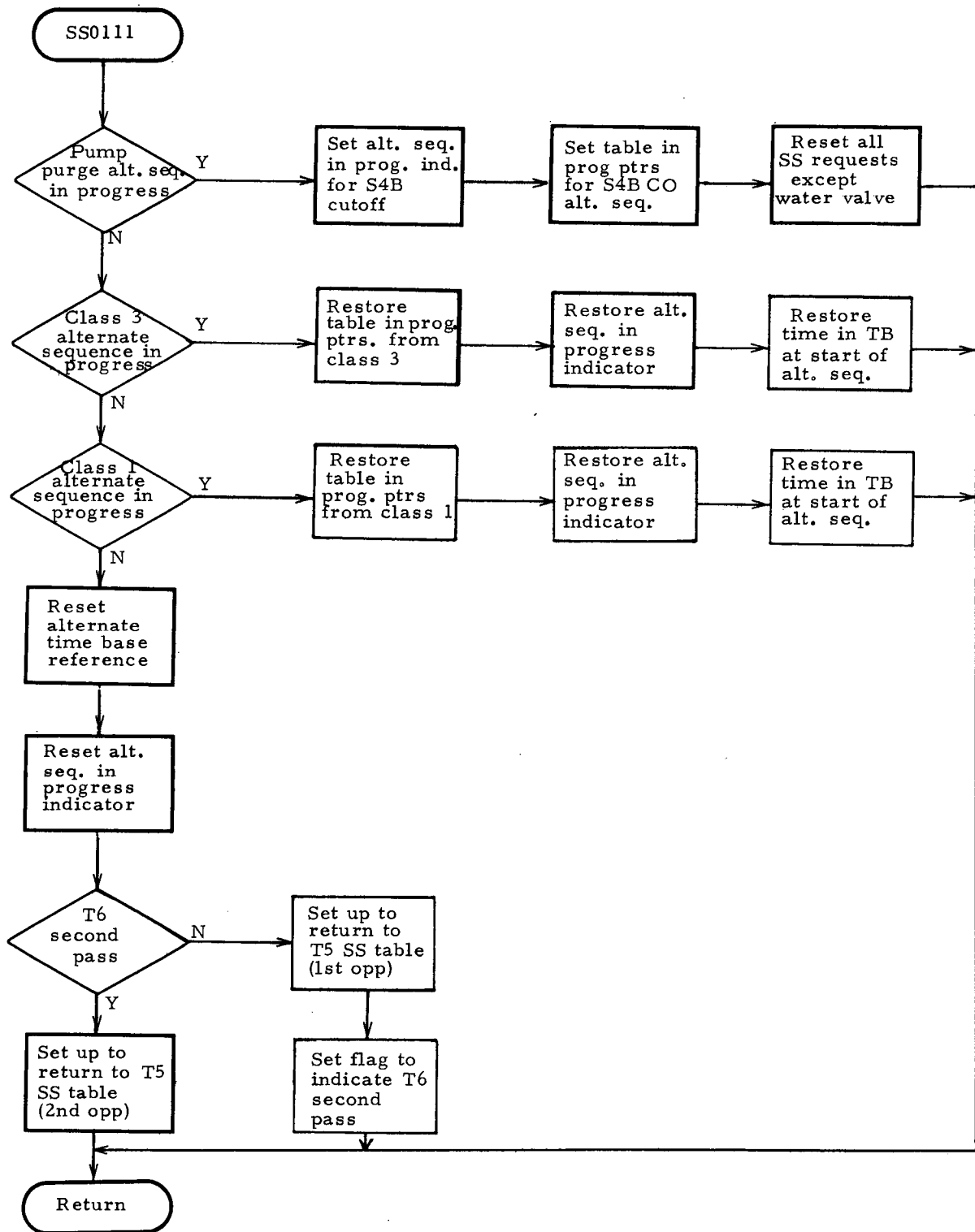


Figure A-10h

SWITCH SELECTOR PROCESSOR
(continued)

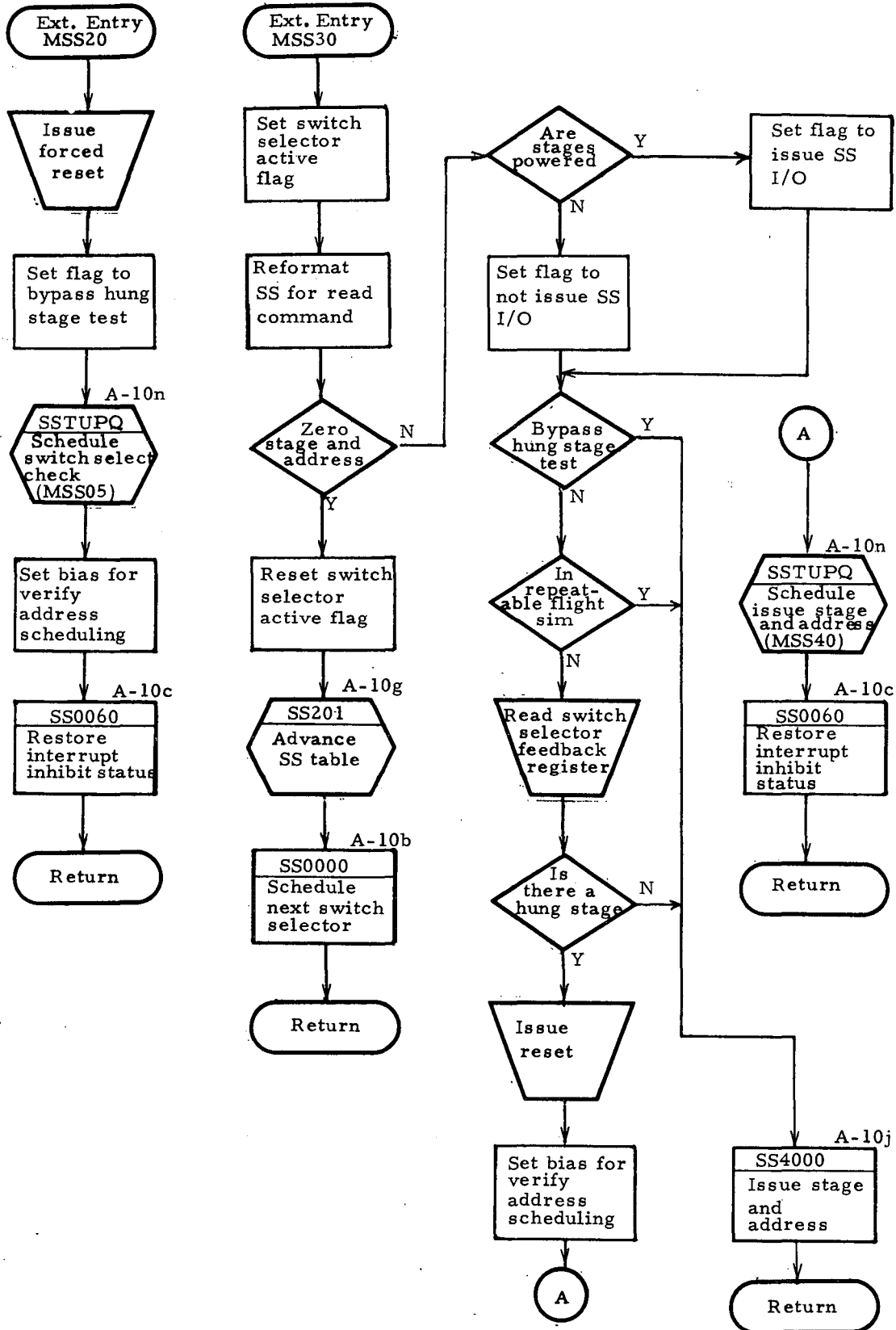


Figure A-10i

SWITCH SELECTOR PROCESSOR
(continued)

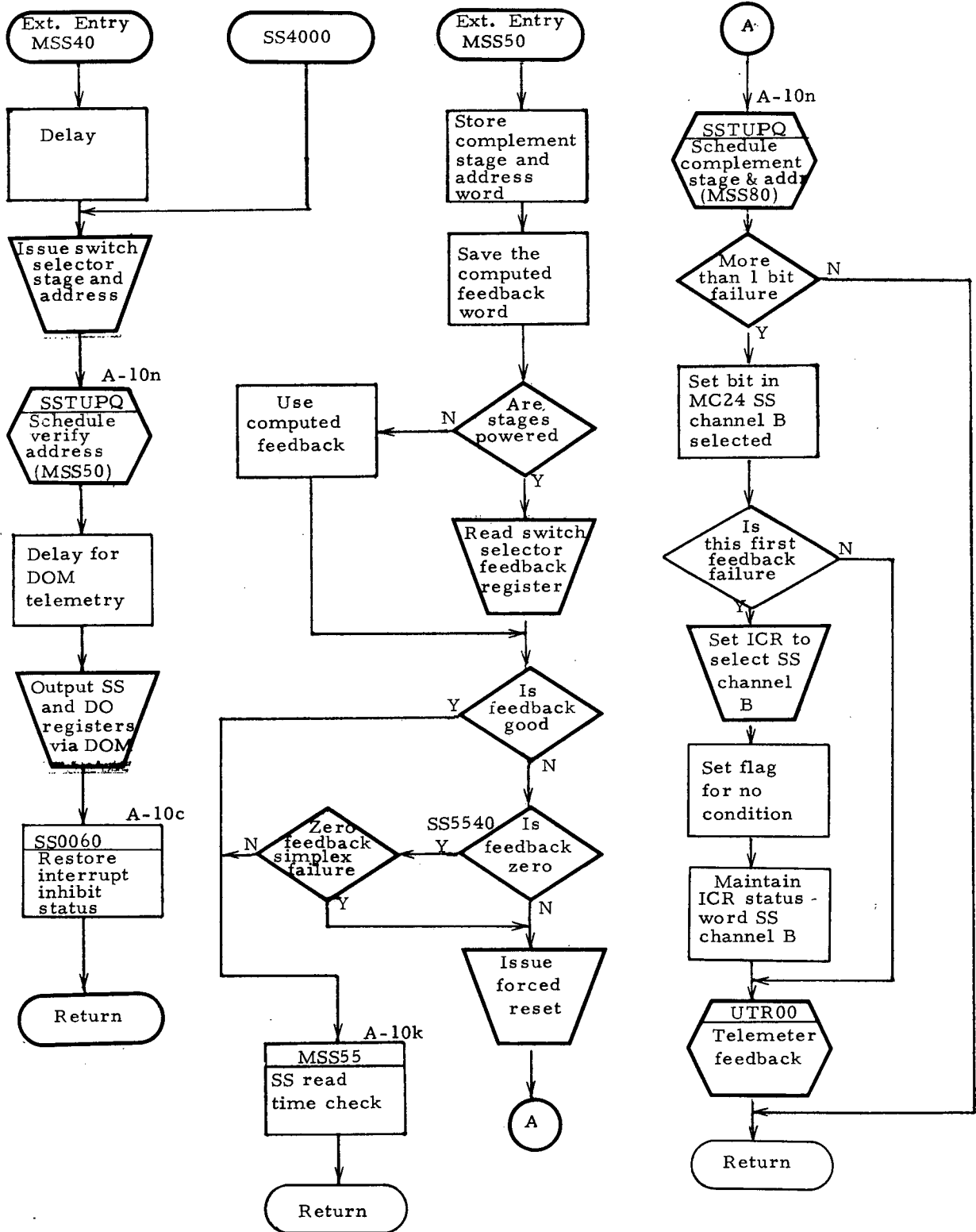


Figure A-10j

SWITCH SELECTOR PROCESSOR
(continued)

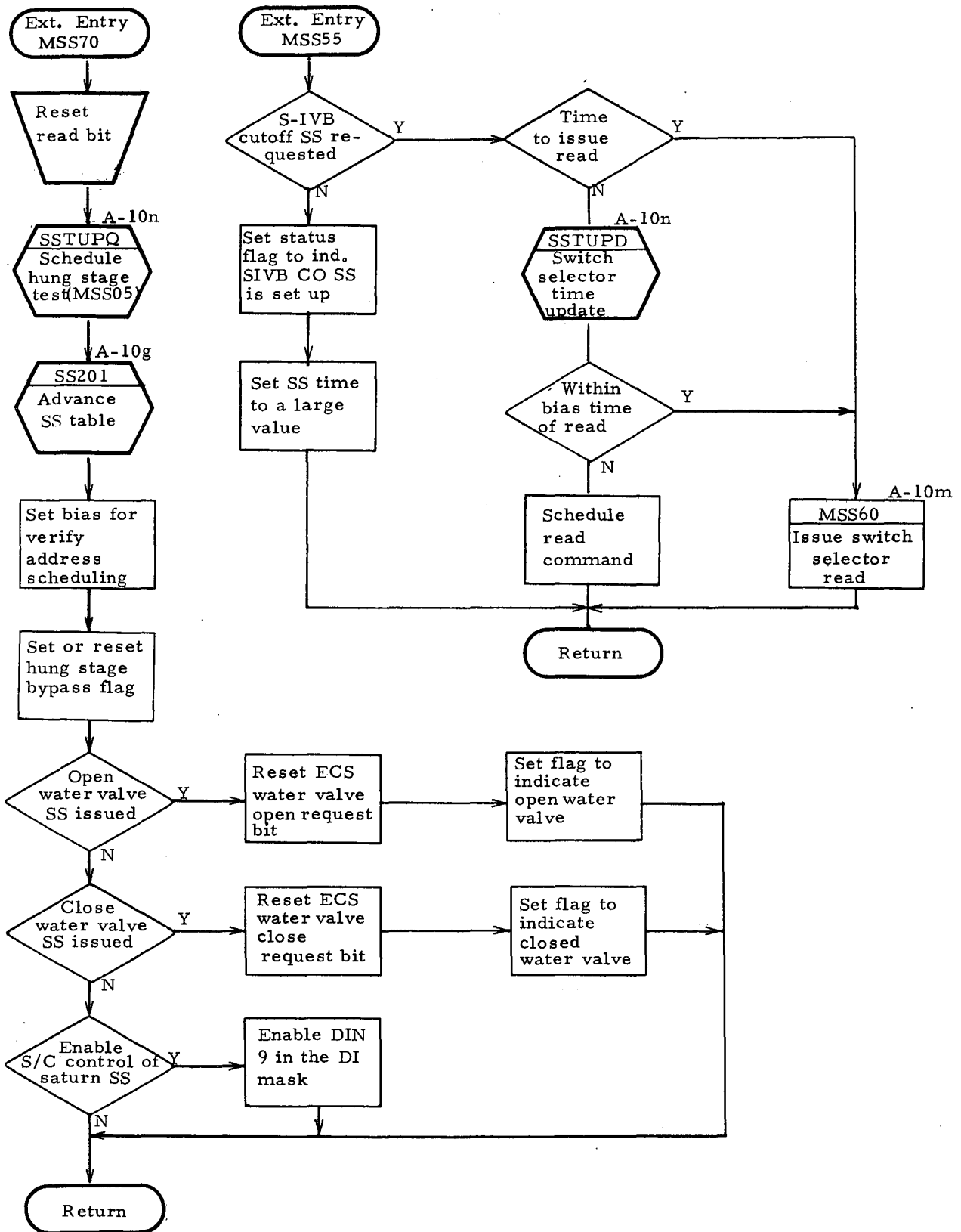


Figure A-10k

SWITCH SELECTOR PROCESSOR
(continued)

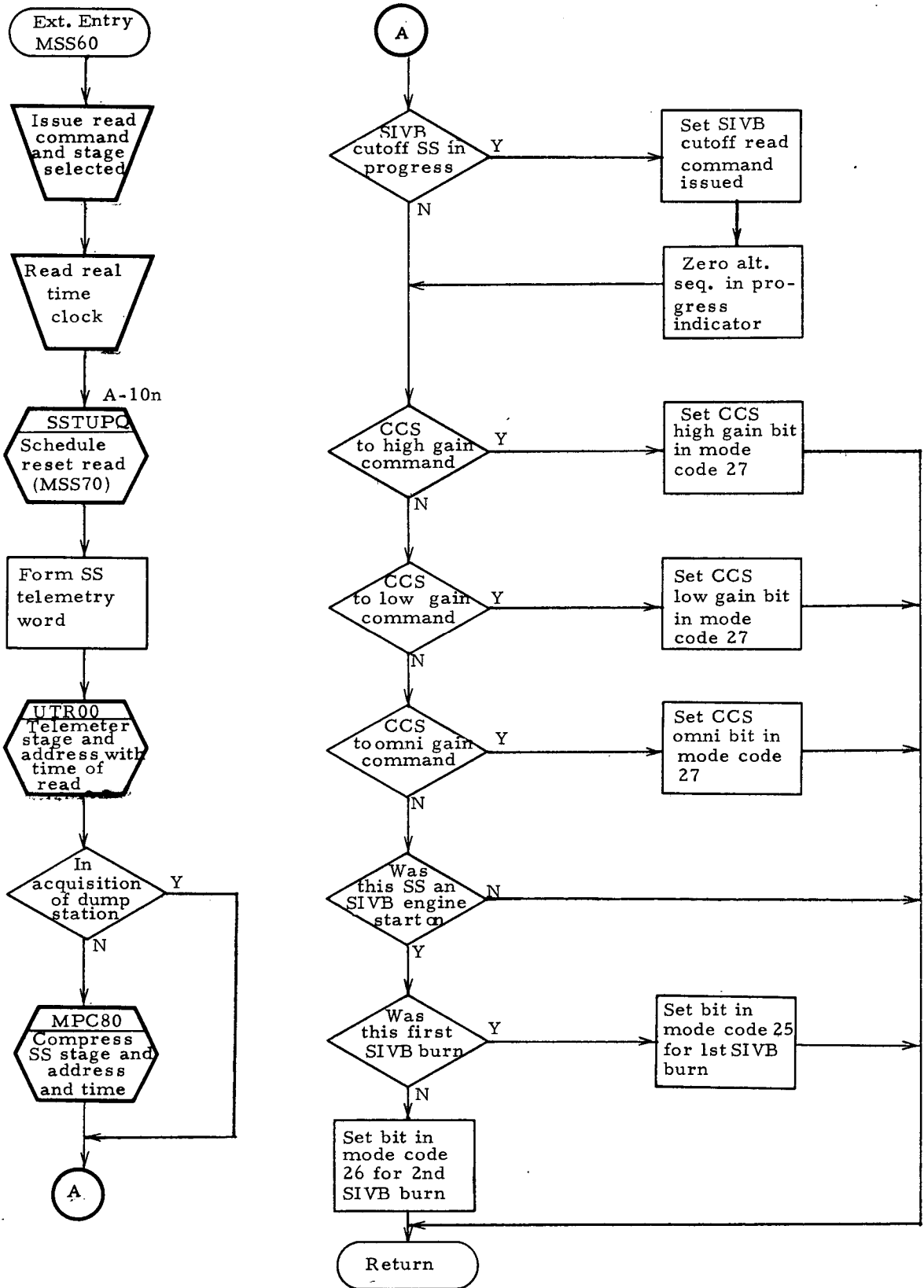


Figure A-10m

SWITCH SELECTOR PROCESSOR
(continued)

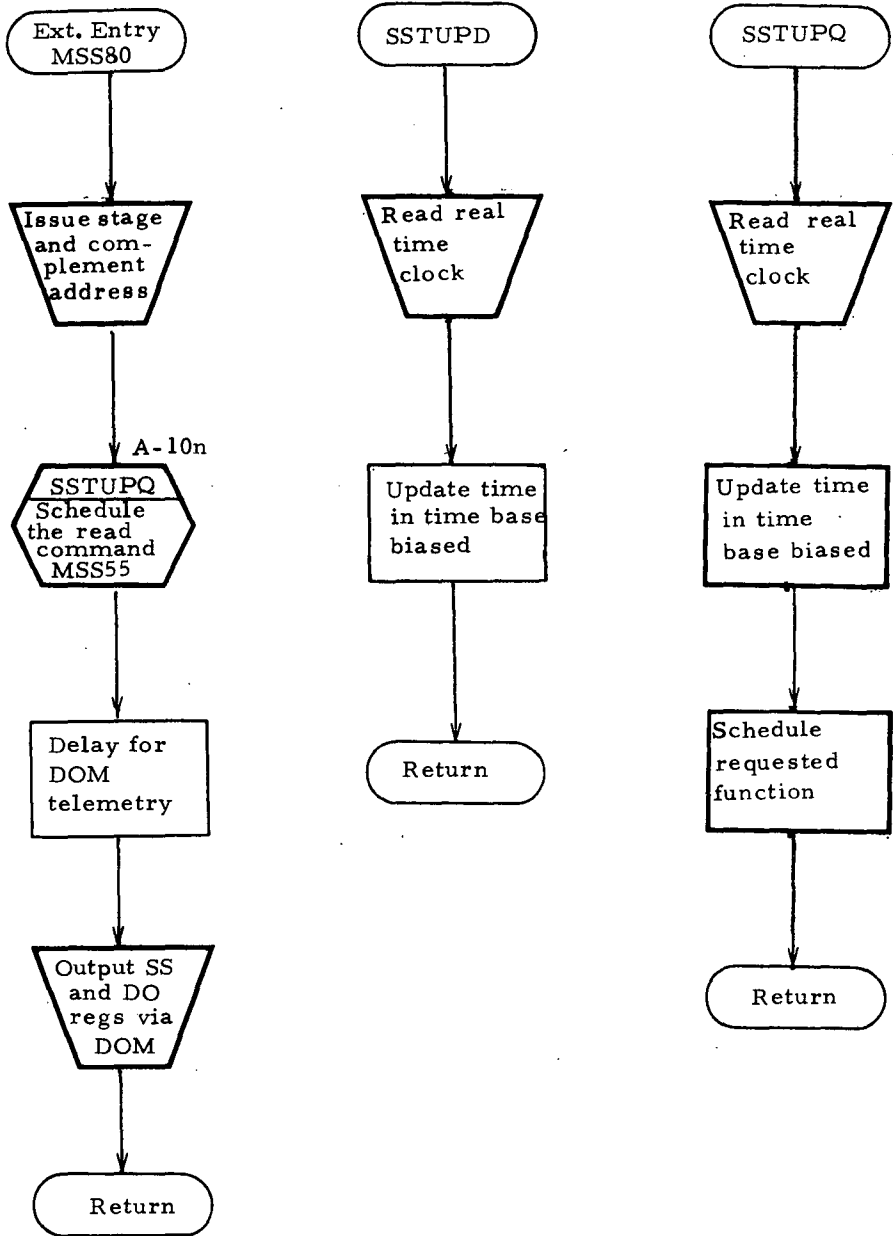


Figure A-10n

A.11 Task Keying (ATMDC)

A.11.1 Description of Operation

Task Keying is an operating system function associated with priority task scheduling; it is the process of entering information concerning a task into a Priority Control Table to enable the task to be dispatched (initiated) on a priority basis. The information includes such items as task priority level, the in-core address of the task, and initial register contents for the task.

Since multiple tasks can usually be keyed for execution on a given priority level, various techniques are used for stacking the additional entries. In the ATMDC operating system, the Priority Control Table (Table A-4) holds a single entry for each priority level. Additional entries are stored in a Priority Overflow Table (Table A-5) with all entries for a given priority level chained together.

Requirements for task keying vary with the design of the operating system. For the ATMDC Flight Program, tasks are keyed in response to events (interrupts or discrettes), based on time, or as requested by another application task.

A.11.2 Unique Language Characteristics Required

The Task Keying kernel requires facilities for formatting and accessing tables. Techniques for linking the overflow entries together in an efficient manner are also desirable.

The kernel also implies a requirement for the capability to identify the task to be keyed. The keying process itself does not require it since the Task ID is simply stored into a table. However, since this is done for the express purpose of dispatching the task (passing control to it) at a later time, the Task ID must provide the means by which the task can be located in core.

A.11.3 Assumptions Made During Coding

Several assumptions were made for the purpose of organizing the control tables. It was assumed that there were ten priority levels in the operating system and that twenty-five entries in the overflow table would suffice. Also, it was assumed that three hardware registers required saving for each task. These assumptions affect only the size of the control tables and could be easily adjusted.

PRIORITY CONTROL TABLE

| | Task ID | Reg 1 Contents | Reg 2 Contents | Reg 3 Contents | Overflow Chain Link |
|-----------|---------|-------------------|-------------------|-------------------|------------------------|
| Level 0 | | | | | |
| Level 1 | | | | | |
| Level 2 | | | | | |
| ⋮ | | | | | |
| ⋮ | | | | | |
| ⋮ | | | | | |
| ⋮ | | | | | |
| Level N-1 | | | | | |

Notes:

- 1) Number of priority levels (N) depends on system requirements. Ten levels were assumed for the kernel.
- 2) During the keying process, the Task ID is either the memory address of the task entry point or some other indicator which can be used to locate the task in memory. After a task has been initiated, this word is used to store the address where task execution is to resume following an interruption. A value of zero for a Task ID indicates that no tasks are currently assigned to that priority level.
- 3) Register storage words are used to save task registers when a task is interrupted. They are initialized to zero when a task first receives control. The number saved depends on system requirements and was arbitrarily chosen as three for the kernel.
- 4) The Overflow Chain Link is either a pointer or an index used to chain task entries together whenever more than one task has been assigned to a given priority level. The additional entries are stored in the Priority Overflow Table. A value of zero indicates no overflow entries exist for that priority level.

Table A-4

PRIORITY OVERFLOW TABLE

| | Overflow Chain Link | Task ID |
|---------|---------------------|---------|
| Entry 0 | | |
| Entry 1 | | |
| Entry 2 | | |
| ⋮ | | |
| ⋮ | | |
| ⋮ | | |
| ⋮ | | |
| Entry M | | |

Notes:

- 1) The Overflow Chain Link has the same meaning as its counterpart in the Priority Control Table. A value of zero indicates end of chain.
- 2) The Task ID also has the same meaning as its counterpart in the Priority Control Table. A value of zero indicates that the entry is not currently assigned.

Table A-5

ATM TASK KEYING

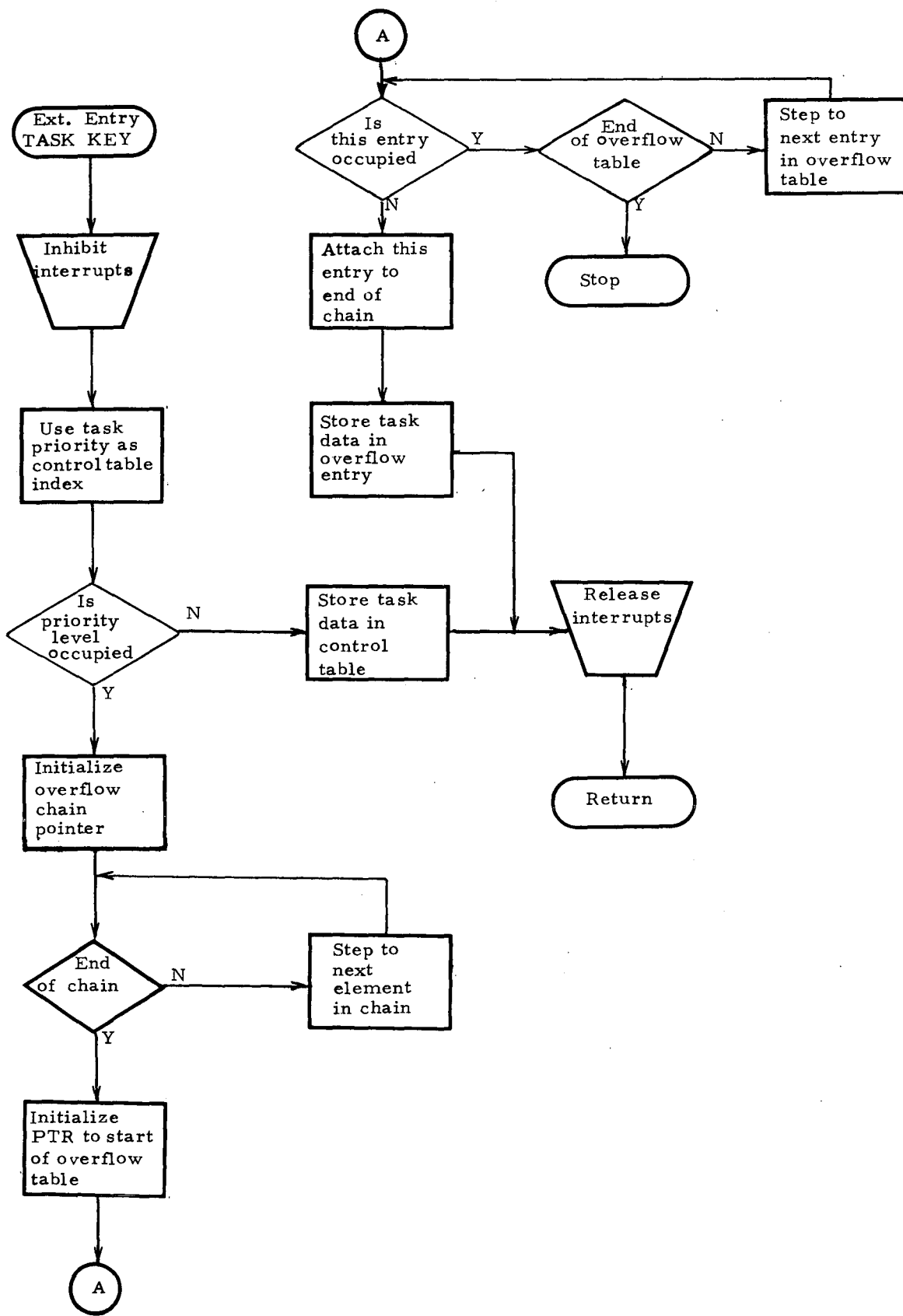


Figure A-11

A.12 Glossaries

The Glossary Tables provided in this paragraph document most of the names declared in the flight program coding. These glossaries are provided as an assistance to reading the flight program listings in Appendix B. It also documents assumptions made about system-defined names.

A.12.1 Input/Output Glossary

Table A-6 contains names and brief descriptions of external devices accessed by the flight program kernels. The File Names are the names used in the actual Input/Output statements. In the HAL coding it is assumed that these names are assigned by the system and are known to the compiler, because of HAL's device-oriented input/output. In SPL the names assigned by the system cannot be used directly in Input/Output statements; a FILE statement must be used to define input/output arguments in terms of system-assigned names. Therefore, for SPL the Device Names of Table A-6 were assumed to be assigned by the system, and the File Names were declared through the FILE statement. In CLASP and CMS-2 input/output is indicated by comments rather than statements of the language.

A.12.2 Interrupt Glossary

Table A-7 contains names of computer interrupts which were assumed to be assigned by the system. The Description identifies the LVDC interrupt corresponding to the Interrupt Name.

A.12.3 Data Glossary

Table A-8 contains names and brief descriptions of common data items declared in the kernel coding. Minor deviations appear in the listings such as break characters in HAL names, and truncation of some names to meet CLASP's eight-character limitation. However, these deviations are easily associated with the corresponding names listed in the Data Glossary.

INPUT/OUTPUT GLOSSARY

| <u>File Name</u> | <u>Device Name</u> | <u>Description</u> |
|------------------|--------------------|--------------------------------|
| CLOCK | TIMER | Real time clock |
| DBG | DOMBUGIM | DOM backup gimbal |
| DCS | DCSINREG | Digital command system input |
| DIR | DISINREG | Discrete input register |
| DOM | SSDOM | Switch selector DOM output |
| DOR | DISOUTRES | Reset discrete output register |
| DOS | DISOUTSET | Set discrete output register |
| EMR | EMREG | Error monitor register |
| ICR | ICREG | Internal control register |
| SS | SSREG | Switch selector command output |
| SSFB | SSFDBK | Switch selector feedback |
| TIM1 | TIMER1 | Timer 1 counter |
| TIM2 | TIMER2 | Timer 2 counter |
| XACC | XACCEL | X-axis accelerometer |
| XBGIM | XBACKUP | X-axis backup gimbal |
| XGIM | XGIMBAL | X-axis fine gimbal |
| XLAB | XLADDER | X-axis ladder |
| YACC | YACCEL | Y-axis accelerometer |
| YBGIM | YBACKUP | Y-axis backup gimbal |
| YGIM | YGIMBAL | Y-axis fine gimbal |
| YLAD | YLADDER | Y-axis ladder |
| ZACC | ZACCEL | Z-axis accelerometer |
| ZBGIM | ZBACKUP | Z-axis backup gimbal |
| ZGIM | ZGIMBAL | Z-axis fine gimbal |
| ZLAD | ZLADDER | Z-axis ladder |

Table A-6

INTERRUPT GLOSSARY

| <u>Interrupt Names</u> | <u>Description</u> |
|------------------------|----------------------|
| T1INT | Timer 1 interrupt |
| T2INT | Timer 2 interrupt |
| TLCINT | TLC interrupt |
| EX1INT | External 1 interrupt |
| EX2INT | External 2 interrupt |
| EX3INT | External 3 interrupt |
| EX4INT | External 4 interrupt |
| EX5INT | External 5 interrupt |
| EX6INT | External 6 interrupt |
| EX7INT | External 7 interrupt |
| EX8INT | External 8 interrupt |
| EX9INT | External 9 interrupt |

Table A-7

DATA GLOSSARY

| <u>Data Name</u> | <u>Description</u> |
|------------------|---|
| CHIBARSTEER | CHI bar steering in progress flag |
| COSTHETA | Cosine of angle between pseudo-nodal vector and descending node |
| DCSDATACOUNT | Input data word count |
| DCSDATCT | Table of function data word requirements |
| DCSERLIM | Limit on errors for a given function |
| DCSERmm | Error tags |
| DCSINDX | Table index derived from mode command |
| DCSMODE | Mode command table |
| DCSMSTAT | Function status table |
| DCSSTCOD | Function telemetry status code table |
| DELTAL3 | Correction to velocity-to-be gained |
| DELTA VVP | Estimated velocity to be gained |
| DELTA2 | IGM intermediate parameter |
| DFACQ | Acquisition gain indicator |
| DFDBF | Disagreement multiplexer failure flag |
| DFDTL | Sector dump in progress flag |
| DFILE | Flight program status word |
| DFIL1 | Timer 2 interrupt level in progress indicator |
| DFIL2 | External interrupt level in progress indicator |
| DFIL3 | Timer 1 interrupt level in progress indicator |
| DFLT | Flight/sim flight indicator word |
| DFMDI | Flight mode indicator |
| DFSMC | Steering misalignment flag |
| DFTBCEP | Time base change indicator for events processor |
| DFTUP | Time update waiting indicator |
| DFWV | Flag which indicates state of water valve (open or close) |
| DFZER | Zero test enable flag |
| DGSSM | Switch selector function to be scheduled |
| DGST2 | Timer 2 function to be scheduled |
| DKAPI | Flight phase status table |
| DKMIR | Minor loop initial rate |
| DKT1 | Timer from GRR when time base 1 was set |
| DLPRL | Periodic processor task rate table |
| DLPTL | Periodic processor task delta T table |

Table A-8

DATA GLOSSARY
(continued)

| <u>Data Name</u> | <u>Description</u> |
|------------------|---|
| DLTTL | Timer 2 task execution time table |
| DPHII | Rate of change of range angle |
| DPHIT | Rate of change of predicted terminal range angle |
| DQST2 | Timer 2 function to be en queued |
| DTBID | Time base indicator |
| DVAC | Accelerometer reading |
| DVACT | Real time clock (RTC) reading associated with DVTAS |
| DVASW | Switch Selector Request status |
| DVA1 | } Coefficients used to convert the attitude corrections from the inertial platform frame to the body frame |
| DVA2 | |
| DVA3 | |
| DVA4 | |
| DVA5 | |
| DVA6 | |
| DVCA | Average of present and past Minor Loop commanded CHI at the time of major computer cycle accelerometer read |
| DVCC | Commanded CHI used in Minor Loop |
| DVD | Intermediate velocity change parameter |
| DVDA | Optisyn A change in velocity |
| DVDB | Optisyn B change in velocity |
| DVDC | Delta CHI |
| DVDGS | Count of disagreement bit hardware failures |
| DVDM | Measured velocity of platform |
| DVDPM | Mask word that specifies which DIN's are to be processed when they change from OFF to ON |
| DVDT | Elapsed time between current and previous major computer cycle accelerometer readings in seconds |
| DVEMR | Error Monitor Register |
| DVEOF | Engine out multiplication factor for backup F/M calculation |

Table A-8
(continued)

DATA GLOSSARY
(continued)

| <u>Data Name</u> | <u>Description</u> |
|------------------|---|
| DVERT | Time error associated with time update |
| DVF | Expected platform velocity change |
| DVFCM | Generated acceleration (backup) |
| DVFOM | Total vehicle acceleration determined from accelerometer readings |
| DVFOR | Thrust of vehicle (backup) |
| DVG | Gravity acceleration |
| DVHDA | Count of A multiplexer failures |
| DVHDB | Count of B multiplexer failures |
| DVIA5 | Temporary storage |
| DVICR | Internal Control Register |
| DVIH | Interrupt inhibit image |
| DVLDB | Ladder converter B selection rate per second |
| DVLRC | Ladder ramp commanded CHI update counter |
| DVMAS | Mass of vehicle (backup) |
| DVMC4 | Mode Code 24 |
| DVMC5 | Mode Code 25 |
| DVMC6 | Mode Code 26 |
| DVMC7 | Mode Code 27 |
| DVMFR | Mass flow rate of vehicle (backup) |
| DVMLD | Minor Loop |
| DVMLR | Number of minor loops per computation cycle |
| DVMLT | Execution time for next minor loop |
| DVM05 | Rate limit for ladders in Minor Loop |
| DVM06 | Magnitude limit for ladders in Minor Loop |
| DVP | Flight phase indicator |
| DVPTG | Previous periodic processor execution time |
| DVRC | Accelerometer reasonableness test constant |
| DVRE | Gimbal failure count |
| DVRTC | Real time clock at last time update |
| DVSST | Switch selector execution time |
| DVTAS | Mission time at major computer cycle accelerometer read in seconds |

Table A-8
(continued)

DATA GLOSSARY
(continued)

| <u>Data Name</u> | <u>Description</u> |
|------------------|--|
| FSSAC | Switch selector processing in progress flag |
| FSSIO | Flag for issuing or bypassing SS I/O |
| FTADV | Normal or class 4 table advance flag |
| FTGOP | Time base 6 second opportunity flag |
| GS | Gravity acceleration in plumbline system |
| GST1M | Timer 1 function to be scheduled |
| GT | Terminal gravity acceleration magnitude |
| GV | Gravity acceleration in injection plane system |
| GVSTAR | Estimated average gravity acceleration for remaining boost flight path |
| GVT | Terminal gravity acceleration vector |
| J1 | IGM intermediate parameter |
| J12 | IGM intermediate parameter |
| J2 | IGM intermediate parameter |
| J3 | IGM intermediate parameter |
| J3P | IGM intermediate parameter |
| KCCT4 | Nominal computation cycle length during first S4B CHI bar steering |
| KCCT8 | Nominal computation cycle length during second S4B CHI bar steering |
| KMU | Gravitational constant |
| KT | Cosine (THETAT)/RT |
| K1 | Coefficient of IGM steering equation |
| K2 | Coefficient of IGM steering equation |
| K3 | Coefficient of IGM steering equation |
| K4 | Coefficient of IGM steering equation |
| LYP | IGM intermediate parameter |
| L1 | IGM intermediate parameter |
| L12 | IGM intermediate parameter |
| L2 | IGM intermediate parameter |
| L3 | IGM intermediate parameter |
| L3P | IGM intermediate parameter |
| MS4 | Rotation matrix from S-system to 4-system |
| M4V | Rotation matrix from 4-system to V-system |

Table A-8
(continued)

DATA GLOSSARY

(continued)

| <u>Data Name</u> | <u>Description</u> |
|------------------|---|
| DVTB | Time in time base at major computer cycle accelerometer read in seconds |
| DVTEX | Real time clock reading at last interrupt |
| DVTGB | Accumulated ground bias time update |
| DVTH | Total gimbal angle |
| DVTI | Time from GRR that the current time base was set |
| DVTMM | Elapsed time in mission from GRR at last time update |
| DVTMR | Mission time at start of reference |
| DVTRB | Elapsed time in current time base including ground bias time updates |
| DVTRR | Elapsed total time in current reference |
| DVTRS | Real time clock recording at start of reference |
| DVVSQ | Sum of the squares of X, Y and Z accelerometer changes |
| DVIMR | Computation cycles per minor loop |
| DV2TG | Time for next timer 2 function |
| EPSILON2 | Time to begin CHI bar steering for first S4B burn |
| EPSILON3 | Time to stop calculating terminal conditions |
| EPTINDX | Events processor table index |
| EPTPTR | Pointer to task for processing an event |
| EPTTBINDX | Table of index values at beginning of time bases |
| EPTTIM | Time of execution for an event |
| FASE | Alternate sequence in progress flag |
| FBRNI | S4B first/second burn flag |
| FBUG | Backup gimbal active, alter RTC |
| FBUGS | RTC conditions flag for gimbals |
| FCLS4 | Class 4 SS sequence in progress flag |
| FDSEN | Mode or data acceptable flag |
| FDSPG | DCS function in progress flag |
| FDSRE | DCS function termination required |
| FFBCH | Switch selector feedback channel flag |
| FGNC | G and C steering in progress flag |
| FHST | Hung stage test flag |

Table A-8
(continued)

DATA GLOSSARY
(continued)

| <u>Data Name</u> | <u>Description</u> |
|------------------|--|
| PHASE | IGM first/second burn indicator |
| PHII | Range angle traveled since liftoff |
| PHIIT | Predicted range angle-to-go |
| PHIT | Predicted terminal range angle |
| P1 | IGM intermediate parameter |
| P12 | IGM intermediate parameter |
| P2 | IGM intermediate parameter |
| PPSTAT | Periodic processor task status table |
| Q1 | IGM intermediate parameter |
| Q12 | IGM intermediate parameter |
| Q2 | IGM intermediate parameter |
| R | Position magnitude |
| REITERATE | Alteration flag |
| ROVEX3 | Biased reciprocal of third phase IGM exhaust velocity |
| RS | Position in plumbline coordinate system |
| RT | Terminal radius magnitude |
| RV | Position in injection plane coordinate system |
| RVT | Terminal position vector |
| R4 | Position in 4-system |
| SINTHETA | Sine of angle between pseudo-nodal vector and descending node |
| SMCFLAG | Steering misalignment corrections flag |
| SSTTBPTR | Table of pointers to switch selector table for each time base |
| SST1PTR | Normal switch selector table pointer |
| SST2PTR | Class 4 switch selector table pointer |
| S1 | IGM intermediate parameter |
| S12 | IGM intermediate parameter |
| S2 | IGM intermediate parameter |
| S4BURN | S4B first/second burn flag |
| TAU1 | First phase IGM ideal burn time |
| TAU2 | Second or fourth phase IGM ideal burn time |
| TAU3 | Third or fifth phase IGM ideal burn time |

Table A-8
(continued)

DATA GLOSSARY
(continued)

| <u>Data Name</u> | <u>Description</u> |
|------------------|--|
| TCI | Time remaining in S4B coast |
| THETAT | Desired terminal path angle |
| TSTAR | Predicted IGM total time-to-go |
| T1C | Time-to-go to IGM initiation in third phase |
| T1I | First phase IGM time-to-go |
| T2I | Second or fourth phase IGM time-to-go |
| T2STAT | Timer 2 task status table |
| T3I | Third or fifth phase IGM time-to-go |
| U1 | IGM intermediate parameter |
| U12 | IGM intermediate parameter |
| U2 | IGM intermediate parameter |
| V | Velocity magnitude |
| VASPI | Alternate sequence in progress status word |
| VATRR | Alternate SS sequence time start |
| VATR4 | Class 4 SS sequence time start |
| VBUB | Gimbal backup bias error |
| VCCYA | Previous pitch command CHI |
| VCCZA | Previous yaw command CHI |
| VCG | High order gimbal, coarse or backup resolution |
| VCG0 | Gimbal reasonableness rate limit constant (backup 2nd pass) |
| VCG1 | Gimbal reasonableness rate limit constant (backup 2nd pass) (crossover) |
| VCG10 | First pass gimbal reasonableness test constant |
| VCG11 | First pass gimbal reasonableness test constant |
| VCMND | Present attitude command |
| VCMND1 | Previous attitude command |
| VCMND2 | Actual attitude command |
| VCOD | Platform gimbal used to compute attitude |
| VDEL | Difference between actual and commanded attitude |
| VDSBL | Storage table for input data |

Table A-8
(continued)

DATA GLOSSARY
(continued)

| <u>Data Name</u> | <u>Description</u> |
|------------------|---|
| VDSEK | Temporary storage for error telemetry |
| VDSRC | Error count |
| VDSSB | Sequence bit indicator |
| VDS01 | Temporary storage for error input data |
| VEX1 | First phase IGM exhaust velocity |
| VEX2 | Second or fourth phase IGM exhaust velocity |
| VEX3 | Third or fifth phase IGM exhaust velocity |
| VF10 | I/O flag for fine or backup gimbals |
| VGBIA | Time base bias to be implemented |
| VGR | Gimbal angle reading |
| VHSTW | Previous stage and address |
| VIRE | Resolver failure limit |
| VMEMR | Temporary storage for EMR reading |
| VMLET | Temporary storage for error telemetry |
| VML0 | Gimbal reasonableness test limits |
| VML1 | Gimbal reasonableness test limits (crossover) |
| VML2 | Low order gimbal resolution |
| VOAC | Previous accelerometer readings |
| VOACT | Mission time (in RTC units) at accelerometer read |
| VOLD | Previous platform gimbal |
| VPOV | Previous measured velocity |
| VPPOT | Periodic processor current mission time |
| VPSTG | Powered stage indicators |
| VS | Velocity in plumblines coordinate system |
| VSCCA | Complimented address |
| VSC1 | Class 1 temporary storage for SST1PTR, VASPI, VATRR |
| VSC3 | Class 3 temporary storage for SST1PTR, VASPI, VATRR |
| VSF | Conversion factor for gimbal angles |
| VSNA | Stage and address word |
| VSNA1 | Stage and address in true form |
| VSSCA | Computed feedback |
| VSSFB | Switch selector feedback in error |

Table A-8
(continued)

DATA GLOSSARY
(continued)

| <u>Data Name</u> | <u>Description</u> |
|------------------|--|
| VSSRT | Switch selector time of issuance |
| VSSTM | Temporary SS time storage |
| VSSW | Bias time |
| VSTG | Powered stage temporary storage |
| VSTGO | Time-to-go to next SS function |
| VT | Terminal velocity magnitude |
| VTD | Elapsed time into launch window |
| VTOLD | Events processor previous event time |
| VV | Velocity in injection plane coordinate system |
| VVT | Terminal velocity vector |
| V0CK | Gimbal angle zero test constant |
| V4 | Velocity in 4-system |

Table A-8
(continued)

APPENDIX B

FLIGHT PROGRAM KERNEL CODING

Each of the four major paragraphs of this Appendix contains all of the coding for one language. The following table indicates on which page the coding of a given kernel (table row) in a given language (table column) begins.

Descriptions and flowcharts of these kernels can be found in Appendix A. The blank entries in the table indicate kernels which were not coded in CMS-2.

| Kernels | Languages | | | |
|-------------------------------|-----------|-------|-----|-------|
| | SPL | CLASP | HAL | CMS-2 |
| Common Data Pool | 83 | 129 | 175 | 229 |
| 1. Initialization | 92 | 135 | 181 | |
| 2. Interrupt Processor | 94 | 137 | 184 | |
| 3. Non-Interrupt Sequencer | 98 | 141 | 188 | |
| 4. Periodic Processor | 100 | 142 | 190 | |
| 5. Events Processor | 101 | 143 | 191 | |
| 6. Iterative Guidance Mode | 103 | 146 | 194 | 234 |
| 7. Digital Command System | 107 | 150 | 199 | 238 |
| 8. Accelerometer Processing | 110 | 154 | 204 | |
| 9. Minor Loop | 113 | 157 | 207 | 244 |
| 10. Switch Selector Processor | 117 | 162 | 213 | 249 |
| 11. ATM Task Keying | 126 | 172 | 226 | 258 |

(BLANK)

SPL COMMON DATA DECLARATIONS

```

START .COMPOOL 'SPL COMMON DATA AND UTILITY ROUTINES'
DECLARE CONTEXTUAL,
    TEMP,
    TEMP1
DECLARE FLOATING R,
    DKT1      ,
    DVDT      ,
    DVEOF     ,
    DVFMC     ,
    DVFOM     ,
    DVFOR     ,
    DVMAS     ,
    DVMFR     ,
    DVMLR     ,
    DVTAS     27,
    DVTB      ,
    DVTI      ,
    DVVSQ     ,
    DV1MR
DECLARE ARRAY (3) FLOATING R,
    DVD       ,
    DVDM      ,
    DVFM      ,
    DVGM      ,
    DVRC
DECLARE FIXED,
    DKMIR      0    CONSTANT = 162.53968 , '40 MILLI-SEC'
    DKTD       0    CONSTANT = 13.676,
    DVACT      0,
    DVA1       4,
    DVA2       4,
    DVA3       4,
    DVA4       4,
    DVA5       4,
    DVA6       4,
    DVERT      0,
    DVMLD      0,
    DVMLT     -2,
    DVM05      0,
    DVM06      0,
    DVPTG     -2,
    DVRTC      0,
    DVSST     -2,
    DVTD       0,
    DVTEX      0,
    DVTGB     -2,
    DVTMM     -2,
    DVTMR     -2,
    DVTRB     -2,
    DVTRR     -2,
    DVTRS     -2,
    DVTT1      0,
    DV2TG     -2
DECLARE ARRAY (3) FIXED,
    DLPRL     -2  CONSTANT =(203174. 243809. 406349.),

```

SPL COMMON DATA DECLARATIONS

```

DLPTL      -2 ,
DLTTL (12) -2 ,
DVCA       25 R,
DVCC       25 R,
DVDA       7 ,
DVDB       7 ,
DVDC       25 R,
DVTH       25 R
DECLARE INTEGER,
  DTBID     ,
  DVDGS     ,
  DVHDA     ,
  DVHDB     ,
  DVLRC     ,
  DVP
ARRAY DVRE (3) INTEGER
DECLARE STATUS,
  DFACQ     (LOSS,GAIN) ,
  DFDRF     (GOOD,FAILED) ,
  DFDTL     (INPROG,NOTINPROG) ,
  DFLT      (FLIGHT,SIM,REP) ,
  DFPHC     (NOTCHANGE,CHANGE) ,
  DFSMC     (ENABLE,DISABLE) ,
  DFTBCEP   (CHANGE,NOCHANGE) ,
  DFTUP     (NO,YES) ,
  DFWV      (CLOSE,OPEN) ,
  DFZER     (ENABLE,DISABLE) ,
  DGMMLM    (MLFS,MLNORM) ,
  DGSSM(MLFS,MLNORM,SS05,SS30,SS40,SS50,SS55,SS60,SS70,SS80),
  DGST2     (T2S,UM00,LR10,EP00,TT10,NU00,EE00,CM00,CM10,
             CM20,EPWM,ER00),
  DGST2     (T2S,UM00,LR10,EP00,TT10,NU00,EE00,CM00,CM10,
             CM20,EPWM,ER00),
  GST1M(MLFS,MLNORM,SS05,SS30,SS40,SS50,SS55,SS60,SS70,SS80)
DECLARE STATUS (IN,OUT), ARSTAT ,
  SASTAT ,
  APSTAT ,
  DVSTAT ,
  DPSTAT ,
  NESTAT ,
  TCSTAT ,
  PASTAT ,
  TTSTAT ,
  IGSTAT ,
  HSSTAT ,
  UGSTAT ,
  TGSTAT ,
  RSSTAT ,
  CSSTAT ,
  MSSTAT ,
  PGSTAT ,
  EBSTAT ,
  CC1STAT ,
  TB1STAT ,
  TB57STAT.

```

SPL COMMON DATA DECLARATIONS

```

                                CTSTAT ,
                                DTSTAT
ARRAY DKAPI (4) STATUS (ACTIVE,INACTIVE)
DECLARE ARRAY STATUS (IN,OUT),
    PPSTAT    (3),
    T2STAT    (12)
DECLARE LOGICAL,
    DFILE     ,
    DFIL1     ,
    DFIL2     ,
    DFIL3     ,
    DVASW     ,
    DVDPM     ,
    DVEMR     ,
    DVICR     ,
    DVIH     ,
    DVLDB     ,
    DVMC4     ,
    DVMC5     ,
    DVMC6     ,
    DVMC7
ARRAY DVAC (3) LOGICAL
DECLARE LOGICAL CONSTANT,
    MSKABSLADDER      =OCT'000001000',
    MSKACCELA         =OCT'777700000',
    MSKACCELB         =OCT'000017776',
    MSKDCSCOMP        =OCT'774000000',
    MSKDCSDD          =OCT'000040000',
    MSKDCSER          =OCT'000077776',
    MSKDCSMC          =OCT'770000000',
    MSKDCSMODE        =OCT'000000020',
    MSKDCSSB          =OCT'004000000',
    MSKDCSTERM        =OCT'200000000',
    MSKDIN9           =OCT'000004000',
    MSKEMRLADB        =OCT'000001000',
    MSKERRORTAG       =OCT'000070000',
    MSKFMDREP         =OCT'000100000',
    MSKFPSCORD        =OCT'100000000',
    MSKFPSINT2        =OCT'000040000',
    MSKFPSISSA        =OCT'001000000',
    MSKGIMBALA        =OCT'377700000',
    MSKICRBG          =OCT'000000020',
    MSKICRCA          =OCT'000040000',
    MSKICRSSCB        =OCT'000010000',
    MSKICRSWG         =OCT'000002000',
    MSKINT             =OCT'157740000',
    MSKMOD180DEG      =OCT'377777776',
    MSKRTC             =OCT'000037776',
    MSKRTCRESET       =OCT'007777540',
    MSKSCCO           =OCT'000100000',
    MSKSSCLS1         =OCT'000003770',
    MSKSSCLS3         =OCT'077774000',
    MSKSSDCS          =OCT'500000000',
    MSKSSDCT          =OCT'405400000',
    MSKSSHIG          =OCT'100720000',

```


SPL COMMON DATA DECLARATIONS

| | | | |
|-------|---------------|------------------|-----------------|
| | MSKSSHS | =OCT'003770000' | |
| | MSKSSLOG | =OCT'100520000' | |
| | MSKSSNSEND | =OCT'377777776' | |
| | MSKSSOMG | =OCT'100070000' | |
| | MSKSSREAD | =OCT'400000000' | |
| | MSKSSRESET | =OCT'200000000' | |
| | MSKSSSCC | =OCT'100310000' | |
| | MSKSSSIVR | =OCT'020230000' | |
| | MSKSSSNA | =OCT'135770000' | |
| | MSKSSSSR | =OCT'174000000' | |
| | MSKSSWV | =OCT'003000000' | |
| | MSKSSWVC | =OCT'101050000' | |
| | MSKSSWVO | =OCT'101450000' | |
| | MSKSSZFSF | =OCT'002000000' | |
| | MSKTMCO | =OCT'700000000' | |
| | MSKTMCI | =OCT'710000000' | |
| | MSKTMCI2 | =OCT'720000000' | |
| | MSKTMCI3 | =OCT'730000000' | |
| | MSKTMCI4 | =OCT'740000000' | |
| | MSKT2INT | =OCT'100000000' | |
| | MSK18UDEG | =OCT'400000000' | |
| MC24. | DECLARE, MSK0 | LOGICAL CONSTANT | =OCT'400000000' |
| | MSK1 | LOGICAL CONSTANT | =OCT'200000000' |
| | MSK2 | LOGICAL CONSTANT | =OCT'100000000' |
| | MSK3 | LOGICAL CONSTANT | =OCT'040000000' |
| | MSK4 | LOGICAL CONSTANT | =OCT'020000000' |
| | MSK5 | LOGICAL CONSTANT | =OCT'010000000' |
| | MSK6 | LOGICAL CONSTANT | =OCT'004000000' |
| | MSK7 | LOGICAL CONSTANT | =OCT'002000000' |
| | MSK8 | LOGICAL CONSTANT | =OCT'001000000' |
| | MSK9 | LOGICAL CONSTANT | =OCT'000400000' |
| | MSK10 | LOGICAL CONSTANT | =OCT'000200000' |
| | MSK11 | LOGICAL CONSTANT | =OCT'000100000' |
| | MSK12 | LOGICAL CONSTANT | =OCT'000040000' |
| | MSK13 | LOGICAL CONSTANT | =OCT'000020000' |
| | MSK14 | LOGICAL CONSTANT | =OCT'000010000' |
| | MSK15 | LOGICAL CONSTANT | =OCT'000004000' |
| | MSK16 | LOGICAL CONSTANT | =OCT'000002000' |
| | MSKMC4SSCR | LOGICAL CONSTANT | =OCT'000001000' |
| | MSK18 | LOGICAL CONSTANT | =OCT'000000400' |
| | MSK19 | LOGICAL CONSTANT | =OCT'000000200' |
| | MSKMC4AMF | LOGICAL CONSTANT | =OCT'000000100' |
| | MSK21 | LOGICAL CONSTANT | =OCT'000000040' |
| | MSK22 | LOGICAL CONSTANT | =OCT'000000020' |
| | MSK23 | LOGICAL CONSTANT | =OCT'000000010' |
| | MSK24 | LOGICAL CONSTANT | =OCT'000000004' |
| | MSK25 | LOGICAL CONSTANT | =OCT'000000002' |
| MC25. | DECLARE, MSK0 | LOGICAL CONSTANT | =OCT'400000000' |
| | MSK1 | LOGICAL CONSTANT | =OCT'200000000' |
| | MSK2 | LOGICAL CONSTANT | =OCT'100000000' |
| | MSK3 | LOGICAL CONSTANT | =OCT'040000000' |
| | MSK4 | LOGICAL CONSTANT | =OCT'020000000' |
| | MSK5 | LOGICAL CONSTANT | =OCT'010000000' |
| | MSK6 | LOGICAL CONSTANT | =OCT'004000000' |
| | MSK7 | LOGICAL CONSTANT | =OCT'002000000' |

SPL COMMON DATA DECLARATIONS

| | | | | |
|-------|----------|------------|------------------|------------------|
| | | MSK8 | LOGICAL CONSTANT | #OCT'001000000', |
| | | MSK9 | LOGICAL CONSTANT | #OCT'000400000', |
| | | MSK10 | LOGICAL CONSTANT | #OCT'000200000', |
| | | MSK11 | LOGICAL CONSTANT | #OCT'000100000', |
| | | MSK12 | LOGICAL CONSTANT | #OCT'000040000', |
| | | MSK13 | LOGICAL CONSTANT | #OCT'000020000', |
| | | MSK14 | LOGICAL CONSTANT | #OCT'000010000', |
| | | MSK15 | LOGICAL CONSTANT | #OCT'000004000', |
| | | MSK16 | LOGICAL CONSTANT | #OCT'000002000', |
| | | MSK17 | LOGICAL CONSTANT | #OCT'000001000', |
| | | MSK18 | LOGICAL CONSTANT | #OCT'000000400', |
| | | MSK19 | LOGICAL CONSTANT | #OCT'000000200', |
| | | MSKMC5481I | LOGICAL CONSTANT | #OCT'000000100', |
| | | MSK21 | LOGICAL CONSTANT | #OCT'000000040', |
| | | MSK22 | LOGICAL CONSTANT | #OCT'000000020', |
| | | MSK23 | LOGICAL CONSTANT | #OCT'000000010', |
| | | MSK24 | LOGICAL CONSTANT | #OCT'000000004', |
| | | MSK25 | LOGICAL CONSTANT | #OCT'000000002' |
| MC26. | DECLARE, | MSKMC688RI | LOGICAL CONSTANT | #OCT'400000000', |
| | | MSK1 | LOGICAL CONSTANT | #OCT'200000000', |
| | | MSK2 | LOGICAL CONSTANT | #OCT'100000000', |
| | | MSK3 | LOGICAL CONSTANT | #OCT'040000000', |
| | | MSK4 | LOGICAL CONSTANT | #OCT'020000000', |
| | | MSK5 | LOGICAL CONSTANT | #OCT'010000000', |
| | | MSK6 | LOGICAL CONSTANT | #OCT'004000000', |
| | | MSK7 | LOGICAL CONSTANT | #OCT'002000000', |
| | | MSK8 | LOGICAL CONSTANT | #OCT'001000000', |
| | | MSK9 | LOGICAL CONSTANT | #OCT'000400000', |
| | | MSK10 | LOGICAL CONSTANT | #OCT'000200000', |
| | | MSK11 | LOGICAL CONSTANT | #OCT'000100000', |
| | | MSK12 | LOGICAL CONSTANT | #OCT'000040000', |
| | | MSK13 | LOGICAL CONSTANT | #OCT'000020000', |
| | | MSKMC6LUI | LOGICAL CONSTANT | #OCT'000010000', |
| | | MSK15 | LOGICAL CONSTANT | #OCT'000004000', |
| | | MSKMC6T86A | LOGICAL CONSTANT | #OCT'000002000', |
| | | MSK17 | LOGICAL CONSTANT | #OCT'000001000', |
| | | MSKMC6D04 | LOGICAL CONSTANT | #OCT'000000400', |
| | | MSK19 | LOGICAL CONSTANT | #OCT'000000200', |
| | | MSK20 | LOGICAL CONSTANT | #OCT'000000100', |
| | | MSKMC6T86C | LOGICAL CONSTANT | #OCT'000000040', |
| | | MSK22 | LOGICAL CONSTANT | #OCT'000000020', |
| | | MSKMC6T86B | LOGICAL CONSTANT | #OCT'000000010', |
| | | MSK24 | LOGICAL CONSTANT | #OCT'000000004', |
| | | MSK25 | LOGICAL CONSTANT | #OCT'000000002' |
| MC27. | DECLARE, | MSK0 | LOGICAL CONSTANT | #OCT'400000000', |
| | | MSK1 | LOGICAL CONSTANT | #OCT'200000000', |
| | | MSKMC7T6D | LOGICAL CONSTANT | #OCT'100000000', |
| | | MSK3 | LOGICAL CONSTANT | #OCT'040000000', |
| | | MSKMC70M8 | LOGICAL CONSTANT | #OCT'020000000', |
| | | MSKMC7L08 | LOGICAL CONSTANT | #OCT'010000000', |
| | | MSKMC7HIG | LOGICAL CONSTANT | #OCT'004000000', |
| | | MSK7 | LOGICAL CONSTANT | #OCT'002000000', |
| | | MSK8 | LOGICAL CONSTANT | #OCT'001000000', |
| | | MSK9 | LOGICAL CONSTANT | #OCT'000400000', |
| | | MSK10 | LOGICAL CONSTANT | #OCT'000200000', |

SPL COMMON DATA DECLARATIONS

```

MSK11          LOGICAL CONSTANT  =OCT'000100000'
MSK12          LOGICAL CONSTANT  =OCT'000040000'
MSK13          LOGICAL CONSTANT  =OCT'000020000'
MSK14          LOGICAL CONSTANT  =OCT'000010000'
MSK15          LOGICAL CONSTANT  =OCT'000004000'
MSK16          LOGICAL CONSTANT  =OCT'000002000'
MSK17          LOGICAL CONSTANT  =OCT'000001000'
MSK18          LOGICAL CONSTANT  =OCT'000000400'
MSK19          LOGICAL CONSTANT  =OCT'000000200'
MSK20          LOGICAL CONSTANT  =OCT'000000100'
MSK21          LOGICAL CONSTANT  =OCT'000000040'
MSK22          LOGICAL CONSTANT  =OCT'000000020'
MSK23          LOGICAL CONSTANT  =OCT'000000010'
MSK24          LOGICAL CONSTANT  =OCT'000000004'
MSK25          LOGICAL CONSTANT  =OCT'000000002'
SSDVASW.  DECLARE, MSKSSS4C0    LOGICAL CONSTANT  =OCT'400000000'
MSKSSSPFC    LOGICAL CONSTANT  =OCT'200000000'
MSKSSSTH6C   LOGICAL CONSTANT  =OCT'100000000'
MSKSSGNSS    LOGICAL CONSTANT  =OCT'040000000'
MSKSSSHLO    LOGICAL CONSTANT  =OCT'020000000'
MSKSSSHHI    LOGICAL CONSTANT  =OCT'010000000'
MSKSSSHQM    LOGICAL CONSTANT  =OCT'004000000'
MSKSSSECSV   LOGICAL CONSTANT  =OCT'002000000'
MSKSSSECS1   LOGICAL CONSTANT  =OCT'001000000'
MSKSSST3A    LOGICAL CONSTANT  =OCT'000400000'
MSKSSSTR6D   LOGICAL CONSTANT  =OCT'000200000'
MSK11          LOGICAL CONSTANT  =OCT'000100000'
MSK12          LOGICAL CONSTANT  =OCT'000040000'
MSK13          LOGICAL CONSTANT  =OCT'000020000'
MSK14          LOGICAL CONSTANT  =OCT'000010000'
MSK15          LOGICAL CONSTANT  =OCT'000004000'
MSKSSSTH6A   LOGICAL CONSTANT  =OCT'000002000'
MSKSSSTR6R   LOGICAL CONSTANT  =OCT'000001000'
MSKSSS4C1    LOGICAL CONSTANT  =OCT'000000400'
MSK19          LOGICAL CONSTANT  =OCT'000000200'
MSK20          LOGICAL CONSTANT  =OCT'000000100'
MSK21          LOGICAL CONSTANT  =OCT'000000040'
MSK22          LOGICAL CONSTANT  =OCT'000000020'
MSK23          LOGICAL CONSTANT  =OCT'000000010'
MSKSSACQU    LOGICAL CONSTANT  =OCT'000000004'
MSKSSLI      LOGICAL CONSTANT  =OCT'000000002'
SSVASPI.  DECLARE, MSKSSS4C0    LOGICAL CONSTANT  =OCT'400000000'
MSKSSSPEC    LOGICAL CONSTANT  =OCT'200000000'
MSKSSCL3     LOGICAL CONSTANT  =OCT'100000000'
MSKSSCL1     LOGICAL CONSTANT  =OCT'040000000'
MSK4          LOGICAL CONSTANT  =OCT'020000000'
MSK5          LOGICAL CONSTANT  =OCT'010000000'
MSKSSST6C    LOGICAL CONSTANT  =OCT'004000000'
MSK7          LOGICAL CONSTANT  =OCT'002000000'
MSK8          LOGICAL CONSTANT  =OCT'001000000'
MSK9          LOGICAL CONSTANT  =OCT'000400000'
MSK10         LOGICAL CONSTANT  =OCT'000200000'
MSK11         LOGICAL CONSTANT  =OCT'000100000'
MSK12         LOGICAL CONSTANT  =OCT'000040000'
MSK13         LOGICAL CONSTANT  =OCT'000020000'

```

SPL COMMON DATA DECLARATIONS

```

MSK14          LOGICAL CONSTANT  =OCT'000010000',
MSK15          LOGICAL CONSTANT  =OCT'000004000',
MSK16          LOGICAL CONSTANT  =OCT'000002000',
MSK17          LOGICAL CONSTANT  =OCT'000001000',
MSK18          LOGICAL CONSTANT  =OCT'000000400',
MSK19          LOGICAL CONSTANT  =OCT'000000200',
MSK20          LOGICAL CONSTANT  =OCT'000000100',
MSK21          LOGICAL CONSTANT  =OCT'000000040',
MSK22          LOGICAL CONSTANT  =OCT'000000020',
MSK23          LOGICAL CONSTANT  =OCT'000000010',
MSK24          LOGICAL CONSTANT  =OCT'000000004',
MSK25          LOGICAL CONSTANT  =OCT'000000002',
OVERLAY MC24 = MC25 = MC26 = MC27 = SSDVASW = SSVASPI
DECLARE FILE CLOCK      DEVICE = TIMER      ,
        FILE  DBG       DEVICE = DOMBUGIM   ,
        FILE  DCS       DEVICE = DCSINREG   ,
        FILE  DIR       DEVICE = DISINREG   ,
        FILE  DOM       DEVICE = SSDOM      ,
        FILE  DOR       DEVICE = DISOUTRES  ,
        FILE  DOS       DEVICE = DISOUTSET  ,
        FILE  EMR       DEVICE = ERRMONREG  ,
        FILE  ICR       DEVICE = INTCONREG  ,
        FILE  MODREG    DEVICE = MODEREG    ,
        FILE  SS        DEVICE = SSREG      ,
        FILE  SSFB      DEVICE = SSFDBK     ,
        FILE  TELDCSDW  DEVICE = PI0574     ,
        FILE  TELDCSEC  DEVICE = PI0055     ,
        FILE  TELDCSSC  DEVICE = PI0030     ,
        FILE  TELGT     DEVICE = PI0574     ,
        FILE  TELMLER   DEVICE = PI0570     ,
        FILE  TELPHIT   DEVICE = PI0414     ,
        FILE  TELRTC    DEVICE = PI0174     ,
        FILE  TELSSFB   DEVICE = PI0500     ,
        FILE  TELSSSA   DEVICE = PI0075     ,
        FILE  TELTAS    DEVICE = PI0000     ,
        FILE  TELTB     DEVICE = PI0031     ,
        FILE  TELTI     DEVICE = PI0561     ,
        FILE  TELT3I    DEVICE = PI0464     ,
        FILE  TELXAC    DEVICE = PI0010     ,
        FILE  TELXDM    DEVICE = PI0024     ,
        FILE  TELX4     DEVICE = PI0444     ,
        FILE  TELYAC    DEVICE = PI0014     ,
        FILE  TELYDM    DEVICE = PI0030     ,
        FILE  TELYD4    DEVICE = PI0450     ,
        FILE  TELY4     DEVICE = PI0450     ,
        FILE  TELZAC    DEVICE = PI0004     ,
        FILE  TELZDM    DEVICE = PI0020     ,
        FILE  TELZ4     DEVICE = PI0434     ,
        FILE  TIM1      DEVICE = TIMER1    ,
        FILE  TIM2      DEVICE = TIMER2    ,
        FILE  XACC      DEVICE = XACCEL    ,
        FILE  XBGIM     DEVICE = XRACKUP   ,
        FILE  XGIM      DEVICE = XGIMRAL   ,
        FILE  XLAD      DEVICE = XLADDER   ,
        FILE  YACC      DEVICE = YACCEL    ,

```

SPL COMMON DATA DECLARATIONS

| | | | | |
|------|-------|----------|---------|---|
| FILE | YBGIM | DEVICE = | YBACKUP | , |
| FILE | YGIM | DEVICE = | YGIMBAL | , |
| FILE | YLAD | DEVICE = | YLADDER | , |
| FILE | ZACC | DEVICE = | ZACCEL | , |
| FILE | ZBGIM | DEVICE = | ZBACKUP | , |
| FILE | ZGIM | DEVICE = | ZGIMRAL | , |
| FILE | ZLAD | DEVICE = | ZLADDER | , |

SPL UTILITY ROUTINES

```

PROC      .UTR00      ''TELEMETRY DELAY FOR MODE REG SETTING OF 70''
ENTRANCE  .UTR01      ''TELEMETRY DELAY FOR MODE REG SETTING OF 71''
ENTRANCE  .UTR02      ''TELEMETRY DELAY FOR MODE REG SETTING OF 72''
ENTRANCE  .UTR03      ''TELEMETRY DELAY FOR MODE REG SETTING OF 73''
ENTRANCE  .UTR04      ''TELEMETRY DELAY FOR MODE REG SETTING OF 74''
          ITEM KTELBIAS FIXED 0 CONSTANT = 2.
          ITEM VTIM FIXED 0
          ITEM VTMC LOGICAL

FNDDATA
          VTMC = MSKTMCO
          GOTO TR00
UTR01.    .UTR01      ''TELEMETRY DELAY FOR MODE REG SETTING OF 71''
          VTMC = MSKTMCO1
          GOTO TR00
UTR02.    .UTR02      ''TELEMETRY DELAY FOR MODE REG SETTING OF 72''
          VTMC = MSKTMCO2
          GOTO TR00
UTR03.    .UTR03      ''TELEMETRY DELAY FOR MODE REG SETTING OF 73''
          VTMC = MSKTMCO3
          GOTO TR00
UTR04.    .UTR04      ''TELEMETRY DELAY FOR MODE REG SETTING OF 74''
          VTMC = MSKTMCO4
TR00 .    LOCK T1INT,T2INT,EX1INT,EX2INT,EX3INT,EX4INT,EX5INT,EX6INT,
          EX7INT,EX8INT,EX9INT
          READ CLOCK,VTIM
          IF VTIM - DVTD LAND MSKRTC GQ DKTD          GOTO TR05
          UNLOCK ''RELEASE PREVIOUSLY ENABLED INTERRUPTS''
          ''ALLOW HIGH PRIORITY TASKS TO INTERRUPT''
          GOTO TR00
TR05 .    WRITE MODREG,VTMC
          DVTD = VTIM + KTELBIAS
EXIT      ''COMMON TELEMETRY DELAY RETURN''
PROC      .UTR30      ''TELEMETRY DELAY FOR INTERRUPT LEVEL 3''
          ITEM KTELBIAS FIXED 0 CONSTANT = 2.
          ITEM VTIM FIXED 0

FNDDATA
TR35.    READ CLOCK,VTIM
          IF VTIM - DVTD LAND MSKRTC LS DKTD          GOTO TR35
          WRITE MODREG,MSKTMCO
          DVTD = VTIM + KTELBIAS
EXIT      ''UTR30''
PROC      .UTR24      ''TELEMETRY DELAY FOR INTERRUPT LEVEL 2''
          ITEM KTELBIAS FIXED 0 CONSTANT = 2.
          ITEM VTIM FIXED 0

FNDDATA
TR20 .    LOCK T1INT,T2INT,EX1INT,EX2INT,EX3INT,EX4INT,EX5INT,EX6INT,
          EX7INT,EX8INT,EX9INT
          READ CLOCK,VTIM
          IF VTIM - DVTD LAND MSKRTC GQ DKTD          GOTO TR25
          UNLOCK T1INT
          GOTO TR20
TR25 .    WRITE MODREG,MSKTMCO4
          DVTD = VTIM + KTELBIAS
EXIT      ''UTR24''
TERM

```

SPL KERNEL 1 INITIALIZATION

```

START      .EGP0      ''MISSION INITIALIZATION''
ENTRANCE   .MPA00     ''PHASE TERMINATION''
ITEM VTD FLOATING
ITEM FGNC STATUS (INACTIVE,ACTIVE)

ENDDATA

LOCK TLCINT,T1INT,T2INT,EX1INT,EX2INT,EX3INT,EX4INT,EX5INT,
        EX6INT,EX7INT,EX8INT,EX9INT
READ XACC,VOAC(0)
READ YACC,VOAC(1)
READ ZACC,VOAC(2)
READ CLOCK,DVACT
IF DFMDI LAND MSKFMDREP NQ 0
    THEN ON T1INT
        READ CLOCK,DVACT
        LOCK T1INT
        TEMP = 0
        END
        TEMP = 1
        WRITE TIM1,TEMP
        UNLOCK T1INT
        IF TEMP EQ 1    WAIT
    END
DFIL1, DFIL2, DFIL3 = 'ACTIVE'
DVRTC,DVTEX,VPPOT = DVACT
DVTMM,DVTRR,DVERT,DVTGR,DVTRS,DVTMR,DTBID,VTD = 0.
.EGP1      ''ACTIVATE INTERRUPT PROCESSOR CHRONIC STATEMENTS''
UNLOCK TLCINT
FGNC = 'INACTIVE'
DVSST = 1.E10
DVMLT = DVMLD = DKMIR
.EGP15     ''SCHEDULE FIRST TIMER 1 FUNCTION''
DVP = 1
GP002.    IF DVP GR 4          WAIT
          IF DKAPI(DVP-1) EQ 'ACTIVE'
              THEN .EGP20     ''START PHASE TIME REFERENCE''
                  GOTO (INP13, INP24, INP13, INP24, *) DVP - 1
          ELSE DVP = DVP + 1
              GOTO GP002
          END
MPA00.    DFPHC = 'CHANGE'
          LOCK TLCINT,T1INT,T2INT,EX1INT,EX2INT,EX3INT,EX4INT,EX5INT,
              EX6INT,EX7INT,EX8INT,EX9INT
          WRITE TIM2,MSKRTC ''LOAD TIMER 2 WITH A LARGE VALUE TO PREVENT
                          TIMER 2 INTERRUPTS FROM OCCURRING''
          WRITE ISR,MSKT2INT ''RESET ANY PENDING TIMER 2 INTERRUPT''
          DFIL1, DFIL2 = 'ACTIVE'
          UNLOCK TLCINT,T1INT
          GOTO GP002
INP13.    ARSTAT = 'IN'
          SASTAT = 'OUT'
          APSTAT = 'IN'
          DVSTAT = 'OUT'
          DPSTAT = 'IN'
          NESTAT = 'IN'
          TCSTAT = 'OUT'

```

SPL KERNEL 1 INITIALIZATION

```

PASTAT = 'OUT'
TTSTAT = 'OUT'
CC1STAT = 'IN'
IGSTAT = 'OUT'
HSSTAT = 'OUT'
OGSTAT = 'OUT'
TGSTAT = 'OUT'
RSSTAT = 'OUT'
CSSTAT = 'OUT'
TB1STAT = 'OUT'
TB57STAT = 'OUT'
MSSTAT = 'IN'
PGSTAT = 'OUT'
EBSTAT = 'IN'
DLPTL = 0.
PPSTAT = 'OUT'
T2STAT = 'OUT', 'IN', 'OUT'
.MIN00    ''PERFORM PHASE 1/3 APPLIC PGM INIT    (NOT CODED)''
.EGP18    ''SCHEDULE NEXT TIMER 2 FUNCTION''
DFIL1, DFIL2, DFIL3 = 'INACTIVE'
DFPHC = 'NOTCHANGE'
UNLOCK ''UNLOCK PREVIOUSLY ENABLED INTERRUPTS''
.NONINTSEQ1 ''PASS CONTROL TO PHASE 1/3 NON-INTERRUPT SEQ''
INP24. CTSTAT = 'OUT'
DTSTAT = 'OUT'
DLPTL = 0.
PPSTAT = 'IN'
T2STAT = 'OUT', 'OUT', 'IN', 'IN', 'IN', 'OUT', 'OUT', 'IN'
.MIN10    ''PERFORM PHASE 2/4 APPLIC PGM INIT    (NOT CODED)''
.EGP18    ''SCHEDULE NEXT TIMER 2 FUNCTION''
DFIL1, DFIL2, DFIL3 = 'INACTIVE'
DFPHC = 'NOTCHANGE'
UNLOCK ''RELEASE PREVIOUSLY ENABLED INTERRUPTS''
.NONINTSEQ2 ''PASS CONTROL TO PHASE 2/4 NON-INTERRUPT SEQ''
PROC      .EGP1          EXTERNAL  EXIT
PROC      .EGP15         EXTERNAL  EXIT
PROC      .EGP20         EXTERNAL  EXIT
PROC      .EGP18         EXTERNAL  EXIT
PROC      .MIN00         EXTERNAL  EXIT
PROC      .MIN10         EXTERNAL  EXIT
PROC      .NONINTSEQ1    EXTERNAL  EXIT
PROC      .NONINTSEQ2    EXTERNAL  EXIT
TERM      ''EGP0''

```


SPL KERNEL 2 INTERRUPT PROCESSING

```

START      .EGP1      ''INTERRUPT PROCESSOR''
ENTRANCE   .EGP15     ''TIMER 1 SCHEDULER''
ENTRANCE   .EGP18     ''TIMER 2 SCHEDULER''
ENTRANCE   .EGP20     ''SYSTEM TIME UPDATE ROUTINE''
DECLARE FIXED CONSTANT,
            KT1BIAS  0 = 9.,
            KT2BIAS  0 = 12.,
            K4SEC    -2 = 16253.968
ENDDATA
''
''RESPONSE FOR TLC INTERRUPT
''
    ON TLCINT
    LOCK T1INT,T2INT,EX1INT,EX2INT,EX3INT,EX4INT,EX5INT,EX6INT,
        EX7INT,EX8INT,EX9INT
    READ CLOCK,DVTEX
    DFIL2,DFIL3 = 'ACTIVE'
    .MSS00      ''PROCESS TLC INTERRUPT          (NOT CODED)''
''THE TLC APPLICATION PROGRAM DOES NOT RETURN CONTROL
''
    END
''
''RESPONSE FOR TIMER 1 INTERRUPT
''
    ON T1INT
    LOCK T1INT,T2INT,EX1INT,EX2INT,EX3INT,EX4INT,EX5INT,EX6INT,
        EX7INT,EX8INT,EX9INT
    READ CLOCK,DVTT1
    DFIL3 = 'ACTIVE'
    GOTO ( ,GP11,GP12,GP13,GP14,GP15,GP16,GP17,GP18,GP19) GST1M
    .MML00      ''FLIGHT SIMULATION MINOR LOOP''      GOTO EGP11
GP11.         .MML20      ''NORMAL MINOR LOOP''      GOTO EGP11
GP12.         .MSS05      ''SWITCH SELECTOR CHECK''   GOTO EGP11
GP13.         .MSS30      ''SWITCH SELECTOR HUNG STAGE TEST'' GOTO EGP11
GP14.         .MSS40      ''SWITCH SELECTOR STAGE,ADDRESS ISSUE'' GOTO EGP11
GP15.         .MSS50      ''SWITCH SELECTOR VERIFY ADDRESS'' GOTO EGP11
GP16.         .MSS55      ''SWITCH SELECTOR READ TIME CHECK'' GOTO EGP11
GP17.         .MSS60      ''SWITCH SELECTOR READ ISSUANCE'' GOTO EGP11
GP18.         .MSS70      ''SWITCH SELECTOR RESET''   GOTO EGP11
GP19.         .MSS80      ''SWITCH SELECTOR COMPLEMENT STAGE,ADD''
EGP11.        .EGP15      ''SCHEDULE NEXT TIMER 1 FUNCTION''
    DFIL3 = 'INACTIVE'
    UNLOCK ''RELEASE PREVIOUSLY ENABLED INTERRUPTS''
    END
''
''RESPONSE FOR TIMER 2 INTERRUPT
''
    ON T2INT
    LOCK T2INT
    DFIL1 = 'ACTIVE'
    GO TO (EGP12,GP21,GP22,GP23,GP24,GP25,GP26,GP27,GP28,GP29,
        GP30,GP31) DGST2
GP21.         .MUM00      ''TIME UPDATE          (NOT CODED) '' GOTO EGP12
GP22.         .MLR10      ''LADDER RAMP PROCESSOR (NOT CODED) '' GOTO EGP12
GP23.         .MEP00      ''EVENTS PROCESSOR''      GOTO EGP12
GP24.         .MTT10      ''TIME TILT GUIDANCE    (NOT CODED) '' GOTO EGP12

```

SPL KERNEL 2 INTERRUPT PROCESSING

```

GP25. .MNU00    ''NAVIGATION UPDATE IMPL (NOT CODED) '' GOTO EGP12
GP26. .MEE00    ''TIME BASE 8 ENABLE      (NOT CODED) '' GOTO EGP12
GP27. .MCM00    ''PHASE 2/4 CONTROL MOD  (NOT CODED) '' GOTO EGP12
GP28. .MCM10    ''PHASE 2/4 CONTROL MOD  (NOT CODED) '' GOTO EGP12
GP29. .MCM20    ''PHASE 2/4 CONTROL MOD  (NOT CODED) '' GOTO EGP12
GP30. .MEPWM    ''WATER METHANOL ACTIVATE(NOT CODED) '' GOTO EGP12
GP31. .MER00    ''EXTRA ACCELEROMETER RD (NOT CODED) ''
EGP12. .EGP18    ''SCHEDULE NEXT TIMER 2 FUNCTION''
      DFIL1 = 'INACTIVE'
      UNLOCK T2INT
      END

''
''RESPONSE FOR EXTERNAL 2 INTERRUPT
''
      ON EX2INT
      LOCK T1INT,T2INT,EX1INT,EX2INT,EX3INT,EX4INT,EX5INT,EX6INT,
        EX7INT,EX8INT,EX9INT
      READ CLOCK,DVTEX    ''READ REAL TIME CLOCK''
      DFIL2 = DFIL3 = 'ACTIVE'
      .MDP28    ''SC INITIATION OF S2/S4B SEPARATION (NOT CODED)''
      DFIL2 = DFIL3 = 'INACTIVE'
      UNLOCK ''RELEASE PREVIOUSLY ENABLED INTERRUPTS''
      END

''
''RESPONSE FOR EXTERNAL 4 INTERRUPT
''
      ON EX4INT
      LOCK T1INT,T2INT,EX1INT,EX2INT,EX3INT,EX4INT,EX5INT,EX6INT,
        EX7INT,EX8INT,EX9INT
      READ CLOCK,DVTEX    ''READ REAL TIME CLOCK''
      DFIL2 = DFIL3 = 'ACTIVE'
      .MTB50    ''S4B ENGINE OUT (NOT CODED)''
      DFIL2 = DFIL3 = 'INACTIVE'
      UNLOCK ''RELEASE PREVIOUSLY ENABLED INTERRUPTS''
      END

''
''RESPONSE FOR EXTERNAL 5 INTERRUPT
''
      ON EX5INT
      LOCK T1INT,T2INT,EX1INT,EX2INT,EX3INT,EX4INT,EX5INT,EX6INT,
        EX7INT,EX8INT,EX9INT
      READ CLOCK,DVTEX    ''READ REAL TIME CLOCK''
      DFIL2 = DFIL3 = 'ACTIVE'
      .MTB30    ''SIC OUTBOARD ENGINE OUT (NOT CODED)''
      DFIL2 = DFIL3 = 'INACTIVE'
      UNLOCK ''RELEASE PREVIOUSLY ENABLED INTERRUPTS''
      END

''
''RESPONSE FOR EXTERNAL 6 INTERRUPT
''
      ON EX6INT
      LOCK T1INT,T2INT,EX1INT,EX2INT,EX3INT,EX4INT,EX5INT,EX6INT,
        EX7INT,EX8INT,EX9INT
      READ CLOCK,DVTEX    ''READ REAL TIME CLOCK''
      DFIL2 = DFIL3 = 'ACTIVE'

```

SPL KERNEL 2 INTERRUPT PROCESSING

```

.MTB40      'S2 PROPELLANT DEPLETION          (NOT CODED)''
DFIL2 = DFIL3 = 'INACTIVE'
UNLOCK 'RELEASE PREVIOUSLY ENABLED INTERRUPTS''
END
''
''RESPONSE FOR EXTERNAL 8 INTERRUPT          ''
''
ON EX8INT
LOCK T1INT,T2INT,EX1INT,EX2INT,EX3INT,EX4INT,EX5INT,EX6INT,
    EX7INT,EX8INT,EX9INT
READ CLOCK,DVTEX      ''READ REAL TIME CLOCK''
DFIL2 = DFIL3 = 'ACTIVE'
.MDS00      ''PROCESS DCS INPUT''
DFIL2 = DFIL3 = 'INACTIVE'
UNLOCK 'RELEASE PREVIOUSLY ENABLED INTERRUPTS''
END
RETURN      ''EGP1''
EGP15.      READ CLOCK,TEMP
            TEMP1 = DVTMM + (TEMP - DVRTC LAND MSKRTC) SCL 0 + KT1BIAS
            CONDITIONS
                DVMLT LQ TEMP1          ,(Y, , , )
                DVMLT LQ DVSST          ,( , Y, , )
                DVSST LQ TEMP1          ,( , , Y, N)
            ACTIONS
                TEMP = 1                  ,(Y, , Y, )
                TEMP = (DVMLT - TEMP1) LSH 1 ,( , Y, , )
                TEMP = (DVSST - TEMP1) LSH 1 ,( , , , Y)
                DGST1M = DGMLM           ,(Y, Y, , )
                GST1M = DGSSM           ,( , , Y, Y)
            ELSE GOTO EGP150 ''THIS POINT SHOULD NEVER BE REACHED LOGICALLY''
            END
            WRITE TIM1,TEMP ''LOAD TIMER 1 WITH TIME-TO-GO FOR FUNCTION''
EGP150.      RETURN      ''EGP15''
EGP18.      DGST2 = 'T2S'
            DV2TG = DVTMM + K4SEC
            FOR I = 1 BY 1 UNTIL 12
                IF T2STAT(I) EQ 'OUT'          GOTO T2S10
                IF DLTTL(I) GR DV2TG          GOTO T2S10
                DGST2 = I
                DV2TG = DLTTL(I)
T2S10.      END
            LOCK T1INT,EX1INT,EX2INT,EX3INT,EX4INT,EX5INT,EX6INT,EX7INT,
                EX8INT,EX9INT
            IF DV2TG LQ DVTMM          GOTO T2S20
            READ CLOCK,TEMP      ''READ REAL TIME CLOCK''
            TEMP = ((DV2TG - DVTMM) SCL 0 + DVERT - (TEMP - DVRTC LAND
                MSKRTC) - KT2RIAS) SCL -1
            IF TEMP LQ 0
T2S20.      TEMP = 1
            WRITE TIM2,TEMP      ''LOAD TIMER 2''
            UNLOCK 'RELEASE PREVIOUSLY ENABLED INTERRUPTS''
            RETURN      ''EGP18''
EGP20.      LOCK T1INT,T2INT,EX1INT,EX2INT,EX3INT,EX4INT,EX5INT,EX6INT,
                EX7INT,EX8INT,EX9INT
            READ CLOCK,TEMP1

```

SPL KERNEL 2 INTERRUPT PROCESSING

```

DVERT = TEMP1 - DVRTC LAND 3
DVTMM = DVTMM + (TEMP1 - DVRTC LAND MSKRTC) RSH 2
DVRTC = TEMP1 - DVERT
DVTRR = DVTMM - DVTMR
CONDITIONS
    DFIL3 EQ 'ACTIVE' , (Y, , )
    DFIL2 EQ 'ACTIVE' , ( , Y, )
    DFIL1 EQ 'ACTIVE' , ( , , Y)
ACTIONS
    UNLOCK T1INT , ( , Y, )
    UNLOCK ''RELEASE PREVIOUSLY ENABLED
        INTERRUPTS EXCEPT TIMER 2'' , ( , , Y)
    RETURN ''EXIT EGP20'' , (Y, Y, Y)
ELSE UNLOCK ''RELEASE PREVIOUSLY ENABLED INTERRUPTS''
END
RETURN ''EGP20''
PROC .MTS00 EXTERNAL EXIT
PROC .MML00 EXTERNAL EXIT
PROC .MML20 EXTERNAL EXIT
PROC .MSS05 EXTERNAL EXIT
PROC .MSS30 EXTERNAL EXIT
PROC .MSS40 EXTERNAL EXIT
PROC .MSS50 EXTERNAL EXIT
PROC .MSS55 EXTERNAL EXIT
PROC .MSS60 EXTERNAL EXIT
PROC .MSS70 EXTERNAL EXIT
PROC .MSS80 EXTERNAL EXIT
PROC .MUM00 EXTERNAL EXIT
PROC .MLR10 EXTERNAL EXIT
PROC .MEP00 EXTERNAL EXIT
PROC .MTT10 EXTERNAL EXIT
PROC .MNU00 EXTERNAL EXIT
PROC .MEE00 EXTERNAL EXIT
PROC .MCM00 EXTERNAL EXIT
PROC .MCM10 EXTERNAL EXIT
PROC .MCM20 EXTERNAL EXIT
PROC .MEPWM EXTERNAL EXIT
PROC .MER00 EXTERNAL EXIT
PROC .MDP28 EXTERNAL EXIT
PROC .MTR50 EXTERNAL EXIT
PROC .MTR30 EXTERNAL EXIT
PROC .MTR40 EXTERNAL EXIT
PROC .MDS00 EXTERNAL EXIT
TERM ''EGP1''

```

SPL KERNEL 3 NON-INTERRUPT SEQUENCER

```

START      .NONINTSEQ1  ''NON-INTERRUPT SEQUENCER FOR PHASES 1 AND 3''
ENTRANCE   .NONINTSEQ2  ''NON-INTERRUPT SEQUENCER FOR PHASES 2 AND 4''
NIS1.     IF ARSTAT   EQ 'IN'   THEN .MAR00   .PERPROC   END
           ''ACCELEROMETER READ''
           IF SASSTAT  EQ 'IN'   THEN .MSA00   .PERPROC   END
           ''SIMULATED ACCEL (NOT CODED)''
           IF APSTAT   EQ 'IN'   THEN .MAP00   .PERPROC   END
           ''ACCELEROMETER PROCESSING''
           IF DVSTAT   EQ 'IN'   THEN .MDV00   .PERPROC   END
           ''F/M CALCULATIONS (NOT CODED)''
           IF DPSTAT   EQ 'IN'   THEN .MDP00   .PERPROC   END
           ''DISCRETE PROCESSOR (NOT CODED)''
           IF NESTAT   EQ 'IN'   THEN .MNE00   .PERPROC   END
           ''BOOST NAVIGATION (NOT CODED)''
           IF TCSTAT   EQ 'IN'   THEN .MTC00   .PERPROC   END
           ''RESTART CALCULATIONS (NOT CODED)''
           IF PASTAT   EQ 'IN'   THEN .MPA00   .PERPROC   END
           ''PHASE ACTIVATOR''
           IF TTSTAT   EQ 'IN'   THEN .MTT00   .PERPROC   END
           ''TIME TILT GUIDANCE (NOT CODED)''
           IF CC1STAT  EQ 'IN'   THEN .MCC10   .PERPROC   END
           ''CHI COMPUTATIONS (NOT CODED)''
           IF IGSTAT   EQ 'IN'   THEN .MIG00   .PERPROC   END
           ''ITERATIVE GUIDANCE MODE''
           IF HSSTAT   EQ 'IN'   THEN .MHS00   .PERPROC   END
           ''S48 CUTOFF PREDICTION (NOT CODED)''
           IF OGSTAT   EQ 'IN'   THEN .MOG00   .PERPROC   END
           ''ORBITAL GUIDANCE (NOT CODED)''
           IF TGSTAT   EQ 'IN'   THEN .MTG00   .PERPROC   END
           ''TARGET UPDATE (NOT CODED)''
           IF RSSTAT   EQ 'IN'   THEN .MRS00   .PERPROC   END
           ''TIME-TO-GO TO RESTART (NOT CODED)''
           IF CSSTAT   EQ 'IN'   THEN .MCS00   .PERPROC   END
           ''TIME BASE 6 CHECK (NOT CODED)''
           IF TB1STAT  EQ 'IN'   THEN .MTB10   .PERPROC   END
           ''TIME BASE 1 (NOT CODED)''
           IF TB57STAT EQ 'IN'   THEN .MTB57   .PERPROC   END
           ''TIME BASE 5/7 (NOT CODED)''
           IF MSSTAT   EQ 'IN'   THEN .MMS00   .PERPROC   END
           ''MINOR LOOP SUPPORT (NOT CODED)''
           IF PGSTAT   EQ 'IN'   THEN .MPG00   .PERPROC   END
           ''SIM PLATFORM GIM ANGLE (NOT CODED)''
           IF ERSTAT   EQ 'IN'   THEN .MEB00   .PERPROC   END
           ''ETC/BTC (NOT CODED)''
           GOTO NIS1
NONINTSEQ2. IF CTSTAT  EQ 'IN'   THEN .MCT00   .PERPROC   END
           ''DATA COMPRESSION TELEM (NOT CODED)''
           IF DTSTAT  EQ 'IN'   THEN .MDT00   .PERPROC   END
           ''SECTOR DUMP TELEMETRY (NOT CODED)''
           .PERPROC   ''INSURE PERIODIC PROCESSOR GETS EXECUTED''
           GOTO NONINTSEQ2
PROC       .MCT00   EXTERNAL  EXIT
PROC       .MDT00   EXTERNAL  EXIT
PROC       .MAR00   EXTERNAL  EXIT
PROC       .MSA00   EXTERNAL  EXIT

```

| | | | |
|------|--------------|----------|------|
| PROC | .MAP00 | EXTERNAL | EXIT |
| PROC | .MDV00 | EXTERNAL | EXIT |
| PROC | .MDP00 | EXTERNAL | EXIT |
| PROC | .MNE00 | EXTERNAL | EXIT |
| PROC | .MTC00 | EXTERNAL | EXIT |
| PROC | .MPA00 | EXTERNAL | EXIT |
| PROC | .MTT00 | EXTERNAL | EXIT |
| PROC | .MCC10 | EXTERNAL | EXIT |
| PROC | .MTG00 | EXTERNAL | EXIT |
| PROC | .MHS00 | EXTERNAL | EXIT |
| PROC | .MOG00 | EXTERNAL | EXIT |
| PROC | .MTG00 | EXTERNAL | EXIT |
| PROC | .MRS00 | EXTERNAL | EXIT |
| PROC | .MCS00 | EXTERNAL | EXIT |
| PROC | .MTB10 | EXTERNAL | EXIT |
| PROC | .MTB57 | EXTERNAL | EXIT |
| PROC | .MMS00 | EXTERNAL | EXIT |
| PROC | .MPG00 | EXTERNAL | EXIT |
| PROC | .MEB00 | EXTERNAL | EXIT |
| PROC | .PERPROC | EXTERNAL | EXIT |
| TERM | 'NONINTSEQ1' | | |

SPL KERNEL 4 PERIODIC PROCESSOR

```

START .PERPROC      ''PERIODIC PROCESSOR''
      ITEM VPPOT FIXED 0
ENDDATA
      READ CLOCK,TEMP      ''READ REAL TIME CLOCK''
      DVPTG = (TEMP - VPPOT LAND MSKRTC) RSH 2
      VPPOT = TEMP
      FOR I = 0 BY 1 UNTIL 3
          IF PPSTAT(I) EQ 'OUT'          GOTO PP20
          DLPTL(I) = DLPTL(I) + DVPTG
          IF DLPTL(I) LS DLPRL(I)        GOTO PP20
          GOTO ( , PP1, PP2, *) I
      .MPC50      '' 50 SEC DATA COMP (NOT CODED)'' GOTO PP10
PP1.   .MPC60      '' 60 SEC DATA COMP (NOT CODED)'' GOTO PP10
PP2.   .MPC99      ''100 SEC DATA COMP (NOT CODED)''
PP10.  DLPTL(I) = 0
PP20.
      END
      RETURN          ''PERPROC''
PROC   .MPC50      EXTERNAL EXIT
PROC   .MPC60      EXTERNAL EXIT
PROC   .MPC99      EXTERNAL EXIT
TERM   ''PERPROC''

```

SPL KERNEL 5 EVENTS PROCESSOR

```

START      .MEP00      ''EVENTS PROCESSOR TIMER 2 ENTRY''
ENTRANCE   .MEP05      ''EVENTS PROCESSOR TIME BASE CHANGE ENTRY''
ENTRANCE   .MEP10      ''EVENTS PROCESSOR RESCHEDULE ENTRY''
''
'' EVENTS PROCESSOR TABLE (THROUGH THE END OF TIME BASE 3) ''
''
'' THE POINTERS WITH A VALUE OF ZERO ARE TO BE SET DURING ''
'' PROGRAM EXECUTION OR ARE USED TO DISABLE THE EXECUTION ''
'' OF THE EVENTS PROCESSOR FOR THE REMAINDER OF A TIME BASE. ''
''
TABLE EPTABLE 131 S 2 (EPTPTR LOCATION, EPTTIM FIXED 13)
PRESET EPTABLE = (LOC'LE285.' 0.0 ''START OF TB0 TABLE''
0 16.0
0 17.0
0 17.5
0 0.0
LOC'LE25.' 0.0 ''START OF TB1 TABLE''
LOC'LE30.' 1.0
0 6.0
LOC'LE35.' 9.0
LOC'LE40.' 10.0
0 14.0
LOC'LE50.' 134.7
0 0.0
LOC'LE55.' 0.0 ''START OF TB2 TABLE''
0 0.0
0 18.4
0 27.5
0 0.0
LOC'LE75.' 0.0 ''START OF TB3 TABLE''
LOC'LE70.' 0.0
LOC'LE250.' 0.0
0 0.0
LOC'LE355.' 0.0
LOC'LE365.' 0.0
LOC'LE82.' 1.4
LOC'LE100.' 4.4
LOC'LE95.' 4.4
LOC'LE90.' 6.7
0 6.7
LOC'LE96.' 6.7
LOC'LE105.' 40.6
LOC'LE115.' 40.6
LOC'LE111.' 58.6
LOC'LE110.' 60.6
0 299.0
0 355.0
0 388.5
0 0.0)
ITEM VTOLD FIXED 13
ITEM EPTINDX INTEGER
ARRAY EPTTBINDX(10) INTEGER CONSTANT = (0 5 13 18 38 55 71 93
107 110)
ENDDATA
EPOO. IF DFTBCEP EQ 'CHANGE'

```


SPL KERNEL 5 EVENTS PROCESSOR

```

EP04A.      THEN DFTBCEP = 'NOCHANGE'
              GOTO EP02
            END
            GOTO EPTPTR(EPTINDX) ''EXECUTE REQUIRED MODULE (NONE CODED)''
FPRET.     LOCK T1INT,T2INT,EX1INT,EX2INT,EX3INT,EX4INT,EX5INT,EX6INT,
            EX7INT,EX8INT,EX9INT
            IF DFTBCEP EQ 'CHANGE'          GOTO EP04A
            EPTINDX = EPTINDX + 1
            DQST2 = 'EP00'
            IF FPTPTR(EPTINDX) NQ 0        GOTO EP03
            T2STAT(DQST2) = 'OUT'
            IF DFIL1 EQ 'INACTIVE' .EGP07 ''RESCHEDULE TIMER2(NOT CODED)''
EP02.     UNLOCK ''RELEASE PREVIOUSLY ENABLED INTERRUPTS''
            RETURN ''MEP00''
EP03.     IF EPTTIM(EPTINDX) EQ VTOLD
            THEN UNLOCK ''REL PREV ENABLED INTERRUPTS''
            GOTO EP00
            END
            VTOLD = EPTTIM(EPTINDX)
            DLTTL(DQST2) = DVTMR + VTOLD*4063.492A10
            GOTO EP02
MEP05.     EPTINDX = EPTTBINDX(DTBID) - 1
MEP10.     EPTINDX = EPTINDX + 1
            DQST2 = 'EP00'
            IF EPTPTR(EPTINDX) NQ 0        GOTO EP08
            T2STAT(DQST2) = 'OUT'
FP20.     IF DFIL1 EQ 'INACTIVE' .EGP07 ''RESCHEDULE TIMER2(NOT CODED)''
            RETURN ''MEP05, MEP10''
EP08.     VTOLD = EPTTIM(EPTINDX)
            DLTTL(DQST2) = DVTMR + VTOLD*4063.492A10
            T2STAT(DQST2) = 'IN'
            GOTO EP20
PRQC      .EGP07          EXTERNAL EXIT
TERM      ''MEP00''

```

SPL KERNEL 6 ITERATIVE GUIDANCE MODE

```

START .MIG00 'ITERATIVE GUIDANCE MODE'
''DUE TO THE SIZE OF IGM, ONLY A SECTION OF IT HAS BEEN CODED. ''
''PART OF THE GUIDANCE COMPUTATIONS HAVE BEEN SELECTED TO DEMON-''
''STRATE MATHEMATICAL OPERATIONS. THE PHASING PORTION OF IGM ''
''HAS NOT BEEN CODED SINCE SIMILAR CAPABILITIES ARE ILLUSTRATED ''
''BY OTHER KERNELS. ''
''
DECLARE ARRAY (3) FLOATING R, GS, GV, GVT, GVSTAR,
RS, RV, R4, RVT,
VS, VV, V4, VVT, DELTAVVP,
MS4 (3,3), M4V (3,3)
DECLARE STATUS, CHIBARSTEER (INPROG, NOTINPROG),
PHASE (BURN1, BURN2),
REITERATE (YES, NO),
SMCFLAG (CALCULATE, NOCALC),
S4BURN (BURN1, BURN2)
DECLARE FLOATING R, L1, L2, L12, L3, L3P, LYP, DELTAL3,
J1, J2, J12, J3, J3P,
Q1, Q2, Q12,
P1, P2, P12,
S1, S2, S12,
U1, U2, U12,
T1I, T2I, T3I, T1C, TCI, TSTAR,
TAU1, TAU2, TAU3,
VEX1, VEX2, VEX3, ROVEX3,
K1, K2, K3, K4,
PHI1, PHIT, PHIIT, DPHI1, DPHIT,
DELTA2, EPSILON2, EPSILON3,
SINTHETA, COSTHETA, THETAT,
GT, R, RT, V, VT
DECLARE FLOATING CONSTANT, KT = .48497964E-7,
KMU = -.39860320E15,
KCCT4 = 1.53,
KCCT8 = 1.55
ENDDATA
''
'' IG251 - IGM GUIDANCE PARAMETERS COMPUTATIONS ''
''
'' ROTATE POSITION AND VELOCITY INTO TARGET PLANE ''
''
IG253. R4 = MS4*RS
.UTR00 'DELAY FOR TELEMETRY AS REQUIRED'
WRITE TELX4,R4(0) 'TELEMETER X POSITION IN 4 SYSTEM'
.UTR00 'DELAY FOR TELEMETRY AS REQUIRED'
WRITE TELY4,R4(1) 'TELEMETER Y POSITION IN 4 SYSTEM'
UNLOCK 'RELEASE INTERRUPTS DISABLED BY TELEM DELAY ROUTINE'
V4 = MS4*VS
.UTR00 'DELAY FOR TELEMETRY AS REQUIRED'
WRITE TELZ4,R4(2) 'TELEMETER Z POSITION IN 4 SYSTEM'
.UTR02 'DELAY FOR TELEMETRY AS REQUIRED'
WRITE TELYD4,V4(1) 'TELEMETER Y VELOCITY IN 4 SYSTEM'
UNLOCK 'RELEASE INTERRUPTS DISABLED BY TELEM DELAY ROUTINE'
''
'' CALCULATE RANGE ANGLE MEASURED IN ORBIT PLANE ''
''

```

```

IG254.  IF T2I EQ 0.
        THEN L12,J12,S12,Q12,P12,U12 = 0.
        GOTO IG259
END
IF T1I EQ 0.
        THEN L1,J1,S1,Q1,P1,U1 = 0.
        GOTO IG258
END
L1 = VEX1*.LOG(TAU1/(TAU1 - T1I))
J1 = L1*TAU1 - VEX1*T1I
S1 = L1*T1I - J1
Q1 = S1*TAU1 - .5*VEX1*T1I**2
P1 = J1*TAU1 - .5*VEX1*T1I**2
U1 = Q1*TAU1 - VEX1*T1I**3/6.
IG258.  L2 = VEX2*.LOG(TAU2/(TAU2 - T2I))
        J2 = L2*TAU2 - VEX2*T2I
        S2 = L2*T2I - J2
        Q2 = S2*TAU2 - .5*VEX2*T2I**2
        P2 = J2*TAU2 - .5*VEX2*T2I**2
        U2 = Q2*TAU2 - VEX2*T2I**3/6.
        L12 = L1 + L2
        J12 = J1 + J2 + L2*T1I
        S12 = S1 - J2 + L12*(T2I + TCI)
        Q12 = Q1 + Q2 + S2*T1I + J1*T2I
        P12 = P1 + P2 + T1I*(2.*J2 + L2*T1I)
        U12 = U1 + U2 + T1I*(2.*Q2 + S2*T1I) + T2I*P1
IG259.  L3P = VEX3*.LOG(TAU3/(TAU3 - T3I))
        LYP = L12 + L3P
        J3P = L3P*TAU3 - VEX3*T3I
        T1C = T1I + T2I + TCI
        TSTAR = T1C + T3I
        PHII = .ATAN(R4(2),R4(0))
        ''
        '' DETERMINE PHASE
        ''
IG260.  IF PHASE EQ 'BURN2' 'OUT OF ORBIT'
IG262.  THEN 'CALCULATE TERMINAL CONDITIONS'
        SINTHETA = RS*VS/(R*V)
        COSTHETA = .SQRT(1. - SINTHETA**2)
        DPHII = V/R*COSTHETA
        DPHIT = VT/RT*.COS(THETAT)
        PHIIT = .5*(DPHII + DPHIT)*TSTAR
        PHIT = PHII + PHIIT
        .UTR02 'DELAY FOR TELEMETRY AS REQUIRED'
        WRITE TELPHIT,PHIT 'TELEMETER TERMINAL RANGE ANGLE'
        UNLOCK 'RELEASE INT LOCKED BY TELEM DELAY ROUTINE'
        IF TSTAR LQ EPSILON3 GOTO IG269
        .MIG30 'CALC TERM RAD, VEL, FLT ANGLE (NOT CODED)'
        GT = -KMU/RT**2
        .UTR00 'DELAY FOR TELEMETRY AS REQUIRED'
        WRITE TELGT,GT 'TELEMETER TERMINAL GRAVITY VECT'
        UNLOCK 'RELEASE INT LOCKED BY TELEM DELAY ROUTINE'
        GVT = GT*.COS(THETAT), 0, GT*.SIN(THETAT)
        RVT = RT*.COS(THETAT), 0, 0
IG269.  PHIT = PHIT - THETAT

```

SPL KERNEL 6 ITERATIVE GUIDANCE MODE

```

ELSE 'CALCULATE INTERMEDIATE PARAMETERS'
  DELTA2 = V*TSTAR - J3P + LYP*T3I - ROVEX3*((TAU1 -
    T1I)*L1 + (TAU2 - T2I)*L2 + (TAU3 - T3I)
    *L3P)*(LYP + V - VT)
  PHIIT = KT*(S12 + DELTA2) 'KT = COSTHETAT/RT'
  PHIT = PHII + PHIIT
  .UTR02 'DELAY FOR TELEMETRY AS REQUIRED'
  WRITE TELPHIT,PHIT 'TELEMETER TERMINAL RANGE ANGLE'
  UNLOCK 'RELEASE INT LOCKED BY TELEM DELAY ROUTINE'

END
''
'' ROTATE POSITION, VELOCITY, GRAVITY TO INJECTION SYSTEM ''
''
IG291. M4V = .COS(PHIT), 0., .SIN(PHIT),
        0., 1., 0.,
        -.SIN(PHIT), 0., .COS(PHIT)
RV = M4V*R4
VV = M4V*V4
GV = M4V*MS4*GS
GVSTAR = .5*(GVT + GV)
DELTAVVP = VVT - VV - TSTAR*GVSTAR
''
'' IG314 - CALCULATE TIME TO GO (NOT CODED) ''
''
IF REITERATE EQ 'YES'
  THEN REITERATE = 'NO'
  L3P = L3
  J3P = J3
  LYP = LYP + DELTA3
  GOTO IG260
END
REITERATE = 'YES'
''
'' IG324 - COMPUTE CORRECTED VELOCITIES TO BE GAINED (NOT CODED) ''
''
'' IG326 - CALCULATE DESIRED PITCH AND YAW (NOT CODED) ''
''
IF CHIBARSTEER EQ 'INPROG' GOTO IG350
IF TSTAR GE EPSILON2 GOTO IG360
IF S4BURN EQ 'BURN1'
  THEN DVMC5 = DVMC5 LXOR MSKMC5CBS
  DVMLR = 25.*KCCT4
  DV1MR = .04/KCCT4
  ELSE DVMC6 = DVMC6 LXOR MSKMC6CRS
  DVMLR = 25.*KCCT8
  DV1MR = .04/KCCT8
END
IG340. CHIBARSTEER = 'INPROG'
IG350. K1,K2,K3,K4 = 0
GOTO IG440
''
'' IG360. IG361 - COMPUTE INTERMEDIATE PARAMETERS (NOT CODED) ''
''
IG440. .UTR00 'DELAY FOR TELEMETRY AS REQUIRED'
WRITE TELT3I,T3I 'TELEMETER T3I'

```

SPL KERNEL 6 ITERATIVE GUIDANCE MODE

```
UNLOCK 'RELEASE INTERRUPTS DISABLED BY TELEM DELAY ROUTINE'  
''  
''IG446 - COMPUTE PITCH AND YAW IN 4-SYSTEM (NOT CODED)''  
''  
IF SMCFLAG EQ 'CALCULATE' .MSM00 'COMP SMC TERMS(NOT CODED)''  
.MCC00 'PERFORM CHI COMPUTATIONS (NOT CODED)''  
IF DFILE LAND MSKFPSINT2  
.EGP32(MSK9CC0) 'ENABLE INTERRUPT 2 (NOT CODED)''  
RETURN 'MIG00''  
PROC .MSM00 EXTERNAL EXIT  
PROC .MCC00 EXTERNAL EXIT  
PROC .EXP32 (MASK) EXTERNAL  
ITEM MASK LOGICAL  
EXIT  
TERM 'MIG00''
```

SPL KERNEL 7 DIGITAL COMMAND SYSTEM

```

START .MDS00 'DIGITAL COMMAND SYSTEM'
DECLARE INTEGER, DCSDATACOUNT,
        DCSERLIM CONSTANT = 7,
        VDSRC
DECLARE LOGICAL, VDS01,
        VDSER,
        VDSSB
DECLARE LOGICAL CONSTANT, DCSER04 =0'040000000',
        DCSER10 =0'100000000',
        DCSER14 =0'140000000',
        DCSER20 =0'200000000',
        DCSER24 =0'240000000',
        DCSER44 =0'440000000',
        DCSER60 =0'600000000',
        DCSER64 =0'640000000',
        DCSER74 =0'740000000'
DECLARE STATUS, FDSER (MODE,DATA),
        FDSPG (INPROG,NOTINPROG),
        FDSRE (TERM,NOTERM),
        DCSINX (ILLEGAL,TRUP,NAVUP,GENSS,SECDMP,
        TELSML,TERM,M5UP,M5IN,TARGUP,SWANO,
        SWANLO,SWANHI,INWCVL,TB8EN,EXMANA,
        TDEEN,EXMANB,S4LI,ALTSEQ6D)
DECLARE ARRAY (20), DCSMSTAT STATUS (ACTIVE,INACTIVE),
        DCSSTCOD LOGICAL CONSTANT,
        DCSDATCT INTEGER CONSTANT,
        DCSMODE (64) INTEGER CONSTANT,
        VDSBL (35) LOGICAL
PRESET DCSSTCOD=(0'000000000' 0'100000000' 0'110000000'
        0'120000000' 0'130000000' 0'140000000'
        0'200000000' 0'220000000' 0'050000000'
        0'310000000' 0'770000000' 0'770000000'
        0'770000000' 0'450000000' 0'170000000'
        0'330000000' 0'600000000' 0'340000000'
        0'520000000' 0'250000000'),
        DCSDATCT=(0 1 35 2 2 3 3(0) 35 8(0) 6 0),
        DCSMODE =(5(0) 8 2(0) 1 2 3 4 5 2(0) 14 6 0 7 2(0) 19
        3(0) 9 0 15 17 8(0) 13 4(0) 18 10 11 12 2(0)
        16 15(0))
ENDDATA
UNLOCK 'RELEASE PREVIOUSLY ENABLED INTERRUPTS'
READ DIR,TEMP 'READ DISCRETE INPUT REGISTER'
READ DCS,VDS01 'READ DCS INPUT REGISTER'
IF TEMP LAND MSKDCSMODE EQ 0 GOTO DS60
''
'' PROCESS DCS MODE COMMAND ''
''
DS09. CONDITIONS
        (VDS01 LSH 7 LXOR VDS01) LAND MSKDCSCOMP
        EQ MSKDCSCOMP , (Y, N, , , )
VDS01 LAND MSKDCSSB EQ 0 , (Y, , N, , )
VDS01 LAND MSKDCSMC EQ MSKDCSTERM , (N, , , , )
FDSER EQ 'MODE' , (Y, , , N, )
DFDTL NQ 'INPROG' AND FDSPG NQ 'INPROG', (Y, , , , N)
ACTIONS

```

SPL KERNEL 7 DIGITAL COMMAND SYSTEM

```

VDSER = DCSER10      ,( , Y, , , )
VDSER = DCSER24      ,( , , Y, , )
VDSER = DCSER20      ,( , , , Y, )
VDSER = DCSER64      ,( , , , , Y)
GOTO DS220            ,( , Y, Y, Y, Y)
GOTO DS20            ,(Y, , , , )
ELSE GOTO DS25
END
DS20.  FDSPG = 'INPROG'
DS25.  DCSINDX = DCSMODE(VDS01 RSH 20)
IF DCSMSTAT(DCSINDX) EQ 'INACTIVE'
    THEN FDSPG = 'NOTINPROG'
    VDSER = DCSER74
    GOTO DS220
END
''TELEMETER STATUS CODE TWICE''
.UTR24  ''DELAY FOR TELEMETRY AS REQUIRED''
WRITE TELDCSSC,DCSSTCOD(DCSINDX)
.UTR24  ''DELAY FOR TELEMETRY AS REQUIRED''
WRITE TELDCSSC,DCSSTCOD(DCSINDX)
UNLOCK ''RELEASE INTERRUPTS DISABLED BY TELEM DELAY ROUTINE''
GOTO DS200  ''ISSUE CRP''
DCSDATACOUNT, VDSSB = 0
GOTO DS100
''
'' PROCESS DCS DATA WORD
''
DS60.  CONDITIONS
    FDSEN EQ 'DATA'      ,(Y, N, )
    (VDS01 LSH 7 LXOR VDS01) LAND MSKDCSCOMP
    EQ MSKDCSCOMP      ,(Y, , N)
    VDS01 LAND MSKDCSSB EQ VDSSB      ,(Y, , )
ACTIONS
    GOTO DS110          ,(Y, , )
    VDSER = DCSER04      ,( , Y, )
    VDSER = DCSER44      ,( , , Y)
ELSE VDSER = DCSER60
END GOTO DS220
DS110. ''TELEMETER DATA WORD TWICE''
.UTR24  ''DELAY FOR TELEMETRY AS REQUIRED''
WRITE TELDCSDW,VDS01
.UTR24  ''DELAY FOR TELEMETRY AS REQUIRED''
WRITE TELDCSDW,VDS01
UNLOCK ''RELEASE INTERRUPTS DISABLED BY TELEM DELAY ROUTINE''
GOTO DS200  ''ISSUE CRP''
VDSRL(DCSDATACOUNT) = VDS01 LAND MSKDCSMC
VDSSB = VDSSB LXOR MSKDCSSB
DCSDATACOUNT = DCSDATACOUNT + 1
DS100. IF DCSDATACOUNT LS DCSDATCT(DCSINDX)
    RETURN  ''MDS00, MORE DATA IS TO BE RECEIVED''
GOTO ( ,DS01,DS02,DS03,DS04,DS05,DS06,DS07,DS08,DS09A,DS10,DS11,
    DS12,DS13,DS14,DS15,DS16,DS17,DS18,DS19) DCSINDX
FDSPG = 'NOTINPROG'
VDSER = DCSER14
GOTO DS220

```

SPL KERNEL 7 DIGITAL COMMAND SYSTEM

```

DS01.      .DS260          ''TIME BASE UPDATE (NOT CODED)'' GOTO DS530
DS02.      .DS330 (= DS235.) ''NAVIGATION UPDATE (NOT CODED)'' GOTO DS530
DS03.      .DS380 (= DS220.) ''GENERALIZED SS (NOT CODED)'' GOTO DS530
DS04.      .DS430          ''SECTOR DUMP (NOT CODED)'' GOTO DS530
DS05.      .DS470          ''SINGLE MEM LOC TEL (NOT CODED)'' GOTO DS530
DS06.      .DS510          ''TERMINATE (NOT CODED)'' GOTO DS530
DS07.      .DS540          ''MANEUVER UPDATE (NOT CODED)'' GOTO DS530
DS08.      .DS550          ''MANEUVER INHIBIT (NOT CODED)'' GOTO DS530
DS09A.     .DS670 (= DS235.) ''TARGET UPDATE (NOT CODED)'' GOTO DS530
DS10.      .DS700          ''ANTENNA TO OMNI (NOT CODED)'' GOTO DS530
DS11.      .DS720          ''ANTENNA TO LOW (NOT CODED)'' GOTO DS530
DS12.      .DS740          ''ANTENNA TO HIGH (NOT CODED)'' GOTO DS530
DS13.      .DS770          ''INHIBIT WATER CONT (NOT CODED)'' GOTO DS530
DS14.      .DS790          ''TIME BASE R ENABLE (NOT CODED)'' GOTO DS530
DS15.      .DS810          ''EXECUTE MANEUVER A (NOT CODED)'' GOTO DS530
DS16.      .DS840          ''TD AND E ENABLE (NOT CODED)'' GOTO DS530
DS17.      .DS860          ''EXECUTE MANEUVER B (NOT CODED)'' GOTO DS530
DS18.      .DS900          ''S4B/IU LUNAR IMPCT (NOT CODED)'' GOTO DS530
DS19.      .DS960          ''ENABLE TR6D ALT SQ (NOT CODED)'' GOTO DS530
''
'' PROCESS DCS ERROR CONDITION ''
''
DS220.     VDSRC = VDSRC + 1
           IF VDSRC LS DCSERLIM
             THEN FDSRE = 'NOTERM'
             ELSE FDSRE = 'TERM'
           END
           VDSER = VDSEK + VDSRC + (VDS01 RSH 12 LAND MSKDCSER)
DS235.    ''TELEMETER ERROR CODE TWICE''
           .UTR24 ''DELAY FOR TELEMETRY AS REQUIRED''
           WRITE TELDCSEC,VDSER
           .UTR24 ''DELAY FOR TELEMETRY AS REQUIRED''
           WRITE TELDCSEC,VDSER
           UNLOCK ''RELEASE INTERRUPTS DISABLED BY TELEM DELAY ROUTINE''
           IF FDSRE EQ 'NOTERM'
             RETURN ''MDS00''
DS530.     VDSRC = 0
           FDSER = 'MODE'
           FDSPE = 'NOTINPROG'
           RETURN ''MDS00''
CLOSE     DS200 ''ISSUE DCS COMMAND RESET PULSE''
           LOCK T1INT,T2INT,EX1INT,EX2INT,EX3INT,EX4INT,EX5INT,EX6INT,
             EX7INT,EX8INT,EX9INT
           WRITE DCS,MSKDCSDO ''RESET COMMAND RECEIVER''
           FOR I = 35 WHILE I GR 0 ''DELAY 4.13 MS''
             I = I - 1
           END
           WRITE DOR,MSKDCSDO ''RESET THE RESET COMMAND''
           UNLOCK ''RELEASE PREVIOUSLY ENABLED INTERRUPTS''
EXIT      ''DS200''
TERM      ''MDS00''

```


SPL KERNEL 8 ACCELEROMETER PROCESSING

```

START .MAROO    'ACCELEROMETER READ ROUTINE'
ENTRANCE .MAPOO  'ACCELEROMETER PROCESSING ROUTINE'
DECLARE FLOATING R, KSN2D CONSTANT =.0348994697, 'SIN 2 DEG'
      COSTHY,
      COSTHZ,
      SINTHY,
      SINTHZ,
      VACZR
ARRAY VPOV (3) FIXED 7
DECLARE FIXED, DELTA 7 ,
      VCCYA 25 R,
      VCCYZ 25 R,
      VOACT 28 -2
DECLARE ARRAY (3) LOGICAL, VOAC,
      MSKAPDG CONSTANT =(OCT'04000000'
                        OCT'01000000'
                        OCT'20000000'),
      MSKAPDF CONSTANT =(OCT'00000010'
                        OCT'00000020'
                        OCT'00000020')
ENDDATA
LOCK T1INT,T2INT,EX1INT,EX2INT,EX3INT,EX4INT,EX5INT,EX6INT,
EX7INT,EX8INT,EX9INT
READ XACC,DVAC(0)  'READ X ACCELEROMETER'
READ YACC,DVAC(1)  'READ Y ACCELEROMETER'
READ ZACC,DVAC(2)  'READ Z ACCELEROMETER'
READ CLOCK,DVACT  'READ REAL TIME CLOCK'
.UTROO  'DELAY FOR TELEMETRY AS REQUIRED'
WRITE TELTI,DVTI  'TELEMETER START TIME OF CURRENT TIME BASE'
TEMP = DVTAS
VOACT = DVTMM + (DVACT - DVRTC - DVERT LAND MSKRTC) SCL 0
DVTAS = .24609375E-3 * VOACT
DVTR = DVTAS - DVTI
DVDT = DVTAS - TEMP
.UTROO  'DELAY FOR TELEMETRY AS REQUIRED'
WRITE TELTB,DVTB  'TELEMETER TIME IN CURRENT TIME BASE'
DVMC4 = DVMC4 LAND MSKRTCRESET
UNLOCK 'RELEASE PREVIOUSLY ENABLED INTERRUPTS'
.UTROO  'DELAY FOR TELEMETRY AS REQUIRED'
WRITE TELXAC,DVAC(0) 'TELEMETER X ACCELEROMETER READING'
.UTROO  'DELAY FOR TELEMETRY AS REQUIRED'
WRITE TELYAC,DVAC(1) 'TELEMETER Y ACCELEROMETER READING'
UNLOCK 'RELEASE INTERRUPTS DISABLED BY TELEM DELAY ROUTINE'
IF 'TIME BASE 1' DKT1 EQ 0. 'NOT SET'
  THEN DVFMC = - DVG(0)
  ELSE DVMAS = DVMAS - DVEOF*DVMFR*DVDT
  DVFMC = DVEOF*DVFOR/DVMAS
END
''
''COMPUTE AVERAGE CHI'S FOR SMC CALCULATIONS''
''
AR41.  DVCA(2) = DVCC(2) RSH 1 + VCCZA RSH 1
      VCCZA = DVCC(2)
      DVCA(1) = DVCC(1) RSH 1 + VCCYA RSH 1
      IF ABS(DVCC(1) - VCCYA) GQ .5A25

```

SPL KERNEL 8 ACCELEROMETER PROCESSING

```

        DVCA(1) = DVCA(1) + MSK180DEG
        VCCYA = DVCC(1)
        .UTROO    ''DELAY FOR TELEMETRY AS REQUIRED''
        WRITE TELZAC,DVAC(2) ''TELEMETER Z ACCELEROMETER READING''
        UNLOCK ''RELEASE INTERRUPTS DISABLED BY TELEM DELAY ROUTINE''
    ''
    ''COMPUTE CHANGES BETWEEN CURRENT AND PREVIOUS ACCELEROMETER    ''
    ''READINGS                                                         ''
    ''
AR100.    DVDA = ((DVAC LAND MSKACCELA) - (VOAC LAND MSKACCELA)) RSH 7
          DVDR = ((DVAC LAND MSKACCELB) - (VOAC LAND MSKACCELB)) LSH 14)
          RSH 7
          VOAC = DVAC
          .UTROO    ''DELAY FOR TELEMETRY AS REQUIRED''
          WRITE TELRTC,DVACT ''TELEMETER REAL TIME CLOCK AT ACCEL READ''
          UNLOCK ''RELEASE INTERRUPTS DISABLED BY TELEM DELAY ROUTINE''
    ''
    ''COMPUTE THE EXPECTED VELOCITY CHANGES                           ''
    ''
AR71.    .USCOO (DVTH(2) = SINTHZ COSTHZ) ''SIN/COS (NOT CODED)''
          .UTROO    ''DELAY FOR TELEMETRY AS REQUIRED''
          WRITE TELTAS,DVTAS ''TELEMETER MISSION ELAPSED TIME''
          UNLOCK ''RELEASE INTERRUPTS DISABLED BY TELEM DELAY ROUTINE''
          .USCOO (DVTH(1) = SINTHY,COSTHY) ''SIN/COS (NOT CODED)''
          DVD = 20.*DVDT*(COSTHY*COSTHZ, SINTHZ, -SINTHY*COSTHZ)
          DVF = DVFOM*DVD
          RETURN ''NAROO''
MAP00.    DVVSQ = 0.
          VACZR = 20.*DVFOM*DVDI*KSN2D
          FOR I = 0 BY 1 UNTIL 3
AP400.    IF ABS(DVDA(I) - DVDB(I)) LG 2.A7 GOTO AP450
          IF ABS(DVDA(I) - DVF(I)) LS ABS(DVDR(I) - DVF(I))
          GOTO AP440
          DVMC4 = DVMC4 LOR MSKAPDG(I) LSH 1
          DELTA = DVDR(I)
          GOTO AP460
AP440.    DVMC4 = DVMC4 LOR MSKAPDG(I)
AP450.    DELTA = DVDA(I)
AP460.    IF ABS(DELTA) GR 1.A7 GOTO AP500
          IF DFZER EQ 'DISABLE' GOTO AP500
          IF ABS(DVF(I)) LS VACZR GOTO AP500
AP470.    DVMC4 = DVMC4 LOR MSKAPDF(I)
AP530.    DVMC4 = DVMC4 LOR MSKAPDG(I) LOR MSKAPDG(I) LSH 1
          DFMC = 'DISABLE'
          DELTA = DVFMC*DVD(I)
          GOTO AP520
AP500.    IF DVF(I) LS 0.
          THEN IF DELTA LS 1.5*DVF(I) - DVRC(I)*DVDT
          OR DELTA GR .5*DVF(I) + DVRC(I)*DVDT GOTO AP530
          ELSE IF DELTA GR 1.5*DVF(I) + DVRC(I)*DVDT
          OR DELTA LS .5*DVF(I) - DVRC(I)*DVDT GOTO AP530
          END
AP510.    DVVSQ = DVVSQ + DELTA**2
AP520.    VPOV(I) = VPOV(I) + DELTA
          DVDM(I) = .05*VPOV(I)

```

SPL KERNEL 8 ACCELEROMETER PROCESSING

```
.UTR01 ''DELAY FOR TELEMETRY AS REQUIRED''  
GOTO (AP521, AP522, AP523, *) I  
AP521. WRITE TELXDM,DVDM(0) ''TELEMETER X MEASURED VELOCITY''  
GOTO AP524  
AP522. WRITE TELYDM,DVDM(1) ''TELEMETER Y MEASURED VELOCITY''  
GOTO AP524  
AP523. WRITE TELZDM,DVDM(2) ''TELEMETER Z MEASURED VELOCITY''  
AP524. UNLOCK ''RELEASE INTERRUPTS LOCKED BY TELEM DELAY ROUT''  
END  
RETURN ''MAP00''  
TERM ''MAR00''
```

SPL KERNEL 9 MINOR LOOP

```

START      .MML00    ''FLIGHT SIM ENTRY TO MINOR LOOP''
ENTRACE    .MML20    ''NORMAL MINOR LOOP ENTRY''
            DECLARE ARRAY (3) FIXED, VML0 14,
            VML1 14,
            VML2 25,
            VCG0 14,
            VCG1 14,
            VCOD 14,
            VOLD 14,
            VDEL 25,
            VCG 25,
            VSF 35 R,
            VBUB 25,
            VCMND 0 R,
            VCMND1 0 R,
            VCMND2 0 R

            DECLARE FIXED,
            KCPBG 14 CONSTANT = 2016.,
            VOCK 25,
            VCG10 14,
            VCG11 14

            DECLARE INTEGER, VIRE
            DECLARE ARRAY (3) STATUS, FBUG (NONE,PASS1,PASS2),
            VFIO (NORMAL,BACKUP,DUMMY)

            DECLARE STATUS FBUGS (NONE,PASS1,PASS2)
            DECLARE ARRAY (3) LOGICAL, VGR, VPGR
            DECLARE LOGICAL, VMEMR, VMLET
            ITEM J INTEGER

ENDDATA

            IF DVLRC EQ 0 GOTO ML01
            DVLRC = DVLRC - 1
MML20.     DVCC = DVCC + DVDC
ML01.     IF FBUGS NG 'NONE' GOTO ML500
ML001.    FOR I = 2 BY -1 WHILE I GO 0
            GOTO (ML201, ML101, , * ) I
            IF VFIO(2) EQ 'NORMAL' THEN READ ZGIM,VGR(2)
            ORIF VFIO(2) EQ 'BACKUP' THEN READ ZBGIM,VGR(2)
            ELSE VGR(2) = VPGR(2)

            END
            GOTO MLOU4
ML101.    READ EMR,VMEMR ''READ ERROR MONITOR REG''
            DVLDR = DVLDR - (VMEMR LAND MSKEMRLADB)
            IF VFIO(1) EQ 'NORMAL' THEN READ YGIM,VGR(1)
            ORIF VFIO(1) EQ 'BACKUP' THEN READ YBGIM,VGR(1)
            ELSE VGR(1) = VPGR(1)

            END
            GOTO MLOU4
ML201.    DVEMR = DVEMR LOR VMEMR
            IF VFIO(0) EQ 'NORMAL' THEN READ XGIM,VGR(0)
            ORIF VFIO(0) EQ 'BACKUP' THEN READ XBGIM,VGR(0)
            ELSE VGR(0) = VPGR(0)

            END
ML004.    IF VGR(I) GO 0 GOTO ML020
ML430.    IF DVDGS LS 0 GOTO ML432
            IF DVDGS EQ 0 GOTO ML020

```

SPL KERNEL 9 MINOR LOOP

```

                GOTO ML637
ML432.          .MDG00 (= J, ML434.) 'PROCESS DISAGREEMENT BIT(UNCODED)'
                'DISAGREEMENT BIT PROCESSING WILL TAKE A NORMAL RETURN IF THE
                'DISAGREEMENT BIT IS FOUND TO BE INVALID. OTHERWISE IT WILL
                'TAKE THE ERROR EXIT TO ML434 AND SET J = 0 IF THE GIMBAL IS
                'VALID OR J = 1 IF THE GIMBAL IS NOT VALID.
                ''
                GOTO ML020
ML434.          GOTO (ML4352, ML4351, ML4350, *) I
ML4350.         IF VFIO(2) EQ 'NORMAL'
                THEN READ ZGIM,TEMP      'RESTART COD COUNTER'
                ELSE READ ZBGIM,TEMP     'RESTART COD COUNTER'
                END
                GOTO (ML020, ML637, *) J
ML4351.         IF VFIO(1) EQ 'NORMAL'
                THEN READ YGIM,TEMP      'RESTART COD COUNTER'
                ELSE READ YBGIM,TEMP     'RESTART COD COUNTER'
                END
                GOTO (ML020, ML637, *) J
ML4352.         IF VFIO(0) EQ 'NORMAL'
                THEN READ XGIM,TEMP      'RESTART COD COUNTER'
                ELSE READ XBGIM,TEMP     'RESTART COD COUNTER'
                END
                GOTO (ML020, ML637, *) J
ML020.          VCOD(I) = VGR(I) LAND MSKGIMBALA
                CONDITIONS
                VCOD(I) EQ 0              ,(Y, , , )
                VOLD(I) EQ 0              ,(Y, , , )
                ABS(VDEL(I)) GQ VOCK      ,(Y, , , )
                ABS(VCOD(I) - VOLD(I)) LS VML0(I) ,( , Y, , )
                ABS(VCOD(I) - VOLD(I)) + VML0(I) GQ VML1(I), ( , , Y, Y)
                VCOD(I) - VOLD(I) LS 0    ,( , , Y, N)
                ACTIONS
                GOTO ML631                  ,(Y, , , )
                VCG(I) = VCG(I) + VML2(I) ,( , , Y, )
                VCG(I) = VCG(I) - VML2(I) ,( , , , Y)
                GOTO ML040                  ,( , Y, Y, Y)
                ESLE GOTO ML630
                END
ML040.          DVTH(I) = VSF(I)*VCOD(I) + VCG(I)
                VOLD(I) = VCOD(I)
                VDEL(I) = DVTH(I) - DVCC(I)
                DFDBF = 'GOOD'
                GOTO (ML245, ML145, ML045, * ) I
ML045.          VCMND(2) = DVA5*VDEL(2) - DVA4*VDEL(1)
                GOTO ML730
ML145.          VCMND(1) = DVA1*VDEL(1) + DVA2*VDEL(2)
                GOTO ML730
ML245.          VCMND(0) = DVA6*(VDEL(0) + DVA3*VDEL(1))
                GOTO ML730
ML630.          VMLET = I + 3
                GOTO ML632
ML631.          VMLET = I
ML632.          VMLET = VMLET LSH 11 + VCOD(I) RSH 14 + VOLD(I)
                IF DVMC6 LAND MSKMC6D04 EQ 0 THEN

```

SPL KERNEL 9 MINOR LOOP

```

      .UTR30    ''DELAY FOR TELEMETRY AS REQUIRED''
      WRITE TELMLR,VMLET ''TELEM MINOR LOOP ERROR MESSAGE''
END
IF DFDBF EQ 'FAILED'      GOTO ML635
DVRE(I) = DVRE(I) + 1
IF DVRE(I) LS 0          GOTO ML637
IF DVRE(I) GR 0          GOTO ML636
VMLET = VCOD(I) RSH 14 + VOLD(I) + MSKERRORTAG
VFIO(I) = 'BACKUP'
VCG(I) = (VCG(I) LAND MSK180DEG) - VRUB(I)
VML2(I) = MSK180DEG
VOLD(I) = (DVTH(I) LAND MSKMOD180DEG)*KCPRG
          LAND MSKGIMBALA
VSF(I) = 1./KCPRG
IF I EQ 2
      THEN WRITE ICR,MSKICR8G    ''SET INTERNAL CON REG''
           DVICR = DVICR LXOR MSKICR8G
END
FRUGS = FRUG(I) = 'PASS1'
VML0(I) = VCG10
VML1(I) = VCG11
IF DVMC6 LAND MSKMC6D04 EQ 0 THEN
      .UTR30    ''DELAY FOR TELEMETRY AS REQUIRED''
      WRITE TELMLR,VMLET ''TELEM MINOR LOOP ERROR MESSAGE''
END
GOTO ML637
ML635.  DVHDR = DVHDB - 1
        DFDRF = 'GOOD'
        DVHDA = DVHDA + 1
        IF DVHDA LS 0          GOTO ML636
        WRITE ICR,MSKICRSWG    ''SET INTERNAL CONTROL REG.''
        DVICR = DVICR LXOR MSKICRSWG
        DVMC4 = DVMC4 LOR MSKMC4AMF
        DVDGS = 0
ML636.  IF DVRE(I) LS VIRE      GOTO ML637
        IF DVMC6 LAND MSKMC6D04 EQ 0 .UD000(MSKGRF) ''SET
          GUIDANCE REFERENCE FAILURE DISCRETES (NOT CODED)''
ML637.  DFSMC = 'DISABLE'
        GOTO ML760
ML730.  IF ABS(VCMND(I)) GR DVM06    VCMND(I) = DVM06
        IF ABS(VCMND(I) - VCMND1(I)) GR DVM05
          VCMND(I) = VCMND1(I) + DVM05
VCMND1(I) = VCMND(I)
IF VCMND(I) LS 0
      THEN VCMND2(I) = MSKABSLADDER - VCMND(I)
      ELSE VCMND2(I) = VCMND(I)
END
ML760.  GOTO (ML260, ML160, ML060, * ) I
ML060.  WRITE ZLAD,VCMND2(2)    ''ISSUE YAW COMMAND''
        IF DVLDB LS 0 WRITE ICR,MSKICRCA
        DVMLT = DVTNM + DVMLD + (DVTT1 - DVRTC LAND MSKRTC) RSH 2
        GOTO LOOPEND
ML160.  WRITE YLAD,VCMND2(1)    ''ISSUE PITCH COMMAND''
        GOTO LOOPEND
ML260.  READ DBG,TEMP ''START SPECIAL DOM BACKUP GIMBAL''

```

SPL KERNEL 9 MINOR LOOP

```

WRITE ZLAD,VCMND2(2)    ''ISSUE YAW COMMAND''
WRITE XLAD,VCMND2(0)    ''ISSUE ROLL COMMAND''
LOOPEND. END
RETURN    ''MML00, MML20''
ML500.   FOR I=0 BY 1 UNITL 3
        GOTO (ML530, ML520, ML540, * ) FBUG(I)
ML520.   FBUG(I) = 'PASS2'
        GOTO ML530.
ML540.   FBUG(I) = 'NONE'
        VML0(I) = VCG0(I)
        VML1(I) = VCG1(I)
ML530.   END
        FBUGS = FBUG(0) LOR FBUG(1) LOR FBUG(2)
        GOTO ML001.
PROC .MD000 (= J, EREXIT.)    EXTERNAL
EXIT
TERM    ''MML00''

```

SPL KERNEL 10 SWITCH SELECTOR PROCESSING

```

START      .MSS00    ''SWITCH SELECTOR CHECK FOR ALTERNATE SEQUENCE''
ENTRANCE   .MSS05    ''NORMAL SWITCH SELECTOR CHECK''
ENTRANCE   .MSS10    ''SWITCH SELECTOR INITIALIZATION FOR TIME BASE SET''
ENTRANCE   .MSS20    ''SWITCH SELECTOR FORCED RESET''
ENTRANCE   .MSS30    ''SWITCH SELECTOR HUNG STAGE TEST''
ENTRANCE   .MSS40    ''SWITCH SELECTOR STAGE AND ADDRESS ISSUANCE''
ENTRANCE   .MSS50    ''SWITCH SELECTOR VERIFY ADDRESS''
ENTRANCE   .MSS55    ''SWITCH SELECTOR READ TIME CHFK''
ENTRANCE   .MSS60    ''SWITCH SELECTOR READ ISSUANCE''
ENTRANCE   .MSS70    ''SWITCH SELECTOR RESET''
ENTRANCE   .MSS80    ''SWITCH SELECTOR COMPLEMENT STAGE AND ADDRESS''

```

```

''
'' SWITCH SELECTOR TABLE ''
''
'' SSTTB1 IS THE SS TABLE FOR TIME BASE 1. OTHER TABLES ''
'' WITH IDENTICAL FORMATS WOULD BE PROVIDED FOR THE OTHER ''
'' TIME BASES AND THE ALTERNATE SEQUENCES. ''
''

```

TABLE SSTTB1 28 SERIAL 2 CONSTANT

```

(TIME      FIXED 10 0 0,
 STAGE     LOGICAL 5 1 0,
 ADDRESS   LOGICAL 8 1 5 )
=( 5.0     OCT'00'  OCT'000'
   6.0     OCT'04'  OCT'152'
   14.0    OCT'01'  OCT'142'
   19.8    OCT'20'  OCT'151'
   20.0    OCT'20'  OCT'131'
   20.2    OCT'20'  OCT'147'
   24.0    OCT'01'  OCT'137'
   27.0    OCT'20'  OCT'042'
   29.0    OCT'01'  OCT'177'
   30.0    OCT'20'  OCT'061'
   32.0    OCT'20'  OCT'022'
   49.5    OCT'01'  OCT'002'
   75.0    OCT'00'  OCT'000'
   90.0    OCT'20'  OCT'042'
   95.0    OCT'20'  OCT'022'
   95.3    OCT'01'  OCT'055'
  105.0    OCT'20'  OCT'175'
  115.1    OCT'01'  OCT'137'
  119.8    OCT'20'  OCT'157'
  120.0    OCT'20'  OCT'137'
  120.1    OCT'01'  OCT'177'
  130.0    OCT'20'  OCT'101'
  132.4    OCT'01'  OCT'035'
  133.6    OCT'20'  OCT'016'
  133.8    OCT'20'  OCT'036'
  134.4    OCT'01'  OCT'037'
  134.6    OCT'01'  OCT'077'
OCT'37777776' OCT'00'  OCT'000')
DECLARE FIXED -2 CONSTANT, KSS500MS    = 2031.7460,
                               KSS500SEC = 2031746.0,
                               KCSSK     = 812.69840,
                               KSSB1     = 70.142856,
                               KSSB2     = 103.58730,

```


SPL KERNEL 10 SWITCH SELECTOR PROCESSING

KSSB3 = 66.206348,
 KSSB4 = 35.825396,
 KSSB5 = 102.65079,
 KSSB6 = 50.825396,
 KSSB7 = 87.460316,
 KSSB8 = 43.825396,
 KSSCK = 2031.7460,
 KSSRB = 201.17460,
 KCSR 13 = 4063.492

DECLARE FIXED -2, VATRR,
 VATR4,
 VGBIA,
 VSSRT,
 VSSTM,
 VSSW,
 VSTG0
 DECLARE LOCATION, SST1PTR,
 SST2PTR
 ARRAY (8) SSTTBPTR LOCATION = (LOC'SSTTB1'
 LOC'SSTTB2'
 LOC'SSTTB3'
 LOC'SSTTB4'
 LOC'SSTTB5'
 0
 LOC'SSTTB7'
 LOC'SSTTB8')

DECLARE ARRAY (3) LOGICAL, VSC1,
 VSC3

DECLARE LOGICAL, VASPI,
 VHSTW,
 VPSTG,
 VSCCA,
 VSNA,
 VSNA1,
 VSSCA,
 VSSF8,
 VSTG

DECLARE STATUS, FASE (NORMAL,ALTERNATE),
 FBRNI (FIRST,SECOND),
 FCLS4 (NOTINPROG,INPROG),
 FFBCH (CHANA,CHANB),
 FHST (NOTEST,TEST),
 FSSAC (INACTIVE,ACTIVE),
 FSSIO (YES,NO),
 FTADV (NORMAL,CLASS4),
 FT60P (PASS1,PASS2)

ENDDATA

LOCK T1INT,T2INT,EX1INT,EX2INT,EX3INT,EX4INT,EX5INT,EX6INT,
 EX7INT,EX8INT,EX9INT

FASE = 'ALTERNATE'

CONDITIONS

DVASW LAND MSKSSS4C0 NQ 0 , (Y , Y , , , , , ,)
 DVASW LAND MSKSSSPEC NQ 0 , (, , Y , , , , ,)
 DVASW LAND MSKSSRB6C NQ 0 , (, , , Y , , , ,)
 DVASW LAND MSKSSCLS1 NQ 0 , (, , , , Y , Y , Y ,)

SPL KERNEL 10 SWITCH SELECTOR PROCESSING

```

DVASW LAND MSKSSTB6A NQ 0      ,( , , , , Y, N, N, )
DVASW LAND MSKSSS4C1 NQ 0      ,( , , , , , Y, N, )
VASPI LAND MSKSSS4C0 NQ 0      ,(Y, N, , , , , )
FSSAC EQ 'ACTIVE'              ,( , , , , , , , Y)

ACTIONS
VSC1(0) = SST1PTR              ,( , , , , Y, Y, Y, )
VSC1(1) = VASPI                 ,( , , , , , Y, Y, Y, )
VSC1(2) = VATRR                 ,( , , , , , Y, Y, Y, )
DVASW = DVASW LAND MSKSSWV      ,(Y, Y, Y, , , , )
DVASW = DVASW LXOR MSKSSTB6C    ,( , , , , Y, , , )
DVASW = DVASW LXOR MSKSSTB6A    ,( , , , , , Y, , )
DVASW = DVASW LXOR MSKSSS4C1    ,( , , , , , , Y, )
DVASW = DVASW LXOR MSKSSTB6B    ,( , , , , , , , Y, )
.EGPO8 'RESCHED T1(UNCODED)''  ,( , Y, Y, , , , )
VASPI = MSKSSS4C0               ,( , Y, , , , , )
VASPI = MSKSSSPEC               ,( , , Y, , , , )
VASPI = VASPI LOR MSKSST6C      ,( , , , , Y, , , )
VASPI = MSKSSCL1               ,( , , , , , Y, Y, Y, )
DVMC6 = DVMC6 LOR MSKMC6TB6C    ,( , , , , Y, , , )
DVMC6 = DVMC6 LOR MSKMC6TB6A    ,( , , , , , Y, , )
DVMC6 = DVMC6 LOR MSKMC6TB6R    ,( , , , , , , , Y, )
SST1PTR = LOC'SSTSIVB'          ,( , Y, , , , , )
SST1PTR = LOC'SSTSIVA'          ,( , , Y, , , , )
SST1PTR = LOC'SSTTB6C'          ,( , , , , Y, , , )
SST1PTR = LOC'SSTTB6A'          ,( , , , , , Y, , )
SST1PTR = LOC'SSTS4C1'          ,( , , , , , , Y, )
SST1PTR = LOC'SSTTB6B'          ,( , , , , , , , Y, )
.SSTUPD (= VATRR)              ,( , , , , Y, Y, Y, Y, )
    'UPDATE SS TIME''
FTADV = 'NORMAL'                ,( , , , , , Y, Y, Y, )
GOTO SS0060                      ,(Y, , , , , , , Y)
GOTO SS1050                      ,( , Y, Y, Y, Y, Y, Y, )
ELSE GOTO SS0000

END
MSS05. FSSAC = 'INACTIVE'
SS0000. IF IND(SST1PTR) NQ MSKSSNSEND
SS0010. THEN .SSTUPD (= VSTGO) 'UPDATE SS TIME''
        VSTGO = VSSRT - VSTGO
        IF VSTGO LS KSS500MS      GOTO MSS30
        IF DFTUP EQ 'YES'
            THEN DVTGB = DVTGB + VGBIA
                VGBIA = 0
                DFTUP = 'NO'
                GOTO SS0010
            ELSE IF DVASW NQ 0      GOTO SS0170
                VSSTM = VSTGO + DVTRB - DVTGB - KCSSK
                DVSST = VSSTM + DVTMR
                DGSSM = 'SS30'
                IF DFIL3 EQ 'INACTIVE' .EGPO8 'RESCHEDT1''
        END
        GOTO SS0060

END
SS0015. IF DVASW NQ 0              GOTO SS0170
        READ CLOCK,TEMP            'READ REAL TIME CLOCK''
        DVSST = DVTMM + KSS500SEC + (TEMP - DVRTC LAND MSKRTC) SCL 0

```

SPL KERNEL 10 SWITCH SELECTOR PROCESSING

DGSSM = 'SS05'
 IF DFIL3 EQ 'INACTIVE' .EGP08 ''RESCHED TIMER 1 (NOT CODED)''
 GOTO SS0060

SS0170.

CONDITIONS
 DVASW LAND MSKSSCLS3 EQ 0 ,(Y, Y, Y, , , , , , , ,)
 DVASW LAND MSKSSACQU NQ 0 ,(Y, , , , , , , , , ,)
 DVASW LAND MSKSSTB6D NQ 0 ,(, Y, N, , , , , , , ,)
 VASPI EQ 0 ,(, , , Y, Y, Y, Y, Y, Y, Y)
 DVASW LAND MSKSSGNSS NQ 0 ,(, , , Y, , , , , , ,)
 DVASW LAND MSKSSSBLO NQ 0 ,(, , , , Y, , , , , ,)
 DVASW LAND MSKSSSBHI NQ 0 ,(, , , , , Y, , , , ,)
 DVASW LAND MSKSSSBOM NQ 0 ,(, , , , , , Y, , , ,)
 DVASW LAND MSKSSECSV NQ 0 ,(, , , , , , , Y, , ,)
 DVASW LAND MSKSSECS1 NQ 0 ,(, , , , , , , , Y, N)

ACTIONS
 VSC3(0) = SST1PTR ,(, , , Y, Y, Y, Y, Y, Y, Y)
 VSC3(1) = VASPI ,(, , , Y, Y, Y, Y, Y, Y, Y)
 VSC3(2) = VATRR ,(, , , Y, Y, Y, Y, Y, Y, Y)
 DVASW = DVASW LXOR MSKSSACQU,(Y, , , , , , , , , ,)
 DVASW = DVASW LXOR MSKSSTB6D,(, Y, , , , , , , , ,)
 DVASW = DVASW LXOR MSKSSLI ,(, , Y, , , , , , , ,)
 DVASW = DVASW LXOR MSKSSGNSS,(, , , Y, , , , , , ,)
 DVASW = DVASW LXOR MSKSSSBLO,(, , , , Y, , , , , ,)
 DVASW = DVASW LXOR MSKSSSBHI,(, , , , , Y, , , , ,)
 DVASW = DVASW LXOR MSKSSSBOM,(, , , , , , Y, , , ,)
 DVASW = DVASW LXOR MSKSST3A ,(, , , , , , , , Y, ,)
 SST2PTR = LOC'SSTGAIN' ,(Y, , , , , , , , , ,)
 SST2PTR = LOC'SSTTB6D' ,(, Y, , , , , , , , ,)
 SST2PTR = LOC'SSTALU' ,(, , Y, , , , , , , ,)
 SST1PTR = LOC'SSTGSS' ,(, , , Y, , , , , , ,)
 SST1PTR = LOC'SSTSBLO' ,(, , , , Y, , , , , ,)
 SST1PTR = LOC'SSTSBHI' ,(, , , , , Y, , , , ,)
 SST1PTR = LOC'SSTSBOM' ,(, , , , , , Y, , , ,)
 SST1PTR = LOC'SSTECSV' ,(, , , , , , , Y, , ,)
 SST1PTR = LOC'SSTECS1' ,(, , , , , , , , Y, ,)
 SST1PTR = LOC'SSTTB3A' ,(, , , , , , , , , Y,)
 VATR4 = 0 ,(Y, Y, , , , , , , , ,)
 .SSTUPD (= VATR4)''SS T UP'' ,(Y, Y, , , , , , , , ,)
 GOTO SS0201 ,(Y, Y, Y, , , , , , , ,)
 GOTO SS0230 ,(, , , Y, Y, Y, Y, Y, Y, Y)

ELSE GOTO SS0060 ''THIS POINT SHOULD NEVER BE REACHED LOGICALLY''
 END

SS0201.

FCL34 = 'INPRUG'
 FTADV = 'CLASS4'
 GOTO SS0235

SS0230.

VASPI = MSKSSCL3
 .SSTUPD (= VATRR) ''UPDATE SS TIME''
 FTADV = 'NORMAL'

SS0235.

.SS210 ''SET UP NEXT SS''
 FHST = 'TEST'
 GOTO SS0000

SS0060.

IF FASE EQ 'ALTERNATE'
 THEN FASE = 'NORMAL'
 UNLOCK ''RELEASE PREVIOUSLY ENABLED INTERRUPTS''

END

SPL KERNEL 10 SWITCH SELECTOR PROCESSING

```

MSS10. RETURN 'SWITCH SELECTOR COMMON RETURN'
      VASPI = 0
      VATRR = 0
      FCLS4 = 'NOTINPROG'
      DVASW = DVASW LAND MSKSSWV
      .EGP08 'RESCHEDULE TIMER 1 (NOT CODED)'
      FTADV = 'NORMAL'
      SST1PTR = SST1PTR(DTBID - 1)
SS1050. .SS210 'SET UP NEXT SS'
      IF FSSAC EQ 'ACTIVE' GOTO MSS20
      VSSW = KSSH1
      GOTO SS0235
MSS20. IF FSSIO EQ 'YES' WRITE SS,MSKSSRESET 'ISSUE FORCED RESET'
      FHST = 'NOTEST'
      DGSSM = 'SS05'
      .SSTUPQ (KSSB8) 'SCHEDULE SWITCH SELECTOR CHECK'
      VSSW = KSSH5
      GOTO SS0060
MSS30. FSSAC = 'ACTIVE'
      VSNA, VSNA1 = VSNA RSH 2 LAND MSKSSNA
      IF VSNA EQ 0
          THEN FSSAC = 'INACTIVE'
          .SS201 'ADVANCE SS TABLE, SET UP NEXT SS'
          GOTO SS0000
      END
      VSTG = VSNA LAND VPSTG
      IF VSTG EQ 0
          THEN FSSIO = 'NO'
          ELSE FSSIO = 'YES'
      END
      IF FHST EQ 'NOTEST' GOTO SS4000
      IF DFLT EQ 'REP' GOTO SS4000
      READ SSFB,TEMP 'READ SS FEEDBACK REGISTER'
      IF TEMP LAND MSKSSHS EQ 0 GOTO SS4000
      IF FSSIO EQ 'YES' WRITE SS,MSKSSRESET 'ISSUE RESET'
      DGSSM = 'SS40'
      .SSTUPQ (KSSB4) 'SCHEDULE STAGE AND ADDRESS ISSUANCE'
      VSSW = KSSB5
      GOTO SS0060
MSS40. FOR I = 22 WHILE I GR 0 'DELAY BEFORE ISSUING STAGE,ADDR'
      I = I - 1
      END
SS4000. IF FSSIO EQ 'YES' WRITE SS,VSNA 'ISSUE STAGE AND ADDRESS'
      DGSSM = 'SS50'
      .SSTUPQ (VSSW) 'SCHEDULE ADDRESS VERIFICATION'
      FOR I = 17 WHILE I GR 0 'DELAY FOR DOM TELEMETRY'
      I = I - 1
      END
      WRITE DOM 'OUTPUT SS AND DO REGS VIA DOM TELEMETRY'
      GOTO SS0060
MSS50. VSCCA = VSNA LXOR MSKSSHS
      VSSCA = VSCCA LAND MSKSSHS
      IF VSTG NQ 0
          THEN READ SSFB,TEMP 'READ SS FEEDBACK REGISTER'
          VSSFH = TEMP LAND MSKSSHS

```

SPL KERNEL 10 SWITCH SELECTOR PROCESSING

```

ELSE VSSF8 = VSSCA
END
IF VSSF8 NQ VSSCA GOTO SS5540
MSS55. IF VASPI LAND MSKSS4CO NQ 0
      THEN DFILE = DFILE LOR MSKFPSISSA
      DVSST = 1.E10
      RETURN 'MSS50, MSS55'

END
IF VSSRT EQ 0 GOTO MSS60
.SSTUPD (= DVTRB) 'UPDATE SS TIME
IF VSSRT = DVTRB LQ KSSRB GOTO MSS60
VSSTM = VSSRT - DVTGB - KSSRB
DVSST = VSSTM + DVTMR
DGSSM = 'SS60'
IF DFIL3 EQ 'INACTIVE' .EGP08 'RESCHED TIMER 1 (NOT CODED)'
RETURN 'MSS50, MSS55'
SS5540. IF VSSF8 EQ 0 AND VSSCA NQ MSKSSZFSF GOTO MSS55
      IF FSSIO EQ 'YES' WRITE SS,MSKSSRESET 'ISSUE RESET'
      DGSSM = 'SS80'
      .SSTUPQ (KSSB6) 'SCHEDULE COMPLEMENTED STAGE AND ADDRESS'
      TEMP = (VSSCA LXOR VSSF8) LSH 7
SS5570. IF TEMP LS 0 GOTO SS5580
      TEMP = TEMP LSH 1
      GOTO SS5570
SS5580. TEMP = TEMP LSH 1
      IF TEMP EQ 0 RETURN 'MSS50'
      DVMC4 = DVMC4 LOR MSKMC4SSCB
      IF FFBCH EQ 'CHANA'
        THEN FFBCH = 'CHANB'
        WRITE ICR,MSKICRSSCB 'SWITCH TO CHANNEL B'
        DVICR = DVICR LOR MSKICRSSCB

END
.UTR30 'DELAY FOR TELEMETRY AS REQUIRED'
WRITE TELSSFB,VSSF8 'TELEMETER SS FEEDBACK'
RETURN 'MSS50, MSS55'
MSS60. TEMP = VSTG LXOR MSKSSREAD
      IF FSSIO EQ 'YES' WRITE SS,TEMP 'ISSUE READ COMMAND'
      READ CLOCK,TEMP 'READ REAL TIME CLOCK FOR SS TELEMETRY'
      DGSSM = 'SS70'
      .SSTUPQ (KSSB2) 'SCHEDULE READ RESET'
      TEMP = VSNA LSH 2 LOR TEMP LAND MSKRTC
      .UTR30 'DELAY FOR TELEMETRY AS REQUIRED'
      WRITE TELSSSA,TEMP 'TELEMETER STAGE/ADDRESS AND READ TIME'
      IF DFACQ NQ 'GAIN' 'COMPRESS DATA BETWEEN STA. (NOT CODED)'
        THEN .MPC80 (DVDCT + MSKKS DCT) 'COMP TIME AND TAG'
        .MPC80 (VSNA RSH 3 + MSKSSDCS) 'COMP STAGE,ADD'

END
IF VASPI LAND MSKSS4CO NQ 0
      THEN VASPI = 0
      DFILE = DFILE LOR MSKFPCORD

END
CONDITIONS
VSNA1 EQ MSKSSHTG , (Y, , , , )
VSNA1 EQ MSKSSLOG , ( , Y, , , )
VSNA1 EQ MSKSSOMG , ( , , Y, , )

```

SPL KERNEL 10 SWITCH SELECTOR PROCESSING

```

        VSNA1 EQ MSKSSSIVR          ,( , , , Y, Y)
        FBRNI EQ 'FIRST'           ,( , , , Y, N)
ACTIONS
        DVMC7 = DVMC7 LOR MSKMC7HIG ,(Y, , , , )
        DVMC7 = DVMC7 LOR MSKMC7LOG ,( , Y, , , )
        DVMC7 = DVMC7 LOR MSKMC7OMG ,( , , Y, , )
        DVMC5 = DVMC5 LOR MSKMC54B1I,( , , , Y, )
        DVMC6 = DVMC6 LOR MSKMC68BRI,( , , , , Y)
ELSE RETURN
END
RETURN    ''MSS60''
MSS70.   IF FFSIO EQ 'YES'   WRITE SS,0  ''RESET READ COMMAND''
        DGSSM = 'SS05'
        .SSTUPQ (KSSR3)  ''SCHEDULE HUNG STAGE TEST''
        .SS201    ''ADVANCE SS TABLE, SET UP NEXT SS''
        VSSW = KSSB1
        IF VHSTW LAND VSTG EQ VSTG
            THEN FHST = 'NOTEST'
            ELSE FHST = 'TEST'
END
CONDITIONS
        VSNA1 EQ MSKSSWVO          ,(Y, , , )
        VSNA1 EQ MSKSSWVC          ,( , Y, , )
        VSNA1 EQ MSKSSSCC          ,( , , , Y)
ACTIONS
        DVASW = DVASW LXOR MSKSSSECS1 ,(Y, , , )
        DVASW = DVASW LXOR MSKSSSECSV ,( , Y, , )
        DFwV = 'OPEN'              ,(Y, , , )
        DFwV = 'CLOSE'             ,( , Y, , )
        DVDPM = DVDPM LOR MSKDIN9  ,( , , , Y)
ELSE RETURN    ''MSS70''
END
RETURN    ''MSS70''
MSS80.   VSNA = VSCCA
        IF FSSI0 EQ 'YFS'   WRITE SS,VSNA  ''ISSUE STAGE/COMP. ADDR.''
        DGSSM = 'SS55'
        .SSTUPQ (KSSR7)  ''SCHEDULE THE READ COMMAND''
        FOR I = 41 WHILE I GR 0  ''DELAY FOR DOM TELEMETRY''
            I = I - 1
END
WRITE DOM          ''OUTPUT SS AND DO REGS VIA DOM TELEMETRY''
RETURN    ''MSS80''
PROC
ENTRANCE .SS201    ''SS TABLE ADVANCE ROUTINE''
        .SS210    ''SS SETUP ROUTINE''
        IF FTADV EQ 'NORMAL'
            THEN SST1PTR = SST1PTR + 2      GOTO SS2020
            ELSE SST2PTR = SST2PTR + 2      GOTO SS2160
END
SS210.   IF FTADV EQ 'NORMAL'      GOTO SS2020
SS2160.  IF IND(SST2PTR) GQ 0      GOTO SS2070
        FCLS4 = 'NOTINPROG'
        DVMC6 = DVMC6 LXOR MSKMC6LUI
        DVMC7 = DVMC7 LXOR MSKMC7T6D
        GOTO SS2090
SS2020.  IF IND(SST1PTR) GQ 0      GOTO SS2030

```

SPL KERNEL 10 SWITCH SELECTOR PROCESSING

```

CONDITIONS
  VASPI LAND MSKSSSPEC NQ 0      ,(Y, , , , )
  VASPI LAND MSKSSCL3 NQ 0      ,( , Y, , , )
  VASPI LAND MSKSSCL1 NQ 0      ,( , , Y, , )
  FT60P EQ 'PASS1'              ,( , , , Y, N)

ACTIONS
  VASPI = MSKSSS4CO              ,(Y, , , , )
  VASPI = VSC3(1)                ,( , Y, , , )
  VASPI = VSC1(1)                ,( , , Y, , )
  VASPI = U                      ,( , , , Y, Y)
  VATRR = VSC3(2)                ,( , Y, , , )
  VATRR = VSC1(2)                ,( , , Y, , )
  VATRR = U                      ,( , , , Y, Y)
  DVASW = DVASW LAND MSKSSWV    ,(Y, , , , )
  FT60P = 'PASS2'                ,( , , , Y, )
  SST1PTR = LOC'SSTSIVB'         ,(Y, , , , )
  SST1PTR = VSC3(0)              ,( , Y, , , )
  SST1PTR = VSC1(0)              ,( , , Y, , )
  SST1PTR = LOC'SSTTB5A'         ,( , , , Y, )
  SST1PTR = LOC'SSTTB5B'         ,( , , , , Y)
ELSE GOTO SS2020 'THIS POINT SHOULD NEVER BE REACHED LOGICALLY'
END
GOTO SS2020
SS2030. IF FCLS4 EQ 'INPRG'      GOTO SS2070
SS2040. VSSRT = IND(SST1PTR)*KCSR + VATRR
        VSNA = IND(SST1PTR,1)
SS2050. VHSTM = VSNA RSH 2 LAND MSKSSSSB
        RETURN 'SS201, SS210, SS2110, SS2160'
SS2070. IF IND(SST1PTR)*KCSR+VATRR-KSSCK GQ IND(SST2PTR)*KCSR+VATR4
        THEN FTADV = 'CLASS4'
         VSSRT = IND(SST2PTR)*KCSR + VATR4
         VSNA = IND(SST2PTR,1)
         GOTO SS2050
SS2090. ELSE FTADV = 'NORMAL'
         GOTO SS2040

END
EXIT 'SS201,SS210'
PRCG .SSTUPD (= TIME) 'UPDATE SWITCH SELECTOR TIME'
ITEM TIME FIXED -2
ENDDATA
READ CLOCK,TEMP 'READ REAL TIME CLOCK'
TIME, DVTRB = DVTGR + DVTRR + (TEMP - DVRTC LAND MSKRTC) RSH 2
EXIT 'SSTUPD'
PRCG .SSTUPQ (BIAS) 'UPDATE SS TIME AND SCHEDULE SS FUNCT.'
ITEM BIAS FIXED -2
ENDDATA
READ CLOCK,TEMP 'READ REAL TIME CLOCK'
DVTRB = DVTGR + DVTRR + (TEMP - DVRTC LAND MSKRTC) RSH 2
VSSTM = BIAS + DVTRR + (TEMP - DVRTC LAND MSKRTC) RSH 2
DVSST = VSSTM + DVTMR
IF DFIL3 EQ 'INACTIVE' .EGP08 'RESCHED TIMER 1 (NOT CODED)'
EXIT 'SSTUPQ'
PRCG .EGP08 EXTERNAL EXIT
PRCG .MPC80 EXTERNAL EXIT
TERM 'MSS00'

```

SPL KERNEL 11 ATM TASK KEYING

```

START .TASKKEY (PRIORITY,TSKPTR) ''ATM TASK KEYING ROUTINE''
      ITEM PRIORITY INTEGER ''PRIORITY LEVEL OF TASK BEING KEYED''
      ITEM TSKPTR LOCATION ''POINTER TO TASK BEING KEYED''
      ITEM CHAIN LOCATION ''OVERFLOW CHAIN POINTER''
      ITEM SLOT LOCATION CONSTANT = LOC'ATMPOVFT'
''
''PRIORITY CONTROL TABLE CONTAINS ONE ENTRY FOR EACH SYSTEM ''
''PRIORITY LEVEL. FOR EACH ENTRY THERE ARE FIVE ITEMS. ''
'' 1. LOCATION POINTER TO THE NEXT EXECUTABLE INSTRUCTION ''
'' OF THE TASK CURRENTLY ASSIGNED TO THAT PRIORITY LEVEL ''
'' OR ZERO IF NO TASK IS CURRENTLY ASSIGNED. ''
'' 2. ) ''
'' 3. ) TASK REGISTER CONTENTS (INITIALLY SET TO ZERO). ''
'' 4. ) ''
'' 5. POINTER TO THE BEGINNING OF THE PRIORITY OVERFLOW ''
'' TABLE CHAIN FOR THAT PRIORITY LEVEL. A VALUE OF ZERO ''
'' INDICATES END OF CHAIN. ''
''
'' NOTE THE NUMBER OF REGISTERS SAVED FOR A TASK WAS ARBITRARI-''
'' LY CHOSEN FOR THIS EXAMPLE AND MAY BE ADJUSTED AS ''
'' REQUIRED. ''
''
      TABLE ATMPCT 10 SERIAL 5
            (ATMTSKPTR LOCATION,
             ATMTSKREG1 LOGICAL,
             ATMTSKREG2 LOGICAL,
             ATMTSKREG3 LOGICAL,
             ATMVFPTTR LOCATION)
''
''THE PRIORITY OVERFLOW TABLE IS USED FOR KEYING TASKS ON A ''
''PRIORITY LEVEL WHICH IS CURRENTLY ASSIGNED TO ANOTHER TASK. ''
''THE ENTRIES ARE NOT ALLOCATED TO A FIXED PRIORITY BUT ARE ''
''ASSIGNED DYNAMICALLY AS REQUIRED. ALL OVERFLOW ENTRIES FOR ''
''EACH PRIORITY LEVEL ARE CHAINED TOGETHER SUCH THAT THE TASKS ''
''CAN BE EXECUTED ON A FIRST-IN-FIRST-OUT BASIS. EACH ENTRY ''
''CONSISTS OF TWO ITEMS. ''
'' 1. POINTER TO NEXT ENTRY IN THE CHAIN. A VALUE OF ZERO ''
'' INDICATES END OF CHAIN. ''
'' 2. LOCATION POINTER TO THE BEGINNING OF THE TASK FOR ''
'' THAT ENTRY. A VALUE OF ZERO INDICATES THAT THE ENTRY ''
'' IS CURRENTLY NOT ASSIGNED TO ANY TASK. ''
''
      TABLE ATMPOVFT 25 SERIAL 2
            (ATMVFPTTR LOCATION,
             ATMTSKPTR LOCATION)
ENDDATA LOCK ''INHIBIT ALL INTERRUPTS''
''
''IF THE REQUESTED PRIORITY LEVEL IS NOT CURRENTLY ASSIGNED, ''
''INITIALIZE THE ENTRY FOR THIS TASK. ''
''
      IF ATMPCT'ATMTSKPTR(PRIORITY) EQ 0
          THEN ATMPCT'ATMTSKPTR(PRIORITY) = TSKPTR
              ATMTSKREG1(PRIORITY) ,
              ATMTSKREG2(PRIORITY) ,

```


SPL KERNEL 11 ATM TASK KEYING

```

                                ATMTSKREG3(PRIORITY) = 0
                                ''
                                '' OTHERWISE, SEARCH FOR THE END OF THE OVERFLOW POINTER CHAIN. ''
                                ''
                                ELSE CHAIN = LOC'ATMPCT'ATMOVFPTR(0)'+ 5*PRIORITY
CHNSRCH.                        IF IND(CHAIN) NE 0
                                THEN CHAIN = IND(CHAIN)
                                GOTO CHNSRCH
                                ''
                                '' WHEN THE END OF THE OVERFLOW POINTER CHAIN HAS BEEN ''
                                '' FOUND, SEARCH FOR AN EMPTY SLOT IN THE OVERFLOW TABLE. ''
                                ''
                                ELSE FOR I = 0 BY 2 UNTIL 50
                                IF IND(SLOT, I+1) EQ 0 GOTO SLTFND
                                END WAIT ''STOP IF OVERFLOW TABLE FULL''
                                ''
                                '' ADD THIS ENTRY TO THE END OF THE OVERFLOW POINTER CHAIN AND ''
                                '' STORE THE TASK POINTER IN IT. ''
                                ''
SLTFND.                        IND(CHAIN) = SLOT + I
                                IND(SLOT, I) = 0
                                IND(SLOT, I+1) = TSKPTR
                                END
                                END
                                UNLOCK
                                RETURN ''TASKKEY''
TERM                            ''TASKKEY''

```

CLASP COMMON DATA DECLARATIONS

```
START      ' CLASP KERNELS '
DECLARE FIXED,
DKRTCSEC      10 CONSTANT = 4063.492,
DKMIR         0 CONSTANT = .040*DKRTCSEC,
DKTD          0 CONSTANT = .00413*DKRTCSEC,
DKT1          10 ,
DLPRL(3) -2  CONSTANT=(50.*DKRTCSEC,60.*DKRTCSEC,100.*DKRTCSEC)X
DLPTL (3) -2 ,
DLTTL (12) -2 ,
DVACT         0 ,
DVA1          4 ,
DVA2          4 ,
DVA3          4 ,
DVA4          4 ,
DVA5          4 ,
DVA6          4 ,
DVCA (3) 25 ,
DVCC (3) 25 ,
DVDC (3) 13 ,
DVDA (3) 7 ,
DVDR (3) 7 ,
DVDC (3) 25 ,
DVDM (3) 11 ,
DVDT          23 ,
DVEOF        25 ,
DVERT         0 ,
DVf (3) 7 ,
DVfMC        19 ,
DVfQM        19 ,
DVfOR        -1 ,
DVg (3) 21 ,
DVmAS        3 ,
DVmFR        5 ,
DVmLD         0 ,
DVmLR        19 ,
DVmLT        -2 ,
DVm05        0 ,
DVm06        0 ,
DVPTG        -2 ,
DVRC (3) 7 ,
DVRTC        0 ,
DVSSST       -2 ,
DVtAS        10 ,
DVtB         10 ,
DVtD         0 ,
DVtEX        0 ,
DVtGB        -2 ,
DVtH (3) 25 ,
DVtI         10 ,
DVtMM        -2 ,
DVtMR        -2 ,
DVtRB        -2 ,
DVtRR        -2 ,
DVtRS        -2 ,
DVtT1        0 ,
```

CLASP COMMON DATA DECLARATIONS

| | | | | |
|------------------|------|---|------------------------|---|
| DVVSQ | 3 | , | | X |
| DV1MR | 25 | , | | X |
| DV2TG | -2 | , | | X |
| TEMP | TEMP | , | | X |
| TEMP1 | TEMP | , | | |
| DECLARE INTEGER, | | | 'NUMERIC INTEGER DATA' | X |
| DFLT | | , | | X |
| DGMLM | | , | | X |
| DGSSM | | , | | X |
| DGST2 | | , | | X |
| DQST2 | | , | | X |
| DTBID | | , | | X |
| DVDGS | | , | | X |
| DVHDA | | , | | X |
| DVHDB | | , | | X |
| DVLRC | | , | | X |
| DVP | | , | | X |
| DVRE (3) | | , | | X |
| EPTINDX | | , | | X |
| GST1M | | | | |
| DECLARE BOOLEAN, | | | | X |
| APSTAT | | , | | X |
| ARSTAT | | , | | X |
| CC1STAT | | , | | X |
| CSSTAT | | , | | X |
| CTSTAT | | , | | X |
| DFACQ | | , | | X |
| DFDBF | | , | | X |
| DFDTL | | , | | X |
| DFIL1 | | , | | X |
| DFIL2 | | , | | X |
| DFIL3 | | , | | X |
| DFPHC | | , | | X |
| DFSMC | | , | | X |
| DFTBCER | | , | | X |
| DFTUP | | , | | X |
| DFWV | | , | | X |
| DFZER | | , | | X |
| DKAPI (4) | | , | | X |
| DTSTAT | | , | | X |
| DPSTAT | | , | | X |
| DVSTAT | | , | | X |
| ERSTAT | | , | | X |
| HSSTAT | | , | | X |
| IGSTAT | | , | | X |
| MSSTAT | | , | | X |
| NESTAT | | , | | X |
| OGSTAT | | , | | X |
| PASTAT | | , | | X |
| PGSTAT | | , | | X |
| PPSTAT (3) | | , | | X |
| RSSTAT | | , | | X |
| SASTAT | | , | | X |
| TB1STAT | | , | | X |
| TB57STAT | | , | | X |

CLASP COMMON DATA DECLARATIONS

| | | | |
|------------------|---|------------------------|---|
| TCSTAT | , | | X |
| TGSTAT | , | | X |
| TTSTAT | , | | X |
| T2STAT (11) | | | |
| DECLARE INTEGER, | | 'LOGICAL INTEGER DATA' | X |
| DFILE | , | | X |
| DFMDI | , | | X |
| DVAC (3) | , | | X |
| DVASW | , | | X |
| DVDPM | , | | X |
| DVEMR | , | | X |
| DVICR | , | | X |
| DVIH | , | | X |
| DVLDB | , | | X |
| DVMC4 | , | | X |
| DVMC5 | , | | X |
| DVMC6 | , | | X |
| DVMC7 | , | | X |
| MSKABLAD | | CONSTANT =0'000001000' | X |
| MSKACCA | | CONSTANT =0'777700000' | X |
| MSKACCR | | CONSTANT =0'000017776' | X |
| MSKDIN9 | | CONSTANT =0'000004000' | X |
| MSKDCSCH | | CONSTANT =0'774000000' | X |
| MSKDCSDO | | CONSTANT =0'000040000' | X |
| MSKDCSER | | CONSTANT =0'000077776' | X |
| MSKDCSMC | | CONSTANT =0'770000000' | X |
| MSKDCSMD | | CONSTANT =0'000000020' | X |
| MSKDCSSB | | CONSTANT =0'004000000' | X |
| MSKDCSTR | | CONSTANT =0'200000000' | X |
| MSKEMRLB | | CONSTANT =0'000001000' | X |
| MSKERTAG | | CONSTANT =0'000070000' | X |
| MSKFMREP | | CONSTANT =0'000100000' | X |
| MSKFPSCR | | CONSTANT =0'100000000' | X |
| MSKFPSI2 | | CONSTANT =0'000040000' | X |
| MSKFPSIS | | CONSTANT =0'001000000' | X |
| MSKGIMA | | CONSTANT =0'377700000' | X |
| MSKICRRG | | CONSTANT =0'000000020' | X |
| MSKICRCA | | CONSTANT =0'000040000' | X |
| MSKICRSB | | CONSTANT =0'000010000' | X |
| MSKICRSQ | | CONSTANT =0'000002000' | X |
| MSKINT | | CONSTANT =0'157740000' | X |
| MSKM180D | | CONSTANT =0'377777776' | X |
| MSKM4AMF | | CONSTANT =0'000000100' | X |
| MSKM4SSB | | CONSTANT =0'000001000' | X |
| MSKM54B1 | | CONSTANT =0'000000100' | X |
| MSKM6D04 | | CONSTANT =0'000000400' | X |
| MSKM6LUI | | CONSTANT =0'000010000' | X |
| MSKM6T6A | | CONSTANT =0'000002000' | X |
| MSKM6T6B | | CONSTANT =0'000000010' | X |
| MSKM6T6C | | CONSTANT =0'000000040' | X |
| MSKM68RR | | CONSTANT =0'400000000' | X |
| MSKM7HIG | | CONSTANT =0'004000000' | X |
| MSKM7LOG | | CONSTANT =0'010000000' | X |
| MSKM7OMG | | CONSTANT =0'020000000' | X |
| MSKM7T6D | | CONSTANT =0'100000000' | X |

CLASP COMMON DATA DECLARATIONS

| | | | |
|----------|----------|----------------|---|
| MSKRTC | CONSTANT | =0'000037776', | X |
| MSKRTCRC | CONSTANT | =0'007777540', | X |
| MSKSCCO | CONSTANT | =0'000100000', | X |
| MSKSSACQ | CONSTANT | =0'000000004', | X |
| MSKSSCL1 | CONSTANT | =0'040000000', | X |
| MSKSSCL3 | CONSTANT | =0'100000000', | X |
| MSKSSCS1 | CONSTANT | =0'000003770', | X |
| MSKSSCS3 | CONSTANT | =0'077774000', | X |
| MSKSSDCS | CONSTANT | =0'500000000', | X |
| MSKSSDCT | CONSTANT | =0'405400000', | X |
| MSKSSECV | CONSTANT | =0'002000000', | X |
| MSKSSEC1 | CONSTANT | =0'001000000', | X |
| MSKSSEND | CONSTANT | =0'37777776', | X |
| MSKSSGNS | CONSTANT | =0'040000000', | X |
| MSKSSHIG | CONSTANT | =0'100720000', | X |
| MSKSSHS | CONSTANT | =0'003770000', | X |
| MSKSSLI | CONSTANT | =0'000000002', | X |
| MSKSSLOG | CONSTANT | =0'100520000', | X |
| MSKSSOMG | CONSTANT | =0'100070000', | X |
| MSKSSRD | CONSTANT | =0'400000000', | X |
| MSKSSRS | CONSTANT | =0'200000000', | X |
| MSKSSSBH | CONSTANT | =0'010000000', | X |
| MSKSSSBL | CONSTANT | =0'020000000', | X |
| MSKSSSBO | CONSTANT | =0'004000000', | X |
| MSKSSSCC | CONSTANT | =0'100310000', | X |
| MSKSSSNA | CONSTANT | =0'135770000', | X |
| MSKSSSPC | CONSTANT | =0'200000000', | X |
| MSKSSSSB | CONSTANT | =0'174000000', | X |
| MSKSSS4B | CONSTANT | =0'020230000', | X |
| MSKSSS4C | CONSTANT | =0'400000000', | X |
| MSKSSS41 | CONSTANT | =0'000000400', | X |
| MSKSST3A | CONSTANT | =0'000400000', | X |
| MSKSST6 | CONSTANT | =0'004000000', | X |
| MSKSST6A | CONSTANT | =0'000002000', | X |
| MSKSST6B | CONSTANT | =0'000001000', | X |
| MSKSST6C | CONSTANT | =0'100000000', | X |
| MSKSST6D | CONSTANT | =0'000200000', | X |
| MSKSSWV | CONSTANT | =0'003000000', | X |
| MSKSSWVC | CONSTANT | =0'101050000', | X |
| MSKSSWVO | CONSTANT | =0'101450000', | X |
| MSKSSZFS | CONSTANT | =0'002000000', | X |
| MSKTMC0 | CONSTANT | =0'700000000', | X |
| MSKTMC1 | CONSTANT | =0'710000000', | X |
| MSKTMC2 | CONSTANT | =0'720000000', | X |
| MSKTMC3 | CONSTANT | =0'730000000', | X |
| MSKTMC4 | CONSTANT | =0'740000000', | X |
| MSKT2INT | CONSTANT | =0'100000000', | X |
| MSK180DG | CONSTANT | =0'400000000', | X |
| VTMC | | | X |
| VTOLD | | | X |

CLASP UTILITY ROUTINES

```

PROC      .UTR00      ''TELEMETRY DELAY FOR MODE REG SETTING OF 70''
          VTMC = MSKTMCO
          .UTR0      ''PERFORM DELAY AS RQUIRED''
EXIT
PROC      .UTR01      ''TELEMETRY DELAY FOR MODE REG SETTING OF 71''
          VTMC = MSKTMCO
          .UTR0      ''PERFORM DELAY AS REQUIRED''
EXIT
PROC      .UTR02      ''TELEMETRY DELAY FOR MODE REG SETTING OF 72''
          VTMC = MSKTMCO
          .UTR0      ''PERFORM DELAY AS REQUIRED''
EXIT
PROC      .UTR03      ''TELEMETRY DELAY FOR MODE REG SETTING OF 73''
          VTMC = MSKTMCO
          .UTR0      ''PERFORM DELAY AS REQUIRED''
EXIT
PROC      .UTR04      ''TELEMETRY DELAY FOR MODE REG SETTING OF 74''
          VTMC = MSKTMCO
          .UTR0      ''PERFORM DELAY AS REQUIRED''
EXIT
PROC      .UTR0      ''TELEMETRY DELAY FOR LEVEL 0''
          DECLARE FIXED 0,
                VTIM,
                KTELBIAS CONSTANT = 2.
TR00.    LOCK T1INT,T2INT,EX1INT,EX2INT,EX3INT,EX4INT,EX5INT,EX6INT,  X
                EX7INT,EX8INT,EX9INT
          DIRECT
                ''READ CLOCK INTO VTIM''
          END
          IF VTIM = DVTD LAND MSKRTC GQ DKTD      THEN GOTO TR05      END
          UNLOCK ''RELEASE PREVIOUSLY ENABLED INTERRUPTS''
          ''ALLOW HIGH PRIORITY TASKS TO INTERRUPT''
          GOTO TR00
TR05.    DIRECT
                ''SET MODE REG WITH CONTENTS OF VTMC''
          END
          DVTD = VTIM + KTELBIAS
EXIT
PROC      .UTR30      ''TELEMETRY DELAY FOR INTERRUPT LEVEL 3''
          DECLARE FIXED 0,
                KTELBIAS CONSTANT = 2. ,
                VTIM
TR35.    DIRECT
                ''READ CLOCK INTO VTIM''
          END
          IF VTIM = DVTD LAND MSKRTC LS DKTD      THEN GOTO TR35      END
          DIRECT
                ''SET MODE REG WITH MSKTMCO''
          END
          DVTD = VTIM + KTELBIAS
EXIT
PROC      .UTR24      ''TELEMETRY DELAY FOR INTERRUPT LEVEL 2''
          DECLARE FIXED 0,
                KTELBIAS CONSTANT = 2. ,
                VTIM

```

CLASP UTILITY ROUTINES

```
TR20.  LOCK T1INT,T2INT,EX1INT,EX2INT,EX3INT,EX4INT,EX5INT,EX6INT,  X
        EX7INT,EX8INT,EX9INT
        DIRECT
        ''READ CLOCK INTO VTIM''
        END
        IF VTIM = DVTD LAND MSKRTC GG DKTD THEN GOTO TR25      END
        UNLOCK T1INT
        GOTO TR20
TR25.  DIRECT
        ''SET MODE REG WITH MSKTC4''
        END
        DVTD = VTIM + KTELBIAS
EXIT   ''UTR24''
```

CLASP KERNEL 1 INITIALIZATION

```

''SYSTEM INITIALIZATION''
EGP0.    LOCK TLCINT T1INT,T2INT,EX1INT,EX2INT,EX3INT,EX4INT,EX5INT,    X
          EX6INT,EX7INT,EX8INT,EX9INT
          DIRECT
            ''READ X ACCELEROMETER INTO VOAC(0)''
            ''READ Y ACCELEROMETER INTO VOAC(1)''
            ''READ Z ACCELEROMETER INTO VOAC(2)''
            ''READ REAL TIME CLOCK INTO DVACT''
          END
          IF DFMDI LAND MSKFMREP
            THEN ON T1INT
              DIRECT
                ''READ REAL TIME CLOCK INTO DVACT''
              END
              LOCK T1INT
              TEMP = 0
              END
              TEMP = 1
              DIRECT
                ''LOAD TIMER 1 WITH 1 BIT''
              END
              UNLOCK T1INT
              IF TEMP            THEN GOTO LOOP.                            END
          END
          DFIL1,DFIL2,DFIL3 = TRUE
          DVRTC,DVTEX,VPPOT = DVACT
          DVTMM,DVTRR,DVERT,DVTGR,DVTRS,DVTMR,DTRID,VTD = 0.
          .EGP1 ''ACTIVATE INTERRUPT PROCESSOR CHRONIC STATEMENTS''
          UNLOCK TLCINT
          FGNC = FALSE
          DVSST = 1.E10
          DVMLT, DVMLD = DKMIR
          .EGP15 ''SCHEDULE FIRST TIMER 1 FUNCTION''
          DVP = 1
          .GP002 ''PASS CONTROL TO PHASE ACTIVATION ROUTINE''
PROC .MPA00 ''PHASE TERMINATION ROUTINE''
          DFPHC = TRUE
          LOCK TLCINT,T1INT,T2INT,EX1INT,EX2INT,EX3INT,EX4INT,EX5INT,    X
              EX6INT,EX7INT,EX8INT,EX9INT
          DIRECT
            ''LOAD TIMER 2 WITH A LARGE VALUE''
            ''RESET ANY PENDING TIMER 2 INTERRUPT''
          END
          DFIL1, DFIL2 = TRUE
          UNLOCK TLCINT,T1INT
          .GP002 ''ACTIVATE THE NEXT MISSION PHASE''
EXIT
PROC .GP002 ''MISSION PHASE ACTIVATION AND CONTROL ROUTINE''
          DECLARE BOOLEAN, FGNC
          DECLARE FIXED, VTD 10
GP0020.    IF DVP GR 4    THEN STOP                                            END
          IF DKAPI (DVP - 1)
            THEN .EGP20 ''START PHASE TIME REFERENCE''
                  GOTO (INP13, INP24, INP13, INP24, *) DVP - 1
            ELSE DVP = DVP + 1

```


CLASP KERNEL 1 INITIALIZATION

GOTU GP0020

```

END
INP13.  ARSTAT,APSTAT,DPSTAT,NESTAT,CC1STAT,MSSTAT,EBSTAT = TRUE
        SASTAT,DVSTAT,TCSTAT,PASTAT,TTSTAT,IGSTAT,HSSTAT,OGSTAT,      X
        TGSTAT,RSSTAT,CSSTAT,TB1STAT,TB57STAT,PGSTAT = FALSE
        DLPTL(*) = 0.
        PPSTAT(*) = FALSE
        T2STAT(*) = FALSE
        T2STAT(0) = TRUE
        MIN00 ''PERFORM PHASE 1/3 APPLICATION PGM INIT (NOT CODED)''
        .EGP18 ''SCHEDULE NEXT TIMER 2 FUNCTION''
        DFIL1, DFIL2, DFIL3, DFPHC = FALSE
        UNLOCK ''RELEASE PREVIOUSLY ENABLED INTERRUPTS''
        .NINTSEQ1 ''PASS CONTROL TO PHASE 1/3 NON-INTERRUPT SEQ''
INP24.  CTSTAT,DTSTAT = FALSE
        DLPTL(*) = 0.
        PPSTAT(*) = TRUE
        T2STAT(0),T2STAT(4),T2STAT(5) = FALSE
        T2STAT(1),T2STAT(2),T2STAT(3),T2STAT(6),T2STAT(7),T2STAT(8),  X
        T2STAT( 9),T2STAT(10) = TRUE
        .MIN10 ''PERFORM PHASE 2/4 APPLICATION PGM INIT (NOT CODED)''
        .EGP18 ''SCHEDULE NEXT TIMER 2 FUNCTION''
        DFIL1,DFIL2,DFIL3, DFPHC = FALSE
        UNLOCK ''RELEASE PREVIOUSLY ENABLED INTERRUPTS''
        .NINTSEQ2 ''PASS CONTROL TO PHASE 2/4 NON-INTERRUPT SEQ''
EXIT    ''GP002''

```

CLASP KERNEL 2 INTERRUPT PROCESSING

```

PROC .EGP1          ''INTERUPT PROCESSOR''
    OPTIMIZE TIME(20)
    ''
    ''RESPONSE FOR TLC INTERRUPT
    ''
    ON TLCINT
    LOCK T1INT,T2INT,EX1INT,EX2INT,EX3INT,EX4INT,EX5INT,EX6INT,  X
        EX7INT,EX8INT,EX9INT
    DIRECT
        ''READ REAL TIME CLOCK INTO DVTEX''
    END
    DFIL2, DFIL3 = TRUE
    .MTS00 ''PROCESS TLC INTERRUPT (NOT CODED)''
    ''THE TLC APPLICATION PROGRAM DOES NOT RETURN CONTROL
    END
    ''
    ''RESPONSE FOR TIMER 1 INTERRUPT
    ''
    ON T1INT
    LOCK T1INT,T2INT,EX1INT,EX2INT,EX3INT,EX4INT,EX5INT,EX6INT,  X
        EX7INT,EX8INT,EX9INT
    DIRECT
        ''READ REAL TIME CLOCK INTO DVTT1''
    END
    DFIL3 = TRUE
    GOTO ( ,GP11,GP12) GST1M
    .MML00 ''FLIGHT SIM MINOR LOOP'' $ GOTO EGP11
GP11.    .MML20 ''NORMAL MINOR LOOP'' $ GOTO EGP11
GP12.    .MSS00 ''SWITCH SELECTOR PROCCESOR''
FGP11.   .EGP15 ''SCHEDULE NEXT T1 FUNCTION''
    DFIL3 = FALSE
    UNLOCK '' RELEASE PREVIOUSLY ENABLED INTERRUPTS''
    END
    ''
    ''RESPONSE FOR TIMER 2 INTERRUPT
    ''
    ON T2INT
    LOCK T2INT
    DFIL1 = TRUE
    GOTO (EGP12,GP21,GP22,GP23,GP24,GP25,GP26,GP27,GP28,GP29,GP30,X
        GP31) DGST2
GP21.    .MUM00 ''TIME UPDATE (NOT CODED)'' $ GOTO EGP12
GP22.    .MLR10 ''LADDER RAMP PROCESSOR (NOT CODED)'' $ GOTO EGP12
GP23.    .MEP00 ''EVENTS PROCESSOR'' $ GOTO EGP12
GP24.    .MTT10 ''TIME TILT GUIDANCE (NOT CODED)'' $ GOTO EGP12
GP25.    .MNU00 ''NAVIGATION UPDATE IMPL (NOT CODED)'' $ GOTO EGP12
GP26.    .MEE00 ''TIME BASE 8 ENABLE (NOT CODED)'' $ GOTO EGP12
GP27.    .MCM00 ''PHASE 2/4 CONTROL MOD (NOT CODED)'' $ GOTO EGP12
GP28.    .MCM10 ''PHASE 2/4 CONTROL MOD (NOT CODED)'' $ GOTO EGP12
GP29.    .MCM20 ''PHASE 2/4 CONTROL MOD (NOT CODED)'' $ GOTO EGP12
GP30.    .MEPWM ''WATER METHANOL ACTIVATE(NOT CODED)'' $ GOTO EGP12
GP31.    .MER00 ''EXTRA ACCELEROMETER RD (NOT CODED)''
EGP12.   .EGP18 ''SCHEDULE NEXT T2 FUNCTION''
    DFIL1 = FALSE
    UNLOCK T2INT

```

CLASP KERNEL 2 INTERRUPT PROCESSING

```

END
''
''RESPONSE FOR EXTERNAL 2 INTERRUPT
''
ON EX2INT
LOCK T1INT,T2INT,EX1INT,EX2INT,EX3INT,EX4INT,EX5INT,EX6INT, X
EX7INT,EX8INT,EX9INT
DIRECT
''READ REAL TIME CLOCK INTO DVTEX''
END
DFIL2, DFIL3 = TRUE
.MDP28 ''SC INITIATION OF S2/S4B SEPARATION (NOT CODED)''
DFIL2, DFIL3 = FALSE
UNLOCK ''RELEASE PREVIOUSLY ENABLED INTERRUPTS''
END
''
''RESPONSE FOR EXTERNAL 4 INTERRUPT
''
ON EX4INT
LOCK T1INT,T2INT,EX1INT,EX2INT,EX3INT,EX4INT,EX5INT,EX6INT, X
EX7INT,EX8INT,EX9INT
DIRECT
''READ REAL TIME CLOCK INTO DVTEX''
END
DFIL2, DFIL3 = TRUE
.MTB50 ''S4B ENGINE OUT (NOT CODED)''
DFIL2, DFIL3 = FALSE
UNLOCK ''RELEASE PREVIOUSLY ENABLED INTERRUPTS''
END
''
''RESPONSE FOR EXTERNAL 5 INTERRUPT
''
ON EX5INT
LOCK T1INT,T2INT,EX1INT,EX2INT,EX3INT,EX4INT,EX5INT,EX6INT, X
EX7INT,EX8INT,EX9INT
DIRECT
''READ REAL TIME CLOCK INTO DVTEX''
END
DFIL2, DFIL3 = TRUE
.MTB30 ''S1C OUTBOARD ENGINE OUT (NOT CODED)''
DFIL2, DFIL3 = FALSE
UNLOCK ''RELEASE PREVIOUSLY ENABLED INTERRUPTS''
END
''
''RESPONSE FOR EXTERNAL 6 INTERRUPT
''
ON EX6INT
LOCK T1INT,T2INT,EX1INT,EX2INT,EX3INT,EX4INT,EX5INT,EX6INT, X
EX7INT,EX8INT,EX9INT
DIRECT
''READ REAL TIME CLOCK INTO DVTEX''
END
DFIL2, DFIL3 = TRUE
.MTB40 ''S2 PROPELLANT DEPLETION (NOT CODED)''
DFIL2, DFIL3 = FALSE

```

CLASP KERNEL 2 INTERRUPT PROCESSING

```

UNLOCK ''RELEASE PREVIOUSLY ENABLED INTERRUPTS''
END
''
''RESPONSE FOR EXTERNAL 8 INTERRUPT
''
ON EX8INT
LOCK T1INT,T2INT,EX1INT,EX2INT,EX3INT,EX4INT,EX5INT,EX6INT, X
EX7INT,EX8INT,EX9INT
DIRECT
''READ REAL TIME CLOCK INTO DVTEX''
END
DFIL2, DFIL3 = TRUE
.MDS00 ''PROCESS DIGITAL COMMAND SYSTEM INPUT''
DFIL2, DFIL3 = FALSE
UNLOCK ''RELEASE PREVIOUSLY ENABLED INTERRUPTS''
END
EXIT ''EGP1''
PROC .EGP15 ''TIMER 1 SCHEDULER''
DECLARE FIXED, KT1BIAS 0 CONSTANT = 9.
DIRECT
''READ REAL TIME CLOCK INTO TEMP''
END
TEMP1 = DVTMM + ((TEMP - DVRTC LAND MSKRTC) + KT1BIAS) RSH 2
IF DVMLT LQ TEMP1
THEN TEMP = 1
GST1M = DGMLM
GOTO EGP150
END
IF DVMLT LQ DVSST
THEN GST1M = DGMLM
TEMP = (DVMLT - TEMP1) LSH 1
ELSE GST1M = 2
IF DVSST LQ TEMP1
THEN TEMP = 1
ELSE TEMP = (DVSST - TEMP1) LSH 1
END
END
EGP150. DIRECT
''LOAD TIMER 1 WITH TEMP''
END
EXIT ''EGP15''
PROC .EGP18 ''TIMER 2 SCHEDULER''
DECLARE FIXED, KT2BIAS 0 CONSTANT = 12.0, X
K4SEC -2 CONSTANT = 4.*DKRTCSEC
DGST2 = 0
DV2TG = DVTMM + K4SEC
FOR I = 0 TO 10
IF NOT T2STAT(I) THEN GOTO T2S10 END
IF DLTTL(I) GR DV2TG THEN GOTO T2S10 END
DGST2 = I + 1
DV2TG = DLTTL(I)
T2S10. END
LOCK T1INT,EX1INT,EX2INT,EX3INT,EX4INT,EX5INT,EX6INT,EX7INT, X
EX8INT,EX9INT
IF DVT2G LQ DVTMM THEN GOTO T2S20 END

```

CLASP KERNEL 2 INTERRUPT PROCESSING

```

DIRECT
    'READ REAL TIME CLOCK INTO TEMP''
END
TEMP = (DV2TG - DVTMM + DVERT - (TEMP - DVRTC LAND MSKRTC)    X
        - KT2BIAS) RSH 1
IF TEMP LQ 0
T2S20.     THEN TEMP = 1
END
DIRECT
    'LOAD TIMER 2 WITH TEMP''
END
UNLOCK 'RELEASE PREVIOUSLY ENABLED INTERRUPTS''
EXIT      'EGP18''
UNTIME
PROC .EGP20 'SYSTEM TIME UPDATE ROUTINE''
LOCK T1INT,T2INT,EX1INT,EX2INT,EX3INT,EX4INT,EX5INT,EX6INT,    X
        EX7INT,EX8INT,EX9INT
DIRECT
    'READ REAL TIME CLOCK INTO TEMP1''
END
DVERT = TEMP1 - DVRTC LAND 3
DVTMM = DVTMM + (TEMP1 - DVRTC LAND MSKRTC)
DVRTC = TEMP1 - DVERT
DVTRR = DVTMM - DVTMR
IF DFIL3
    THEN GOTO OUT
END
IF DFIL2
    THEN UNLOCK T1INT
        GOTO OUT
END
IF DFIL1
    THEN UNLOCK 'RELEASE PREVIOUSLY ENABLED INT. EXCEPT T2''
        GOTO OUT
END
UNLOCK 'RELEASE PREVIOUSLY ENABLED INTERRUPTS''
OUT. EXIT 'EGP20''

```

CLASP KERNEL 3 NON-INTERRUPT SEQUENCER

```

PRGC .NINTSEQ1      ''NON-INTERRUPT SEQUENCER FOR PHASES 1 AND 3''
NIS1.  IF ARSTAT    THEN .MAR00 $ .PERPROC      END
          ''ACCELEROMETER READ''
      IF SASSTAT    THEN .MSA00 $ .PERPROC      END
          ''SIMULATED ACCEL      (NOT CODED)''
      IF APSTAT     THEN .MAP00 $ .PERPROC      END
          ''ACCELEROMETER PROCESSING''
      IF DVSTAT     THEN .MDV00 $ .PERPROC      END
          ''F/M CALCULATIONS      (NOT CODED)''
      IF DPSTAT     THEN .MDP00 $ .PERPROC      END
          ''DISCRETE PROCESSOR    (NOT CODED)''
      IF NESTAT     THEN .MNE00 $ .PERPROC      END
          ''BOOST NAVIGATION      (NOT CODED)''
      IF TCSTAT     THEN .MTC00 $ .PERPROC      END
          ''RESTART CALCULATIONS  (NOT CODED)''
      IF PASTAT     THEN .MPA00 $ .PERPROC      END
          ''PHASE ACTIVATOR''      (NOT CODED)''
      IF TTSTAT     THEN .MTT00 $ .PERPROC      END
          ''TIME TILT GUIDANCE    (NOT CODED)''
      IF CC1STAT    THEN .MCC10 $ .PERPROC      END
          ''CHI COMPUTATIONS      (NOT CODED)''
      IF IGSTAT     THEN .MIG00 $ .PERPROC      END
          ''ITERATIVE GUIDANCE MODE''
      IF HSSTAT     THEN .MHS00 $ .PERPROC      END
          ''S4B CUTOFF PREDICTION (NOT CODED)''
      IF OGSTAT     THEN .MOG00 $ .PERPROC      END
          ''ORBITAL GUIDANCE      (NOT CODED)''
      IF TGSTAT     THEN .MTG00 $ .PERPROC      END
          ''TARGET UPDATE          (NOT CODED)''
      IF RSSTAT     THEN .MRS00 $ .PERPROC      END
          ''TIME-TO-GO TO RESTART (NOT CODED)''
      IF CSSTAT     THEN .MCS00 $ .PERPROC      END
          ''TIME BASE 6-CHECK     (NOT CODED)''
      IF TB1STAT    THEN .MTB10 $ .PERPROC      END
          ''TIME BASE 1           (NOT CODED)''
      IF TB57STAT   THEN .MTB57 $ .PERPROC      END
          ''TIME BASE 5/7        (NOT CODED)''
      IFMSSTAT     THEN .MMS00 $ .PERPROC      END
          ''MINOR LOOP SUPPORT    (NOT CODED)''
      IF PGSTAT     THEN .MPG00 $ .PERPROC      END
          ''SIM PLATFORM GIM ANGLE (NOT CODED)''
      IF EBSTAT     THEN .MEB00 $ .PERPROC      END
          ''ETC/BTC              (NOT CODED)''
      GOTO NIS1
EXIT     ''NINTSEQ1''
PROC .NINTSEQ2      ''NON-INTERRUPT SEQUENCER FOR PHASES 2 AND 4''
NIS2.  IF CTSTAT    THEN .MCT00 $ .PERPROC      END
          ''DATA COMPRESSION TELEM(NOT CODED)''
      IF DTSTAT     THEN .MDT00
          ''SECTOR DUMP TELEMETRY (NOT CODED)''
      .PERPROC ''INSURE PERIODIC PROCESSOR GETS EXECUTED''
      GOTO NIS2
EXIT     ''NINTSEQ2''

```

CLASP KERNEL 4 PERIODIC PROCESSOR

```

PROC .PERPROC      ''PERIODIC PROCESSOR''
  DECLARE FIXED, VPPOT 0
  DIRECT
    ''READ REAL TIME CLOCK INTO TEMP''
  END
  DVPTG = (TEMP - VPPOT LAND MSKRTC) RSH 2
  VPPOT = TEMP
  FOR I = 0 TO 2
    IF NOT PPSTAT(I) THEN GOTO PP20          END
    DLPTL(I) = DLPTL(I) + DVPTG
    IF DLPTL(I) LS DLPRL(I) THEN GOTO PP20  END
    GOTO ( , PP1, PP2, *) I
    .MPC50      ''50 SEC DATA COMP (NOT CODED)'' $ GOTO PP10
  PP1.         .MPC60      ''60 SEC DATA COMP (NOT CODED)'' $ GOTO PP10
  PP2.         .MPC99      ''100 SEC DATA COMP (NOT CODED)''
  PP10.        DLPTL(I) = 0
  PP20.        END
EXIT          ''PERPROC''

```

CLASP KERNEL 5 EVENTS PROCESSOR

```

PROC .MEP00      'EVENTS PROCESSOR (TIMER 2 ENTRY)''
''
'' EVENTS PROCESSOR TABLE ''
''
'' ONLY A PORTION OF THE TABLE (THROUGH TIME BASE 3) ''
'' HAS BEEN CODED. ''
''
'' EACH ENTRY CONSISTS OF TWO WORDS: ''
'' 1. AN INDEX IDENTIFYING THE APPLICATION MODULE TO ''
'' PERFORM PROCESSING FOR THE EVENT. ''
'' 2. EVENT EXECUTION TIME. ''
''
'' NOTE: AN ENTRY INDEX WITH A VALUE OF ZERO IS EITHER SET ''
'' DYNAMICALLY IN REAL TIME OR IS USED TO DISABLE ''
'' THE EVENTS PROCESSOR FOR THE REMAINDER OF A TIME ''
'' BASE. ''
''
DECLARE INTEGER, EPTABLE (2, 131)
= ( 1 0 'START OF TIME BASE 0 TABLE' X
0 16.0*DKRTCSEC X
0 17.0*DKRTCSEC X
0 17.5*DKRTCSEC X
0 0 X
2 0 'START OF TIME BASE 1 TABLE' X
3 1.0*DKRTCSEC X
0 6.0*DKRTCSEC X
4 9.0*DKRTCSEC X
5 10.0*DKRTCSEC X
0 14.0*DKRTCSEC X
6 134.7*DKRTCSEC X
0 0 X
7 0 'START OF TIME BASE 2 TABLE' X
0 0 X
0 18.4*DKRTCSEC X
0 27.5*DKRTCSEC X
0 0 X
8 0 'START OF TIME BASE 3 TABLE' X
9 0 X
10 0 X
0 0 X
11 0 X
12 0 X
13 1.4*DKRTCSEC X
14 4.4*DKRTCSEC X
15 4.4*DKRTCSEC X
16 6.7*DKRTCSEC X
0 6.7*DKRTCSEC X
17 6.7*DKRTCSEC X
18 40.6*DKRTCSEC X
19 40.6*DKRTCSEC X
20 58.6*DKRTCSEC X
21 60.6*DKRTCSEC X
0 299.0*DKRTCSEC X
0 355.0*DKRTCSEC X
0 388.5*DKRTCSEC X

```


CLASP KERNEL 5 EVENTS PROCESSOR

```

                                0      0)
EP00. IF DFTBCEP
EP04A.     THEN DFTBCEP = FALSE
                                GOTO EP02
END
GOTO (EP100,EP101,EP102,EP103,EP104,EP105,EP106,EP107,EP108, X
      EP109,EP110,EP111,EP112,EP113,EP114,EP115,EP116,EP117, X
      EP118,EP119) EPTABLE(0,EPTINDX) - 1
EP100. .LE285 ''SCHEDULE WATER METHANOL      (NOT CODED)'' $ GOTO EP01
EP101. .LE25  ''TIME BASE 1 SETUP          (NOT CODED)'' $ GOTO EP01
EP102. .LE30  ''COMMAND INIT OF YAW MANEUVER(NOT CODED)'' $ GOTO EP01
EP103. .LE35  ''COMMAND TERM OF YAW MANEUVER(NOT CODED)'' $ GOTO EP01
EP104. .LE40  ''SET ACCEL REASON. TEST CONST(NOT CODED)'' $ GOTO EP01
EP105. .LE50  ''START TIME BASE 2          (NOT CODED)'' $ GOTO EP01
EP106. .LE55  ''DISABLE THRUST CONSTRAINT   (NOT CODED)'' $ GOTO EP01
EP107. .LE75  ''TIME BASE 3 SETUP          (NOT CODED)'' $ GOTO EP01
EP108. .LE70  ''SET ACCEL REASON. TEST CONST(NOT CODED)'' $ GOTO EP01
EP109. .LE250 ''CHNG GIMB REASON. TEST CONST(NOT CODED)'' $ GOTO EP01
EP110. .LE355 ''DEQUEUE TIME TILT          (NOT CODED)'' $ GOTO EP01
EP111. .LE365 ''F/M UNCERT FOR THRUST MISAL (NOT CODED)'' $ GOTO EP01
EP112. .LE82  ''ENABLE DIN 22 AND INT 2     (NOT CODED)'' $ GOTO EP01
EP113. .LE100 ''SET ACCEL BACKUP PROFILE    (NOT CODED)'' $ TOTO EP01
EP114. .LE95  ''SET ACCEL REASON. TEST CONST(NOT CODED)'' $ GOTO EP01
EP115. .LE90  ''ENABLE DIN 19              (NOT CODED)'' $ GOTO EP01
EP116. .LE96  ''ENQUEUE F/M CALC, SMOOTHING (NOT CODED)'' $ GOTO EP01
EP117. .LE105 ''ENQUEUE IGM                (NOT CODED)'' $ GOTO EP01
EP118. .LE115 ''SET MINOR LOOP PARAMETERS  (NOT CODED)'' $ GOTO EP01
EP119. .LE111 ''SET SMC FLAG                (NOT CODED)'' $ GOTO EP01
EP120. .LE110 ''ENQUEUE SMC                (NOT CODED)''
EP01. LOCK T1INT,T2INT,EX1INT,EX2INT,EX3INT,EX4INT,EX5INT,EX6INT, X
      EX7INT,EX8INT,EX9INT
      IF DFTBCEP THEN GOTO EP04A END
      EPTINDX = EPTINDX + 1
      DQST2 = 3 ''SET INDEX FOR EP ENTRY IN T2 SCHED CONT TABLE''
      IF EPTABLE(0,EPTINDX)
EP03.     THEN IF EPTABLE(1,EPTINDX) EQ VTOLD
              THEN UNLOCK ''RELEASE PREVIOUSLY ENABLED INT''
              GOTO EP00
              END
              VTOLD = EPTABLE(1,EPTINDX)
              DLTTL(DQST2) = DVTMR + VTOLD
              ELSE T2STAT(DQST2) = FALSE
              IF NOT DFIL1
                  THEN .EGP07 ''RESCHEDULE T2 (NOT CODED)''
              END
      END
EP02. UNLOCK ''RELEASE PREVIOUSLY ENABLED INTERRUPTS''
EXIT  ''MEP00''
PROC .MEP05 ''EVENTS PROCESSOR (TIME BASE CHANGE ENTRY)''
      DECLARE INTEGER, EPTBC(10)CONSTANT = 0 5 13 18 38 55 71 93 107 110
      EPTINDX = EPTBC(DTRID) - 1
      .MEP10
EXIT  ''MEP05''
PROC .MEP10 ''EVENTS PROCESSOR (RESCHEDULE ENTRY)''
      EPTINDX = EPTINDX + 1

```

CLASP KERNEL 5 EVENTS PROCESSOR

```

DQST2 = 3 ''SET INDEX FOR EP ENTRY IN T2 SCHED CONT TABLE''
IF EPTABLE(0,EPTINDX)
EPOR.      THEN VTOLD = EPTABLE(1,EPTINDX)
            DLTTL(DQST2) = DVTMR + VTOLD
            T2STAT(DQST2) = TRUE
            ELSE T2STAT(DQST2) = FALSE
END
IF NOT DFIL1
            THEN .EGP07      ''RESCHEDULE TIMER 2      (NOT CODED)''
END
EXIT      ''MEP10''
```

CLASP KERNEL 6 ITERATIVE GUIDANCE MODE

```

PROC .MIG00      ''ITERATIVE GUIDANCE MODE''
  DECLARE BOOLEAN,
    CHIBARST ,
    REITERAT ,
    PHASE ,
    SMCFLAG ,
    S4BURN
  DECLARE FIXED,
    COSTHETA 25, DELTAL3 12, DELTAVVP (3) 11,
    DELTA2 2, DPHII 33, DPHIT 33,
    EPSILON2 15, EPSILON3 15, GS (3) 21,
    GT 20, GV (3) 21, GVSTAR (3) 21,
    GVT (3) 20, J1 4, J12 2,
    J2 4, J3 2, J3P 2,
    KCCT4 23 CONSTANT = 1.53 ,
    KCCT8 23 CONSTANT = 1.55 ,
    KMU -24 CONSTANT = -.39860320E15 ,
    KT 48 CONSTANT = .48497964E-7 ,
    K1 2, K2 2, K3 -2,
    K4 -2, LYP 12, L1 12,
    L12 12, L2 12, L3 12,
    L3P 12, MS4(3,3) 25, M4V(3,3) 25,
    PHII 25, PHIIT 25, PHIT 25,
    P1 -5, P12 -6, P2 -5,
    Q1 -5, Q12 -5, Q2 -5,
    R 2, ROVEX3 36, RS (3) 2,
    RT 2, RV (3) 2, RVT (3) 2,
    R4 (3) 2, SINTHETA 25, S1 4,
    S12 2, S2 4, TAU1 15,
    TAU2 15, TAU3 15, TCI 15,
    THETAT 25, TSTAR 15, T1C 15,
    T1I 15, T2I 15, T3I 15,
    U1 -12, U12 -13, U2 -12,
    V 11, VEX1 12, VEX2 12,
    VEX3 12, VS (3) 11, VT 11,
    VV (3) 11, VVT (3) 11, V4 (3) 11
''DUE TO THE SIZE OF IGM, ONLY A SECTION OF IT HAS BEEN CODED. ''
''PART OF THE GUIDANCE COMPUTATIONS HAVE BEEN SELECTED TO DEMON-''
''STRATE MATHEMATICAL OPERATIONS. THE PHASING PORTION OF IGM ''
''HAS NOT BEEN CODED SINCE SIMILAR CAPABILITIES ARE ILLUSTRATED ''
''BY OTHER KERNELS. ''
''
'' IG251 - IGM GUIDANCE PARAMETERS COMPUTATIONS ''
''
'' ROTATE POSITION AND VELOCITY INTO TARGET PLANE ''
''
IG253. R4 = MS4/*RS
.UTR00 ''DELAY FOR TELEMETRY AS REQUIRED''
DIRECT ''TELEMETER X POSITION IN 4 SYSTEM, R4(0)''
END
.UTR00 ''DELAY FOR TELEMETRY AS REQUIRED''
DIRECT ''TELEMETER Y POSITION IN 4 SYSTEM, R4(1)''
END

```

CLASP KERNEL 6 ITERATIVE GUIDANCE MODE

```

UNLOCK ''RELEASE INTERRUPTS DISABLED BY TELEMETRY DELAY ROUT''
V4 = MS4/* /VS
.UTR00 ''DELAY FOR TELEMETRY AS REQUIRED''
DIRECT
''TELEMETER Z POSITION IN 4 SYSTEM, R4(2)''
END
.UTR02 ''DELAY FOR TELEMETRY AS REQUIRED''
DIRECT
''TELEMETER Y VELOCITY IN 4 SYSTEM, V4(1)''
END
UNLOCK ''RELEASE INTERRUPTS DISABLED BY TELEMETRY DELAY ROUT''
''
'' CALCULATE RANGE ANGLE MEASURED IN ORBIT PLANE
''
IG254. IF T2I EQ 0.
      THEN L12,J12,S12,Q12,P12,U12 = 0.
          GOTO IG259
END
IF T1I EQ 0.
      THEN L1,J1,S1,Q1,P1,U1 = 0.
          GOTO IG258.
END
L1 = VEX1*.LOG(TAU1/(TAU1 - T1I))
J1 = L1*TAU1 - VEX1*T1I
S1 = L1*T1I - J1
Q1 = S1*TAU1 - .5*VEX1*T1I**2
P1 = J1*TAU1 - .5*VEX1*T1I**2
U1 = Q1*TAU1 - VEX1*T1I**3/6.
IG258. L2 = VEX2*.LOG(TAU2/(TAU2 - T2I))
      J2 = L2*TAU2 - VEX2*T2I
      S2 = L2*T2I - J2
      Q2 = S2*TAU2 - .5*VEX2*T2I**2
      P2 = J2*TAU2 - .5*VEX2*T2I**2
      U2 = Q2*TAU2 - VEX2*T2I**3/6.
      L12 = L1 + L2
      J12 = J1 + J2 + L2*T1I
      S12 = S1 - J2 + L12*(T2I + TCI)
      Q12 = Q1 + Q2 + S2*T1I + J1*T2I
      P12 = P1 + P2 + T1I*(2.*J2 + L2*T1I)
      U12 = U1 + U2 + T1I*(2.*Q2 + S2*T1I) + T2I*P1
IG259. L3P = VEX3*.LOG(TAU3/(TAU3 - T3I))
      LYP = L12 + L3P
      J3P = L3P*TAU3 - VEX3*T3I
      T1C = T1I + T2I + TCI
      TSTAR = T1C + T3I
      PHII = .ATAN(R4(2),R4(0))
''
'' DETERMINE PHASE
''
IG260. IF PHASE ''IS FOR LEAVING ORBIT''
IG262. THEN ''CALCULATE TERMINAL CONDITIONS''
      SINTHETA = (RS/* /VS)/(R*V)
      COSTHETA = .SQRT(1. - SINTHETA**2)
      DPHII = V/R*COSTHETA
      DPHIT = VT/RT*.COS(THETAT)

```

CLASP KERNEL 6 ITERATIVE GUIDANCE MODE

```

PHIIT = .5*(DPHII + DP HIT)*TSTAR
PHIT = PHII + PHIIT
.UTR02 'DELAY FOR TELEMETRY AS REQUIRED''
DIRECT 'TELEMETER TERMINAL RANGE ANGLE, PHIT''
END
UNLOCK 'RELEASE INT LOCKED BY TELEM DELAY ROUTINE''
IF TSTAR LG EPSILON3 THEN GOTO IG269 END
.MIG30 'CALL TERM RAD, VEL, FLT ANGLE (NOT CODED)''
GT = - KMU/RT**2
.UTR00 'DELAY FOR TELEMETRY AS REQUIRED''
DIRECT 'TELEMETER TERMINAL GRAVITY VECTOR, GT''
END
UNLOCK 'RELEASE INT LOCKED BY TELEM DELAY ROUTINE''
GVT(0) = GT*.COS(THETAT)
GVT(1) = 0.
GVT(2) = GT*.SIN(THETAT)
RVT(0) = RT*.COS(THETAT)
RVT(1), RVT(2) = 0.
PHIT = PHIT - THETAT
IG269. ELSE 'CALCULATE INTERMEDIATE PARAMETERS'' X
DELTA2 = V*TSTAR - J3P + LYP*T3I - ROVEX3*((TAU1 - X
T1I)*L1 + (TAU2 - T2I)*L2 + (TAU3 - T3I) X
*L3P)*(LYP + V -VT)
PHIIT = KT*(S12 + DELTA2)
PHIT = PHII + PHIIT
.UTR02 'DELAY FOR TELEMETRY AS REQUIRED''
DIRECT 'TELEMETER TERMINAL RANGE ANGLE, PHIT''
END
UNLOCK 'RELEASE INT LOCKED BY TELEM DELAY ROUTINE''
END
''
'' ROTATE POSITION, VELOCITY, GRAVITY TO INJECTION SYSTEM ''
''
IG291. M4V(0,0), M4V(2,2) = .COS(PHIT)
M4V(0,2) = .SIN(PHIT)
M4V(2,0) = -.SIN(PHIT)
M4V(1,1) = 1.
M4V(1,0), M4V(0,1), M4V(2,1), M4V(1,2) = 0.
RV = M4V**/R4
VV = M4V**/V4
GV = M4V**/MS4**/GS
GVSTAR(*) = .5*(GVT(*) + GV(*))
DELTAVVP(*) = VVT(*) - VV(*) - TSTAR*GVSTAR(*)
''
'' IG314 - CALCULATE TIME-TO-GO (NOT CODED)''
''
IF REITERAT
THEN REITERAT = FALSE
L3P = L3
J3P = J3
LYP = LYP + DELTAL3
GOTO IG260

```

CLASP KERNEL 6 ITERATIVE GUIDANCE MODE

```

ELSE REITERAT = TRUE
END
''IG324 - COMPUTE CORRECTED VELOCITIES TO BE GAINED (NOT CODED)''
''IG326 - CALCULATE DESIRED PITCH AND YAW (NOT CODED)''
IF CHIBARST THEN GOTO IG350 END
IF TSTAR GW EPSILON2 THEN GOTO IG360 END
IF S4BURN
THEN DVMC5 = DVMC5 LXOR MSKM5CBS
DVMLR = 25.*KCCT4
DV1MR = .04/KCCT4
ELSE DVMC6 = DVMC6 LXOR MSKM6CBS
DVMLR = 25.*KCCT8
DV1MR = .04/KCCT8
END
IG340. CHIBARST = TRUE
IG350. K1, K2, K3, K4 = 0.
GOTO IG440
IG360.'''IG361 - COMPUTE INTERMEDIATE PARAMETERS (NOT CODED)''
IG440. .UTR00 ''DELAY FOR TELEMETRY AS REQUIRED''
DIRECT
''TELEMETER T3I''
END
UNLOCK ''RELEASE INTERRUPTS DISABLED BY TELEMETRY DELAY ROUT''
''IG446 - COMPUTE PITCH AND YAW IN 4 SYSTEM (NOT CODED)''
IF SMCFLAG THEN .MSM00 END ''COMP SMC TERMS (NOT CODED)''
.MCC00 ''PERFORM CHI COMPUTATIONS'' (NOT CODED)''
IF DFILE LAND MSKFPSI2
THEN .EGP32(MSKSCCO) ''ENABLE INTERRUPT 2 (NOT CODED)''
END
EXIT ''MIG00''

```

CLASP KERNEL 7 DIGITAL COMMAND SYSTEM

```

PROC .MDS00      ''DIGITAL COMMAND SYSTEM''
  DECLARE BOOLEAN,
    DCSMSTAT (20),
    FDSER      ,
    FDSPG      ,
    FDSRE
  DECLARE INTEGER,
    DCSDATCT (20) CONSTANT = (0 1 35 2 2 3 3(0) 35 8(0) 6 0),
    DCSDTCNT ,
    DCSERLIM  CONSANT = 7,
    DCSER04   CONSANT = 0'040000000',
    DCSER10   CONSANT = 0'100000000',
    DCSER14   CONSANT = 0'140000000',
    DCSER20   CONSANT = 0'200000000',
    DCSER24   CONSANT = 0'240000000',
    DCSER44   CONSANT = 0'440000000',
    DCSER60   CONSANT = 0'600000000',
    DCSER64   CONSANT = 0'640000000',
    DCSER74   CONSANT = 0'740000000',
    DCSINDX   ,
    DCSMODE (64) CONSTANT = (5(0) 8 2(0) 1 2 3 4 5 2(0) 14 6 0 7
                             2(0) 19 3(0) 9 0 15 17 8(0) 13 4(0)
                             18 10 11 12 2(0) 16 15(0)),
    DCSSTCOD(20) CONSTANT = (0'000000000' 0'100000000' 0'110000000'X
                             0'120000000' 0'130000000' 0'140000000'X
                             0'200000000' 0'220000000' 0'050000000'X
                             0'310000000' 0'770000000' 0'770000000'X
                             0'770000000' 0'450000000' 0'170000000'X
                             0'330000000' 0'600000000' 0'340000000'X
                             0'520000000' 0'250000000'),
    VDSBL (35),
    VDSER ,
    VDSRC ,
    VDSSB ,
    VDS01
  UNLOCK ''RELEASE PREVIOUSLY ENABLED INTERRUPTS''
  DIRECT
    ''READ DISCRETE INPUT REGISTER INTO TEMP''
    ''READ DIGITAL COMMAND SYSTEM INPUT INTO VDS01''
  END
  IF NOT TEMP LAND MSKDCSMD      THEN GOTO DS60      END
  ''
  ''PROCESS DCS MODE COMMAND      ''
  ''
DS09.  IF (VDS01 LSH 7 LXOR VDS01) LAND MSKDCSCM NQ MSKDCSCM
        THEN VDSER = DCSER10
        GOTO DS220
  END
  IF VDS01 LAND MSKDCSSB
        THEN VDSER = DCSER24
        GOTO DS220
  END
  IF VDS01 LAND MSKDCSMC EQ MSKDCSTR      THEN GOTO DS25      END
  IF NOT FDSER
        THEN VDSER = DCSER20

```

CLASP KERNEL 7 DIGITAL COMMAND SYSTEM

```

                GOTO DS220
            END
            IF DFDL OR FDSPG
                THEN VDSER = DCSER64
                GOTO DS220
            END
DS20.          FDSPG = TRUE
DS25.          DCSINDX = DCSMODE(VDS01 RSH 20)
                IF NOT DCSMSTAT(DCSINDX)
                    THEN FDSPG = FALSE
                    VDSER = DCSER74
                    GOTO DS220
            END
            ''TELEMETER STATUS CODE TWICE''
            .UTR24  ''DELAY FOR TELEMETRY AS REQUIRED''
            DIRECT
                ''TELEMETER DCS STATUS CODE, DCSSTCOD(DCSINDX)''
            END
            .UTR24  ''DELAY FOR TELEMETRY AS REQUIRED''
            DIRECT
                ''TELEMETER DCS STATUS CODE, DCSSTCOD(DCSINDX)''
            END
            UNLOCK ''RELEASE INTERRUPTS DISABLED BY TELEMETRY DELAY ROUT''
            .DS200  ''ISSUE CRP''
            DCSDTCNT, VDSSB = 0
            GOTO DS100
        ''
        ''
        ''PROCESS DCS DATA WORD
        ''
DS60.          IF FDSEN
                THEN VDSER = DCSER04
                GOTO DS220
            END
            IF (VDS01 LSH 7 LXOR VDS01) LAND MSKDCSCM NQ MSKDCSCM
                THEN VDSER = DCSER44
                GOTO DS220
            END
            IF VDS01 LAND MSKDCSSB NQ VDSSB
                THEN VDSER = DCSER60
                GOTO DS220
            END
DS110. ''TELEMETER DATA WORD TWICE''
        .UTR24  ''DELAY FOR TELEMETRY AS REQUIRED''
        DIRECT
            ''TELEMETER DCS DATA WORD, VDS01''
        END
        .UTR24  ''DELAY FOR TELEMETRY AS REQUIRED''
        DIRECT
            ''TELEMETER DCS DATA WORD, VDS01''
        END
        UNLOCK ''RELEASE INTERRUPTS DISABLED BY TELEMETRY DELAY ROUT''
        .DS200  ''ISSUE CRP''
        VDSBL(DCSDTCNT) = VDS01 LAND MSKDCSCM
        VDSSB = VDSSB LXOR MSKDCSSB
        DCSDTCNT = DCSDTCNT + 1
    
```


CLASP KERNEL 7 DIGITAL COMMAND SYSTEM

```

DS100.  IF DCSDTCNT LS DCSDATCT(DCSINDX) THEN GOTO DCSEXIT      END
        GOTO (,DS01,DS02,DS03,DS04,DS05,DS06,DS07,DS08,DS09A,DS10, X
            DS11,DS12,DS13,DS14,DS15,DS16,DS17,DS18,DS19) DCSINDX
        FDSPG = FALSE
        VDSER = DCSE14
        GOTO DS220
DS01.   .DS260          ''TIME BASE UPDATE (NOT CODED)'' $ GOTO DS530
DS02.   .DS330(=DS235.) ''NAVIGATION UPDATE (NOT CODED)'' $ GOTO DS530
DS03.   .DS380(=DS220.) ''GENERALIZED SS (NOT CODED)'' $ GOTO DS530
DS04.   .DS430          ''SECTOR DUMP (NOT CODED)'' $ GOTO DS530
DS05.   .DS470          ''SINGLE MEM LOC TEL(NOT CODED)'' $ GOTO DS530
DS06.   .DS510          ''TERMINATE (NOT CODED)'' $ GOTO DS530
DS07.   .DS540          ''MANEUVER UPDATE (NOT CODED)'' $ GOTO DS530
DS08.   .DS550          ''MANEUVER INHIBIT (NOT CODED)'' $ GOTO DS530
DS09A.  .DS670(=DS235.) ''TARGET UPDATE (NOT CODED)'' $ GOTO DS530
DS10.   .DS700          ''ANTENNA TO OMNI (NOT CODED)'' $ GOTO DS530
DS11.   .DS720          ''ANTENNA TO LOW (NOT CODED)'' $ GOTO DS530
DS12.   .DS740          ''ANTENNA TO HIGH (NOT CODED)'' $ GOTO DS530
DS13.   .DS770          ''INHIBIT WATER CONT(NOT CODED)'' $ GOTO DS530
DS14.   .DS790          ''TIME BASE B ENABLE(NOT CODED)'' $ GOTO DS530
DS15.   .DS810          ''EXECUTE MANEUVER A(NOT CODED)'' $ GOTO DS530
DS16.   .DS840          ''TD AND E ENABLE (NOT CODED)'' $ GOTO DS530
DS17.   .DS860          ''EXECUTE MANEUVER B(NOT CODED)'' $ GOTO DS530
DS18.   .DS900          ''S4B/IU LUNAR IMPCT(NOT CODED)'' $ GOTO DS530
DS19.   .DS960          ''ENABLE TB6D ALT SQ(NOT CODED)'' $ GOTO DS530
        ''
        ''PROCESS DCS ERROR CONDITION
        ''
DS220.  VDSRC = VDSRC + 1
        IF VDSRC LS DCSE14M
            THEN FDSRE = FALSE
            ELSE FDSRE = TRUE
        END
        VDSER = VDSER + VDSRC + (VDS01 RSH 12 LAND MSKDCSER)
DS235.  ''TELEMETER ERROR CODE TWICE''
        .UTR24 ''DELAY FOR TELEMETRY AS REQUIRED''
        DIRECT
            ''TELEMETER DCS ERROR CODE, VDSER''
        END
        .UTR24 ''DELAY FOR TELEMETRY AS REQUIRED''
        DIRECT
            ''TELEMETER DCS ERROR CODE, VDSER''
        END
        UNLOCK ''RELEASE INTERRUPTS DISABLED BY TELEMETRY DELAY ROUT''
        IF NOT FDSRE THEN GOTO DCSEXIT      END
DS530.  VDSRC = 0
        FDSER = TRUE
        FDSPG = FALSE
CLOSE   .DS200 ''ISSUE DCS COMMAND RESET PULSE''
        LOCK T1INT,T2INT,EX1INT,EX2INT,EX3INT,EX4INT,EX5INT,EX6INT, X
            EX7INT,EX8INT,EX9INT
        DIRECT
            ''SET COMMAND RESET BIT IN DISCRETE OUTPUT REGISTER''
            ''DELAY 4.13 MS''
            ''RESET COMMAND RESET BIT IN DISCRETE OUTPUT REGISTER''

```

CLASP KERNEL 7 DIGITAL COMMAND SYSTEM

```
END
UNLOCK ''RELEASE PREVIOUSLY ENABLED INTERRUPTS''
EXIT ''DS200''
DCSEXIT EXIT ''MDS00''
```

CLASP KERNEL 8 ACCELEROMETER PROCESSING

```

PROC .MAR00      ''ACCELEROMETER READ ROUTINE''
  DECLARE FIXED,
    COSTHY 25,
    COSTHZ 25,
    SINTHY 25,
    SINTHZ 25,
    VCCYA 25,
    VCCZA 25,
    VOACT 25 -2
  DECLARE INTEGER,
    VOAC (3)
  LOCK T1INT,T2INT,EX1INT,EX2INT,EX3INT,EX4INT,EX5INT,EX6INT,EX7INT, X
    EX8INT,EX9INT
  DIRECT          ''ENTER DIRECT MODE TO PERFORM I/O''
                  ''READ X ACCELEROMETER INTO DVAC(0)''
                  ''READ Y ACCELEROMETER INTO DVAC(1)''
                  ''READ Z ACCELEROMETER INTO DVAC(2)''
                  ''READ REAL TIME CLOCK INTO DVACT''
  END
  .UTR00      ''DELAY FOR TELEMETRY AS REQUIRED''
  DIRECT
    ''TELEMETER START TIME OF CURRENT TIME BASE, DVTI''
  END
  TEMP = DVTAS
  VOAXT = DVTMM + (DVACT - DVRTC - DVERT LAND MSKRTC)
  DVTAS = .24609375E-3 * VOACT
  DVTB = DVTAS - DVTI
  DVDT = DVTAS - TEMP
  .UTR00      ''DELAY FOR TELEMETRY AS REQUIRED''
  DIRECT
    ''TELEMETER TIME IN CURRENT TIME BASE, DVTB''
  END
  DVMC4 = DVMC4 LAND MSKRTCRS
  UNLOCK      ''RELEASE INTERRUPT INHIBITS''
  .UTR00      ''DELAY FOR TELEMETRY AS REQUIRED''
  DIRECT
    ''TELEMETER X ACCELEROMETER READING, DVAC(0)''
  END
  .UTR00      ''DELAY FOR TELEMETRY AS REQUIRED''
  DIRECT
    ''TELEMETER Y ACCELEROMETER READING, DVAC(1)''
  END
  UNLOCK ''RELEASE INTERRUPTS DISABLED BY TELEMETRY DELAY ROUTINE''
  IF ''TIME BASE 1'' DKT1 EQ 0.    ''NOT SET''
    THEN DVFMC = - DVG(0)
    ELSE DVMAS = DVMAS - DVEOF*DVFMR*DVDT
         DVFMC = DVEOF*DVFOR/DVMAS
  END
AR41.DVCA(2) = (DVCC(2) RSH 1) + (VCCZA RSH 1)
VCCZA = DVCC(2)
DVCA(1) = (DVCC(1) RSH 1) + (VCCYA RSH 1)
IF ABS(DVCC(1) - VCCYA) GG .5    ''COMPARE TO 90 DEG IN PIRADS''
  THEN DVCA(1) = DVCA(1) - 1. ''ADJUST BY 180 DEG IN PIRADS''END
VCCYA = DVCC(1)
.UTR00      ''DELAY FOR TELEMETRY AS REQUIRED''

```

CLASP KERNEL 8 ACCELEROMETER PROCESSING

```

DIRECT
    'TELEMETER Z ACCELEROMETER READING, DVAC(2)''
END
UNLOCK 'RELEASE INTERRUPTS DISABLED BY TELEMETRY DELAY ROUTINE''
AR100. DVDA(*) = (DVAC(*) LAND MSKACCA) - (VOAC(*) LAND MSKACCA)RSH 7
      DVDB(*) = ((DVAC(*) LAND MSKACCB) - (VOAC(*) LAND MSKACCB)
      LSH 14) RSH 7
      VOAC(*) = DVAC(*)
      .UTR00 'DELAY FOR TELEMETRY AS REQUIRED''
DIRECT
    'TELEMETER REAL TIME CLOCK AT ACCEL READ, DVACT''
END
UNLOCK 'RELEASE INTERRUPTS DISABLED BY TELEMETRY DELAY ROUTINE''
AR71. .USC00 (DVTH(2) = SINTHZ,COSTHZ) 'OBTAIN SIN/COS (NOT CODED)''
      .UTR00 'DELAY FOR TELEMETRY AS REQUIRED''
DIRECT
    'TELEMETER MISSION ELAPSED TIME, DVTAS''
END
UNLOCK 'RELEASE INTERRUPTS DISABLED BY TELEMETRY DELAY ROUTINE''
      .USC00 (DVTH(1) = SINTHY,COSTHY) 'OBTAIN SIN/COS (NOT CODED)''
      DVD(0) = 20.*DVRT*COSTHY*COSTHZ
      DVD(1) = 20.*DVRT*SINTHZ
      DVD(2) = -20.*DVRT*SINTHY*COSTHZ
      DVF(*) = DVFOM*DVD(*)
EXIT 'MAR00''
PROC .MAP00 'ACCELEROMETER PROCESSING ROUTINE''
      DECLARE FIXED,
      DELTA 7,
      KSN2D 25 CONSTANT = .0348994967, 'SINE 2 DEGREES''
      VACZR 7,
      VPOV (3) 7
      DECLARE INTEGER (3) CONSTANT,
      MSKAPDG = (0'0400000000',
      0'0100000000',
      0'2000000000'),
      MSKAPOF = (0'000000010',
      0'0000000200',
      0'000000020')
      DVVSQ = 0.
      VACZR = 20.*DVFOM*DVRT*KSN2D
      FOR I = 0 TO 2
AP400. IF ABS(DVDA(I) - DVDB(I)) LG 2. THEN GOTO AP450
      IF ABS(DVDA(I) - DVF(I)) LS ABS(DVDH(I) - DVF(I))
      THEN GOTO AP440
      DVMC4 = DVMC4 LOR MSKAPDG(I) LSH 1
      DELTA = DVDB(I)
      GOTO AP460
AP440. DVMC4 = DVMC4 LOR MSKAPDG(I)
AP450. DELTA = DVDA(I)
AP460. IF ABS(DELTA) GR 1. THEN GOTO AP500
      IF DFZER EQ FALSE THEN GOTO AP500
      IF ABS(DVF(I)) LS VACZR THEN GOTO AP500
      DVMC4 = DVMC4 LOR MSKAPOF(I)
AP530. DVMC4 = DVMC4 LOR MSKAPDG(I) LOR MSKAPDG(I) LSH 1
      DFSCM = FALSE

```

CLASP KERNEL 8 ACCELEROMETER PROCESSING

```

DELTA = DVFMC*DVD(I)
GOTO AP520
AP500.  IF DVF(I) LS 0.
        THEN IF DELTA LS 1.5*DVF(I) - DVRC(I)*DVDT           X
            OR DELTA GR .5*DVF(I) + DVRC(I)*DVDT
            THEN GOTO AP530                                     END
        ELSE IF DELTA GR 1.5*DVF(I) + DVRC(I)*DVDT           X
            OR DELTA LS .5*DVF(I) - DVRC(I)*DVDT
            THEN GOTO AP530                                     END
END
AP510.  DVVSQ = DVVSQ + DELTA**2
AP520.  VPOV(I) = VPOV(I) + DELTA
        DVDM(I) = .05*VPOV(I)
        .UTR01 ''DELAY FOR TELEMETRY AS REQUIRED''
        GOTO (AP521, AP522, AP523, *) I
AP521.  DIRECT
        ''TELEMETER X MEASURED VELOCITY, DVDM(0)''
        END
        GOTO AP524
AP522.  DIRECT
        ''TELEMETER Y MEASURED VELOCITY, DVDM(1)''
        END
        GOTO AP524
AP523.  DIRECT
        ''TELEMETER Z MEASURED VELOCITY, DVDM(2)''
        END
AP524.  UNLOCK ''RELEASE INTERRUPTS LOCKED BY TELEMETRY DELAY ROUT''
        END
EXIT ''MAP00''

```

CLASP KERNEL 9 MINOR LOOP

```

PROC .MML00      ''FLIGHT SIM ENTRY TO MINOR LOOP''
  IF DVLRC EQ 0
    THEN DVCC(*) = DVCC(*) - DVDC(*)
    ELSE DVLRC = DVLRC - 1
  END
  .MML20      ''EXECUTE MINOR LOOP''
EXIT ''MML00''
PROC .MML20      ''NORMAL MINOR LOOP ENTRY''
  OPTIMIZE TIME(20)
  DECLARE BOOLEAN,
    FBUGS
  DECLARE FIXED,
    KCPRG      14  CONSTANT = 2016.,
    VBUR (3)   25 ,
    VCG (3)    25 ,
    VCG0 (3)   14 ,
    VCG1 (3)   14 ,
    VCG10      14 ,
    VCG11      14 ,
    VCMND (3,3) 0 ,
    VCOD (3)   14 ,
    VDEL (3)   25 ,
    VGR (3)    0 ,
    VML0 (3)   14 ,
    VML1 (3)   14 ,
    VML2 (3)   14 ,
    VOLD (3)   14 ,
    VPGR (3)   0 ,
    VSF (3)    35 ,
    VOCK      25
  DECLARE INTEGER,
    FBUG (3),
    VFIO (3),
    VIRE ,
    VMEMR ,
    VMLET
    DVCC(*) = DVCC(*) + DVDC(*)
    IF FBUGS THEN GOTO ML500
ML001.FOR I = 2 BY -1 TO 0
  GOTO (ML201, ML101, , *) I
  IF VFIO(2) EQ 0 ''NORMAL''
    THEN DIRECT ''READ Z GIMBAL INTO VGR(2)''
    ELSE IF VFIO(2) EQ 1 ''BACKUP''
      THEN DIRECT''READ Z BACKUP INTO VGR(2)''
      ELSE VGR(2) = VPGR(2)
  END
  END
  GOTO ML004
ML101. DIRECT
  ''READ ERROR MONITOR REGISTER INTO VMEMR''
  END
  DVLDB = DVLDB - (VMEMR LAND MSKEMRLB)
  IF VFIO(1) EQ 0 ''NORMAL''
    THEN DIRECT ''READ Y GIMBAL INTO VGR(1)''
    ELSE IF VFIO(1) EQ 1 ''BACKUP''

```

CLASP KERNEL 9 MINOR LOOP

```

                THEN DIRECT 'READ Y BACKUP INTO VGR(1)'' END
                ELSE VGR(1) = VPGR(1)
            END
        END
    END
    GOTO ML004
ML201.  DVEMR = DVEMR LOR VMEMR
        IF VFIO(0) EQ 0 'NORMAL''
            THEN DIRECT 'READ X GIMBAL INTO VGR (0)'' END
            ELSE IF VFIO(0) EQ 1 'BACKUP''
                THEN DIRECT 'READ X BACKUP INTO VGR(0)'' END
                ELSE VGR(0) = VPGR(0)
            END
        END
    END
    IF VGR(1) GE 0. THEN GOTO ML020 END
    IF DVDGS LS 0 THEN GOTO ML432 END
    IF DVDGS EQ 0 THEN GOTO ML020
    ELSE GOTO ML637 END
ML432. .MDG00 (= J, ML434.) 'PROCESS DISAGREEMENT BIT (NOT CODED)''
        'DISAGREEMENT BIT PROCESSING WILL TAKE A NORMAL RETURN IF THE ''
        'DISAGREEMENT BIT IS FOUND TO BE INVALID. OTHERWISE IT WILL ''
        'TAKE THE ERROR EXIT TO ML434 AND SET J = 0 IF THE GIMBAL IS ''
        'INVALID OR J = 1 IF THE GIMBAL IS NOT VALID. ''
        ''
        GOTO ML020
ML434.  GOTO (ML4352, ML4351, ML4350, *) I
ML4350. IF VFIO(2) EQ 0 'NORMAL''
        THEN DIRECT
            'RESTART Y COD COUNTER''
            END
        ELSE DIRECT
            'RESTART Y COD COUNTER (BACKUP)''
            END
        END
    END
    GOTO (ML020, ML637, *) J
ML4351. IF VFIO(1) EQ 0 'NORMAL''
        THEN DIRECT
            'RESTART X COD COUNTER''
            END
        ELSE DIRECT
            'RESTART X COD COUNTER (BACKUP)''
            END
        END
    END
    GOTO (ML020, ML637, *) J
ML4352. IF VFIO(0) EQ 0 'NORMAL''
        THEN DIRECT
            'RESTART Z COD COUNTER''
            END
        ELSE DIRECT
            'RESTART Z COD COUNTER (BACKUP)''
            END
        END
    END
    GOTO (ML020, ML637, *) J
ML020.  VCOD(I) = VGR(I) LAND MSKGIMA
        IF VCOD(I) EQ 0. AND VOLD(I) EQ 0. AND ABS(VDEL(I)) GE VOCK
            THEN GOTO ML631

```

CLASP KERNEL 9 MINOR LOOP

```

END
IF ABS(VCOD(I) - VOLD(I)) LS VML0(I)
  THEN GOTO ML040
END
IF ABS(VCOD(I) - VOLD(I)) + VML0(I) GE VML1(I)
  THEN IF VCOD(I) LS VOLD(I)
    THEN VCG(I) = VCG(I) + VML2(I)
    ELSE VCG(I) = VCG(I) - VML2(I)
  END
  GOTO ML040
  ELSE GOTO ML630
END
ML040. DVTH(I) = VSF(I)*VCOD(I) + VCG(I)
      VOLD(I) = VCOD(I)
      VDEL(I) = DVTH(I) - DVCC(I)
      DFDBF = TRUE
      GOTO (ML245, ML145, ML045, *) I
ML045. VCMND(2,0) = DVA5*VDEL(2) - DVA4*VDEL(1)
      GOTO ML730
ML145. VCMND(1,0) = DVA1*VDEL(1) + DVA2*VDEL(2)
      GOTO ML730
ML245. VCMND(0,0) = DVA6*(VDEL(0) + DVA3*VDEL(1))
      GOTO ML730
ML630. VMLET = I + 3
      GOTO ML632
ML631. VMLET = I
ML632. VMLET = VMLET LSH 11 LOR VCOD(I) RSH 14 LOR VOLD(I)
      IF DVMC6 LAND MSKM6D04 EQ 0 THEN
        .UTR30    ''DELAY FOR TELEMETRY AS REQUIRED''
        DIRECT
        ''TELEMETER MINOR LOOP ERROR MESSAGE, VMLET''
      END
      END
      IF NOT DFDBF          THEN GOTO ML635          END
      DVRE(I) = DVRE(I) + 1
      IF DVRE(I) LS 0      THEN GOTO ML637          END
      IF DVRE(I) GR 0      THEN GOTO ML636          END
      VMLET = VCOD(I) RSH 14 LOR VOLD(I) LOR MSKERTAG
      VFIO(I) = 1 ''BACKUP''
      VCG(I) = (VCG(I) LAND MSK180DG) - VBUR(I)
      VML2(I) = -1. ''180 DEGREES IN PIRADS''
      VOLD(I) = (DVTH(I) LAND MSKM180D)*KCPBG LAND MSKGIMA
      VSF(I) = 1./KCPBG
      IF I EQ 2
        THEN DIRECT
          ''SET INTERNAL CONTROL REGISTER TO SELECT BU GIM''
        END
        DVICR = DVICR LXOR MSKICRBG
      END
      FBUGS,FBUG(I) = 1 ''PASS1''
      VML0(I) = VCG10
      VML1(I) = VCG11
      IF DVMC6 LAND MSKM6D04 EQ 0 THEN
        .UTR30    ''DELAY FOR TELEMETRY AS REQUIRED''
        DIRECT

```


CLASP KERNEL 9 MINOR LOOP

```

                                ''TELEMETER MINOR LOOP ERROR MESSAGE, VMLET''
                                END
                                END
                                GOTO ML637
ML635.  DVHDB = DVHDB - 1
        DFDBF = TRUE
        DVHDA = DVHDA + 1
        IF DVHDA LS 0                THEN GOTO ML636                END
        DIRECT
        ''SET INTERNAL CONTROL REGISTER TO SWITCH GIMBAL ORDER''
        END
        DVICR = DVICR LXOR MSKICRS0
        DVMC4 = DVMC4 LOR MSKM4AMF
        DVDGS = 0
ML636.  IF DVRE(I) GR VIRE AND DVMC6 LAND MSKM6D04 EQ 0
        THEN .UD000(MSKGRF) ''SET GUID REF FAIL DISC(NOT CODED)''
        END
ML637.  DFSMC = FALSE
        GOTO ML760
ML730.  IF ABS(VCMND(I,0)) GR DVM06    THEN VCMND(I,0) = DVM06    END
        IF ABS(VCMND(I,0) - VCMND(I,1)) GR DVM05
        THEN VCMND(I,0) = VCMND(I,1) + DVM05
        END
        VCMND(I,1) = VCMND(I,0)
        IF VCMND(I,0) LS 0.
        THEN VCMND(I,2) = MSKABLAD - VCMND(I,0)
        ELSE VCMND(I,2) = VCMND(I,0)
        END
ML760.  GOTO (ML260, ML160, ML060, *) I
ML060.  DIRECT
        ''ISSUE YAW COMMAND''
        END
        IF DVLDB LS 0
        THEN DIRECT
        ''SET INTERNAL CONTROL REG TO SELECT CONVERTER A''
        END
        END
        DVMLT = DVTMM + DVMLD + (DVTT1 - DVRTC LAND MSKRTC) RSH 2
        GOTO ML900
ML160.  DIRECT
        ''ISSUE PITCH COMMAND''
        END
        GOTO ML900
ML260.  DIRECT
        ''START SPECIAL DOM BACKUP GIMBAL''
        ''ISSUE YAW COMMAND''
        ''ISSUE ROLL COMMAND''
        END
ML900.  END
        GOTO MLEXIT
ML500.  FOR I = 0 TO 2
        GOTO (ML530, ML520, ML540, *) FBUG(I)
ML520.  FBUG(I) = 2
        GOTO ML530
ML540.  FBUG(I) = 0

```

CLASP KERNEL 9 MINOR LOOP

```
VML0(I) = VCG0(I)
VML1(I) = VCG1(I)
ML530. END
FRUGS = FBUG(0) LOR FRUG(1) LOR FBUG(2)
GOTO ML001
UNTIME
MLEXIT. EXIT 'MML20'
```

CLASP KERNEL 10 SWITCH SELECTOR PROCESSING

```

PROC .MSS00      'SWITCH SELECTOR PROCESSING'
''
'' SWITCH SELECTOR TABLE
''
'' THE SWITCH SELECTOR TABLE IS MADE UP OF A NUMBER OF
'' SMALLER TABLES, ONE FOR EACH TIME BASE AND FOR EACH
'' OF THE ALTERNATE SS SEQUENCES. THE SMALLER TABLES ARE
'' ORGANIZED INTO ONE LARGE TABLE BY AN OVERLAY. ONLY
'' THE TIME BASE 1 SEQUENCE HAS BEEN CODED.
''
'' EACH TABLE ENTRY REPRESENTS A SINGLE SS COMMAND AND
'' CONSISTS OF TWO WORDS.
''     1. TIME OF SS ISSUANCE.
''     2. SS STAGE AND ADDRESS.
''
DECLARE INTEGER, SSTABLE (2,1000)
DECLARE INTEGER, SSTTB1 (2,28) CONSTANT
      =( 5.0*DKRTCSEC  0'000000000'  X
        6.0*DKRTCSEC  0'106500000'  X
        14.0*DKRTCSEC 0'026100000'  X
        16.8*DKRTCSEC 0'406440000'  X
        20.0*DKRTCSEC 0'405440000'  X
        20.2*DKRTCSEC 0'406340000'  X
        24.0*DKRTCSEC 0'025740000'  X
        27.0*DKRTCSEC 0'402100000'  X
        29.0*DKRTCSEC 0'027740000'  X
        30.0*DKRTCSEC 0'403040000'  X
        32.0*DKRTCSEC 0'401100000'  X
        49.5*DKRTCSEC 0'020100000'  X
        75.0*DKRTCSEC 0'000000000'  X
        90.0*DKRTCSEC 0'402100000'  X
        95.0*DKRTCSEC 0'401100000'  X
        95.3*DKRTCSEC 0'022640000'  X
        105.0*DKRTCSEC 0'407640000'  X
        115.1*DKRTCSEC 0'025740000'  X
        119.8*DKRTCSEC 0'406740000'  X
        120.0*DKRTCSEC 0'405740000'  X
        120.1*DKRTCSEC 0'027740000'  X
        130.0*DKRTCSEC 0'404040000'  X
        132.4*DKRTCSEC 0'021640000'  X
        133.6*DKRTCSEC 0'400700000'  X
        133.8*DKRTCSEC 0'401700000'  X
        134.4*DKRTCSEC 0'021740000'  X
        134.6*DKRTCSEC 0'023740000'  X
        0'377777776'  0'000000000' )
''
'' ALTHOUGH THE REMAINING GROUPS OF SWITCH SELECTORS HAVE
'' NOT BEEN CODED, THEY ARE REFERENCED IN THE FOLLOWING
'' OVERLAY STATEMENT TO INDICATE HOW THE OVERALL SS TABLE
'' WOULD BE ORGANIZED.
''
OVERLAY SSTABLE = SSTTB1,SSTTB2,SSTTB3,SSTTB4,SSTTB5,SSTTB6, X
              SSTTB7,SSTTB8,SSTSIVB,SSTSIVA,SSTTB6A,SSTTB6B,SSTTB6C, X
              SST34C1,SSTGAIN,SSTTB6D,SSTALU,SSTGSS,SSTSBLO,SSTS8HI, X
              SSTSBOM,SSTEC5V,SSTEC5I,SSTTB3A,SSTTB5A,SSTTB5B

```

```

DECLARE BOOLEAN,
    FASE ,
    FBRNI ,
    FCLS4 ,
    FFBCH ,
    FHST ,
    FSSAC ,
    FSSIO ,
    FTADV ,
    FT60P
DECLARE FIXED -2 CONSTANT,
    KCSSK = .200*DKRTCSEC ,
    KSSB1 = .018*DKRTCSEC - 3. ,
    KSSB2 = .025*DKRTCSEC - 2. ,
    KSSB3 = .019*DKRTCSEC - 11. ,
    KSSB4 = .013*DKRTCSEC - 17. ,
    KSSB5 = .026*DKRTCSEC - 3. ,
    KSSB6 = .013*DKRTCSEC - 2. ,
    KSSB7 = .023*DKRTCSEC - 6. ,
    KSSB8 = .013*DKRTCSEC - 9. ,
    KSSRB = .050*DKRTCSEC - 2. ,
    KSS500MS = .500*DKRTCSEC ,
    KSS500S = 500.*DKRTCSEC
DECLARE FIXED -2,
    VATRR ,
    VATR4 ,
    VGBIA ,
    VSSRT ,
    VSSTM ,
    VSSW ,
    VSTGO
DECLARE INTEGER,
    SSTBPTR (8) CONSTANT = ( L'SSTB1'-L'SSTABLE',
                             L'SSTB2'-L'SSTABLE',
                             L'SSTB3'-L'SSTABLE',
                             L'SSTB4'-L'SSTABLE',
                             L'SSTB5'-L'SSTABLE',
                             0,
                             L'SSTB7'-L'SSTABLE',
                             L'SSTB8'-L'SSTABLE'),
    SST1PTR ,
    SST2PTR ,
    VASPI ,
    VHSTW ,
    VPSTG ,
    VSCCA ,
    VSC1 (3) ,
    VSC3 (3) ,
    VSNA ,
    VSNA1 ,
    VSSCA ,
    VSSFR ,
    VSTG
GOTO ( ,MSS05,MSS10,MSS20,MSS30,MSS40,MSS50,MSS55,MSS60,MSS70,X
      MSS80) DGSSM

```

CLASP KERNEL 10 SWITCH SELECTOR PROCESSING

```

LOCK T1INT,T2INT,EX1INT,EX2INT,EX3INT,EX4INT,EX5INT,EX6INT, X
EX7INT,EX8INT,EX9INT
FASE = FALSE
IF DVASW LAND MSKSS94C
    THEN DVASW = DVASW LAND MSKSS94C
        IF VASPI LAND MSKSS94C THEN GOTO SS0060          END
        .EGP08          ''RESCHEDULE TIMER 1 (NOT CODED)''
        VSAPI = MSKSS94C
        SST1PTR = L'SSTSIVB'-L'SSTABLE'
        GOTO SS1050
END
IF DVASW LAND MSKSS9PC
    THEN DVASW = DVASW LAND MSKSS9PC
        .EGP08          ''RESCHEDULE TIMER 1 (NOT CODED)''
        VSAPI = MSKSS9PC
        SST1PTR = L'SSTSIVA'-L'SSTABLE'
        GOTO SS1050
END
IF DVASW LAND MSKSS6C
    THEN DVASW = DVASW LXOR MSKSS6C
        VASPI = VASPI LOR MSKSS6C
        DVMC6 = DVMC6 LOR MSKM6T6C
        SST1PTR = L'SSTTB6C'-L'SSTABLE'
        .SSTUPD(=VATRR) ''UPDATE SS TIME''
        GOTO SS1050
END
IF DVASW LAND MSKSS91
    THEN VSC1(0),VSC1(1),VSC1(2) = SST1PTR,VASPI,VATRR
        VASPI = MSKSS91
        .SSTUPD(=VATRR) ''UPDATE SSTRIME''
        FTADV = TRUE
        IF DVASW LAND MSKSS6A
            THEN DVASW = DVASW LXOR MSKSS6A
                DVMC6 = DVMC6 LOR MSKM6T6A
                SST1PTR = L'SSTTB6A'-L'SSTABLE'
            ELSE IF DVASW LAND MSKSS41
                THEN DVASW = DVASW LXOR MSKSS41
                    SST1PTR = L'SSTS4C1'-L'SSTABLE'
                ELSE DVASW = DVASW LXOR MSKSS6B
                    DVMC6 = DVMC6 LOR MSKM6T6B
                    SST1PTR = L'SSTTB6B'-L'SSTABLE'
            END
        END
        GOTO SS1050
END
IF FSSAC
    THEN GOTO SS0060
    ELSE GOTO SS0000
END
FSSAC = FALSE
MSS05.  IF SSTABLE(0,SST1PTR) NQ MSKSSEND
SS0000.  THEN .SSTUPD(=VSTGO) ''UPDATE SS TIME''
SS0010.  VSTGO = VSSRT - VSTGO
        IF VSTGO LS K99500MS THEN GOTO MSS30          END
        IF DFTUP

```

CLASP KERNEL 10 SWITCH SELECTOR PROCESSING

```

        THEN DVTGB = DVTGB + VGBIA
            VGBIA = 0.
            DFTUP = FALSE
            GOTO SS0010
        ELSE IF DVASW THEN GOTO SS0170 END
            VSSTM = VSTGO + DVTRB - DVTGB - KCSSK
            DVSST = VSSTM + DVTMR
            DGSSM = 4 ''SET SS ENTRY FOR MSS30''
            IF NOT DFIL3
                THEN .EGP08 ''RESCHED T1(NOT CODED)''
            END
        END
        GOTO SS0060
SS0015. END
        IF DVASW THEN GOTO SS0170 END
        DIRECT
            ''READ REAL TIME CLOCK INTO TEMP''
        END
        DVSST = DVTMM + KSS500S + ((TEMP - DVRTC LAND MSKRTC) RSH 2)
        DGSSM = 1 ''SET SS ENTRY INDEX FOR MSS05''
        IF NOT DFIL3 THEN .EGP08 ''RESCHEDULE T1(NOT CODED)''END
        GOTO SS0060
SS0170. IF NOT DVASW LAND MSKSSCS3
        THEN IF DVASW LAND MSKSSACQ
            THEN DVASW = DVASW LXOR MSKSSACQ
                SST2PTR = L'SSTGAIN'-L'SSTABLE'
                .SSTUPD(= VATR4) ''UPDATE SS TIME''
            ELSE IF DVASW LAND MSKSST6D
                THEN DVASW = DVASW LXOR MSKSST6D
                    SST2PTR = L'SSTTB6D'-L'SSTABLE'
                    .SSTUPD(= VATR4)''UPDATE SS TM''
                ELSE DVASW = DVASW LXOR MSKSSLI
                    SST2PTR = L'SSTALU'-L'SSTABLE'
                    VATR4 = 0.
            END
        END
        FCLS4 = TRUE
        FTADV = FALSE
        FHST = TRUE
        .SS210 ''SET UP CLASS 4 ALTERNATE SEQUENCE''
        GOTO SS0000
        END
        IF VASPI THEN GOTO SS0060 END
        VSC3(0), VSC3(1), VSC3(2) = SST1PTR, VASPI, VATRR
        VASPI = MSKSSCL3
        .SSTUPD(= VATRR) ''UPDATE SS TIME''
        FTADV = TRUE
        FHST = TRUE
        IF DVASW LAND MSKSSGNS
            THEN DVASW = DVASW LXOR MSKSSGNS
                SST1PTR = L'SSTGSS'-L'SSTABLE''
            GOTO SS0230
        END
        IF DVASW LAND MSKSSSBL
            THEN DVASW = DVASW LXOR MSKSSSBL

```

CLASP KERNEL 10 SWITCH SELECTOR PROCESSING

```

SST1PTR = L'SSTSBLO'-L'SSTABLE''
GOTO SS0230

END
IF DVASW LAND MSKSSSBH
    THEN DVASW = DVASW LXOR MSKSSSBH
    SST1PTR = L'SSTSBHI'-L'SSTABLE''
    GOTO SS0230

END
IF DVASW LAND MSKSSSBO
    THEN DVASW = DVASW LXOR MSKSSSBO
    SST1PTR = L'SSTSBOM'-L'SSTABLE''
    GOTO SS0230

END
IF DVASW LAND MSKSSECV
    THEN SST1PTR = L'SSTECSV'-L'SSTABLE''
    GOTO SS0230

END
IF DVASW LAND MSKSSEC1
    THEN SST1PTR = L'SSTECS1'-L'SSTABLE''
    ELSE SST1PTR = L'SSTB3A'-L'SSTABLE''
    DVASW = DVASW LXOR MSKSST3A

END
SS0230. .SS210 'SET UP SS TABLE''
GOTO SS0000
SS0060. IF NOT FASE
    THEN FASE = TRUE
    UNLOCK 'RELEASE PREVIOUSLY ENABLED INTERRUPTS''

END
GOTO SSEXIT
MSS10. VASPI, VATRR, FCLS4 = 0
DVASW = DVASW LAND MSKSSWV
.EGPO8 'RESCHEDULE TIMER 1 (NOT CODED)''
FTADV = TRUE
SST1PTR = SST1PTR(DTBID - 1)
SS1050. .SS210 'SET UP NEXT SS''
IF FSSAC THEN GOTO MSS20 END
VSSW = KSSB1
FHST = TRUE
GOTO SS0000
MSS20. IF FSSIO
    THEN DIRECT
    'ISSUE, FORCED SS RESET''
    END

END
FHST = FALSE
SSTUPQ (KSSB8,1) 'SCHEDULE SS CHECK, MSS05''
VSSW = KSSB5
GOTO SS0060
MSS30. FSSAC = TRUE
VSNA, VSNA1 = VSNA RSH 2 LAND MSKSSSNA
IF NOT VSNA
    THEN FSSAC = FALSE
    .SS201 'ADVANCE TO NEXT SS''
    GOTO SS0000

END

```

CLASP KERNEL 10 SWITCH SELECTOR PROCESSING

```

VSTG = VSNA LAND VPSTG
FSSIO = VSTG
IF NOT FHST          THEN GOTO SS4000          END
IF DFLT EQ 2        THEN GOTO SS4000          END
DIRECT
    'READ SS FEEDBACK REGISTER INTO TEMP'
END
IF NOT TEMP LAND MSKSSHS      THEN GOTO SS4000          END
IF FSSIO
    THEN DIRECT
        'ISSUE SS RESET'
    END
END
.SSTUPQ(KSSB4,5) 'SCHEDULE STAGE AND ADDRESS ISSUANCE MSS40'
VSSW = KSSB5
GOTO SS0060
MSS40. FOR I = 0 TO 22 'DELAY BEFORE ISSUING STAGE AND ADDRESS'
        A = I
END
SS4000. IF FSSIO
        THEN DIRECT
            'ISSUE STAGE AND ADDRESS FROM VSNA'
        END
END
.SSTUPQ(VSSW,6) 'SCHEDULE ADDRESS VERIFICATION, MSS50'
FOR I = 0 TO 17 'DELAY FOR DOM TELEMETRY'
    A = I
END
DIRECT
    'OUTPUT SS AND DO REGISTERS VIA DOM TELEMETRY'
END
GOTO SS0060
MSS50. VSCCA = VSNA LXOR MSKSSHS
        VSSCA = VSCCA LAND MSKSSHS
        IF VSTG
            THEN DIRECT
                'READ SS FEEDBACK REGISTER INTO TEMP'
            END
            VSSFBR = TEMP LAND MSKSSHS
            ELSE VSSFBR = VSSCA
        END
MSS55. IF VSSFBR NQ VSSCA      THEN GOTO SS5540          END
        IF VASPI LAND MSKSS84C
            THEN DFILE = DFILE LOR MSKFPSIS
                DVSST = 1.E10
                GOTO SSEXIT
        END
        IF NOT VSSRT          THEN GOTO MSS60          END
        .SSTUPD(= DVTRB) 'UPDATE SS TIME'
        IF VSSRT = DVTRB LQ KSSRB      THEN GOTO MSS60          END
        VSSTM = VSSRT - DVTGB - KSSRB
        DVSST = VSSTM + DVTMR
        DGSSM = 8 'SET SS ENTRY INDEX FOR MSS60'
        IF NOT DFIL3
            THEN .EGP08          'RESCHEDULE TIMER 1          (NOT CODED)'

```


CLASP KERNEL 10 SWITCH SELECTOR PROCESSING

```

END
SS5540. IF VSSF8 EQ 0 AND VSSCA NQ MSKSSZF8 THEN GOTO MSS55 END
        GOTO SSEXIT
        IF FSSIO
            THEN DIRECT
                ''ISSUE SS RESET''
            END
        END
        .SSTUPQ(KSSB6,10) ''SCHED COMPLEMENTED STAGE, ADDRESS, MSS80''
        TEMP = (VSSCA LXOR VSSF8) LSH 7
SS5570. IF TEMP LS 0 THEN GOTO SS5580 END
        TEMP = TEMP LSH 1
        GOTO SS5570
SS5580. TEMP = TEMP LSH 1
        IF NOT TEMP THEN GOTO SSEXIT END
        DVMC4 = DVMC4 LOR MSKM4SSB
        IF FFBCH
            THEN FFBCH = FALSE
            DIRECT
                ''SET SS CHANNEL B BIT IN INTERNAL CONT REG''
            END
            DVICR = DVICR LOR MSKICRSB
        END
        .UTR30 ''DELAY FOR TELEMETRY AS REQUIRED''
        DIRECT
            ''TELEMETER SS FEEDBACK, VSSF8''
        END
        GOTO SSEXIT
MSS60. TEMP = VSTG LXOR MSKSSRD
        IF FSSIO
            THEN DIRECT
                ''ISSUE READ COMMAND FROM TEMP''
            END
        END
        DIRECT
            ''READ REAL TIME CLOCK INTO TEMP''
        END
        .SSTUPQ(KSSB2,9) ''SCHEDULE READ RESET, MSS70''
        TEMP = VSNA LSH 2 LOR TEMP LAND MSKRTC
        .UTR30 ''DELAY FOR TELEMETRY AS REQUIRED''
        DIRECT
            ''TELEMETER STAGE/ADDRESS AND READ TIME, TEMP''
        END
        IF NOT DFAQQ ''COMPRESS DATA WHEN NOT OVER STAT. (NOT CODED)''
            THEN TEMP = DVDCT + MSKSSDCT
                .MPC80(TEMP) ''COMPRESS TIME AND TAG''
                TEMP = (VSNA RSH 3) + MSKSSDCS
                .MPC80(TEMP) ''COMPRESS STAGE AND ADDRESS''
        END
        IF VASPI LAND MSKSSS4C
            THEN VASPI = 0
            DFILE = DFILE LOR MSKFPSR
        END
        IF VSNA1 EQ MSKSSHIG
            THEN DVMC7 = DVMC7 LOR MSKM7HIG
    
```

CLASP KERNEL 10 SWITCH SELECTOR PROCESSING

```

        GOTO SSEXIT
    END
    IF VSNA1 EQ MSKSSLOG
        THEN DVMC7 = DVMC7 LOR MSKM7LOG
        GOTO SSEXIT
    END
    IF VSNA1 EQ MSKSSOMG
        THEN DVMC7 = DVMC7 LOR MSKM7OMG
        GOTO SSEXIT
    END
    IF VSNA1 EQ MSKSSS4B
        THEN IF FBRNI
            THEN DVMC5 = DVMC5 LOR MSKM54B1
            ELSE DVMC6 = DVMC6 LOR MSKM68BR
        END
    END
    GOTO SSEXIT
MSS70.  IF FSSIO
        THEN DIRECT
            'RESET SS READ COMMAND'
        END
    END
    .SSTUPQ(KSSH3,1) 'SCHEDULE HUNG STAGE TEST, MSS05'
    .SS201 'ADVANCE TO NEXT SS'
    VSSW = KSSB1
    FHST = VHSTW LAND VSTG LXOR VSTG
    IF VSNA1 EQ MSKSSWVO
        THEN DVASW = DVASW LXOR MSKSSEC1
        DFWV = FALSE
        GOTO SSEXIT
    END
    IF VSNA1 EQ MSKSSWVC
        THEN DVASW = DVASW LXOR MSKSSECV
        DFWV = TRUE
        GOTO SSEXIT
    END
    IF VSNA1 EQ MSKSSSCC
        THEN DVDPM = DVDPM LOR MSKDIN9
    END
    GOTO SSEXIT
MSS80.  VSNA = VSCCA
    IF FSSIO
        THEN DIRECT
            'ISSUE STAGE AND COMPLEMENTED ADDRESS'
        END
    END
    .SSTUPQ(KSSB7,7) 'SCHEDULE READ COMMAND, MSS55'
    FOR I = 0 TO 41 'DELAY FOR DOM TELEMETRY'
        A = I
    END
    DIRECT
        'OUTPUT SS AND DO REGISTERS VIA DOM TELEMETRY'
    END
    GOTO SSEXIT
CLOSE  .SS201 'SS TABLE ADVANCE ROUTINE'

```

CLASP KERNEL 10 SWITCH SELECTOR PROCESSING

```

IF FTADV
    THEN SST1PTR = SST1PTR + 1
    ELSE SST2PTR = SST2PTR + 1
END
.SS210    'SET UP NEXT SWITCH SELECTOR'
EXIT      'SS201'
CLOSE     .SS210    'SS SELECTION AND SETUP ROUTINE'
IF FTADV    THEN GOTO SS2020
SS2160.   IF SSTABLE(0,SST2PTR) GQ 0    THEN GOTO SS2070
FCLS4 = FALSE
DVMC6 = DVMC6 LXOR MSKM6LUI
DVMC7 = DVMC7 LXOR MSKM7T6D
GOTO SS2090
    THEN VASPI = MSKSSS4C
    DVASW = DVASW LAND MSKSSWV
    SST1PTR = L'SSTSIVR'-L'SSTABLE'
    GOTO SS2020
END
IF VASPI LAND MSKSSCL3
    THEN SST1PTR, VASPI, VATRR = VSC3(0), VSC3(1), VSC3(2)
    GOTO SS2020
END
IF VASPI LAND MSKSSCL1
    THEN SST1PTR, VASPI, VATRR = VSC1(0), VSC1(1), VSC1(2)
    GOTO SS2020
END
VASPI, VATRR = 0
IF FT60P
    THEN SST1PTR = L'SSTTB5A'-L'SSTABLE'
    FT60P = FALSE
    ELSE SST1PTR = L'SSTTB5B'-L'SSTABLE'
END
GOTO SS2020
SS2030.   IF NOT FCLS4    THEN GOTO SS2040
SS2070.   IF SSTABLE(0,SST1PTR) + VATRR = KSS500MS GQ
    SSTABLE(0,SST2PTR) + VATR4
    THEN FTADV = FALSE
    VSSRT = SSTABLE(0,SST2PTR) + VATR4
    VSNA = SSTABLE(1,SST2PTR)
SS2090.   ELSE FTADV = TRUE
SS2040.   VSSRT = SSTABLE(0,SST1PTR) + VATRR
    VSNA = SSTABLE(1,SST1PTR)
END
SS2050.   VHSTW = VSNA RSH 2 LAND MSKSSSSB
EXIT      'SS210'
SSEXIT.   EXIT 'MSS00'
PROC .SSTUPD (= TIME) 'SS TIME UPDATE ROUTINE'
    DECLARE FIXED, TIME=2
    DIRECT
        'READ REAL TIME CLOCK INTO TEMP'
    END
    TIME, DVTRB = DVTGB + DVTRR + (TEMP - DVRTC LAND MSKRTC) RSH 2
EXIT      'SSTUPD'
PROC .SSTUPQ (BIAS, ID) 'UPDATE SS TIME AND SCHEDULE REQUESTED FUNC'
    DECLARE FIXED BIAS =2

```

```
DECLARE INTEGER, ID
  DIRECT
    ''READ REAL TIME CLOCK INTO TEMP''
  IF VASPI LAND MSKSSSPC
SS2020.  IF SSTABLE (0,SST1PTR) GQ 0  THEN GOTO SS2030  END
  END
  DGSSM = ID
  DVTRB = DVTGB + DVTRR + (TEMP - DVRTC LAND MSKRTC) RSH 2
  VSSTM = RIAS + DVTRR + (TEMP - DVRTC LAND MSKRTC) RSH 2
  DVSST = VSSTM + DVTMR
  IF NOT DFIL3
    THEN .EGP08  ''RESCHEDULE TIMER 1  (NOT CODED)''
  END
EXIT  ''SSTUPQ''
```

CLASP KERNEL 11 ATM TASK KEYING

```

PROC .TASKKEY (PRIORITY,TSKPTR) 'ATM TASK KEYING ROUTINE'
  DECLARE INTEGER,
    PRIORITY , 'PRIORITY LEVEL OF TASK BEING KEYED'
    TSKPTR   , 'LOCATION POINTER TO TASK BEING KEYED'
    I       , 'OVERFLOW TABLE POINTER CHAIN INDEX'
    J       , 'OVERFLOW TABLE OPEN-SLOT INDEX'
  ''
  ''PRIORITY CONTROL TABLE CONTAINS ONE ENTRY FOR EACH SYSTEM
  ''PRIORITY LEVEL.  FOR EACH ENTRY THERE ARE FIVE ITEMS.
  '' 1. LOCATION POINTER TO THE NEXT EXECUTABLE INSTRUCTION
  ''   OF THE TASK CURRENTLY ASSIGNED TO THAT PRIORITY LEVEL
  ''   OR ZERO IF NO TASK IS CURRENTLY ASSIGNED.
  '' 2. )
  '' 3. ) TASK REGISTER CONTENTS (INITIALLY SET TO ZERO).
  '' 4. )
  '' 5. POINTER TO THE BEGINNING OF THE PRIORITY OVERFLOW
  ''   TABLE CHAIN FOR THAT PRIORITY LEVEL.  A VALUE OF ZERO
  ''   INDICATES END OF CHAIN.
  ''
  '' NOTE THE NUMBER OF REGISTERS SAVED FOR A TASK WAS ARBITRARI-
  ''   LY CHOSEN FOR THIS EXAMPLE AND MAY BE ADJUSTED AS
  ''   REQUIRED.
  DECLARE INTEGER ATMPCT (5,10)
  ''
  ''THE PRIORITY OVERFLOW TABLE IS USED FOR KEYING TASKS ON A
  ''PRIORITY LEVEL WHICH IS CURRENTLY ASSIGNED TO ANOTHER TASK.
  ''THE ENTRIES ARE NOT ALLOCATED TO A FIXED PRIORITY BUT ARE
  ''ASSIGNED DYNAMICALLY AS REQUIRED.  ALL OVERFLOW ENTRIES FOR
  ''EACH PRIORITY LEVEL ARE CHAINED TOGETHER SUCH THAT THE TASKS
  ''CAN BE EXECUTED ON A FIRST-IN-FIRST-OUT BASIS.  EACH ENTRY
  ''CONSISTS OF TWO ITEMS.
  '' 1. POINTER TO NEXT ENTRY IN THE CHAIN.  A VALUE OF ZERO
  ''   INDICATES END OF CHAIN.
  '' 2. LOCATION POINTER TO THE BEGINNING OF THE TASK FOR
  ''   THAT ENTRY.  A VALUE OF ZERO INDICATES THAT THE ENTRY
  ''   IS CURRENTLY NOT ASSIGNED TO ANY TASK.
  DECLARE INTEGER ATMPOVFT(2,26)
  ''
  ''NOTE  SINCE AN END-OF-CHAIN INDICATOR HAS A VALUE OF ZERO,
  ''   THE FIRST ENTRY IN THE OVERFLOW TABLE CANNOT BE USED.
  ''
  LOCK      'INHIBIT ALL INTERRUPTS'
  ''
  ''IF THE REQUESTED PRIORITY LEVEL IS NOT CURRENTLY ASSIGNED,
  ''INITIALIZE THE ENTRY FOR THIS TASK.
  ''
  IF ATMPCT(0,PRIORITY) EQ 0
    THEN ATMPCT(0,PRIORITY) = TSKPTR
        ATMPCT(1,PRIORITY),
        ATMPCT(2,PRIORITY),
        ATMPCT(3,PRIORITY) = 0
  ''
  ''OTHERWISE, SEARCH FOR THE END OF THE OVERFLOW POINTER CHAIN.
  ''

```


PRECEDING PAGE BLANK NOT FILMED

HAL COMMON DATA DECLARATIONS

DECLARE SCALAR,

DKT1 ,
DVA1 ,
DVA2 ,
DVA3 ,
DVA4 ,
DVA5 ,
DVA6 ,
DVDT ,
DVEOF ,
DVFMC ,
DVFOM ,
DVFOR ,
DVMAS ,
DVMFR ,
DVMLR ,
DVTAS ,
DVTB ,
DVTI ,
DVVSQ ,
DV1MR ,
STEMP ;

DECLARE VECTOR,

DVCA ,
DVCC ,
DVD ,
DVDA ,
DVDR ,
DVDC ,
DVDM ,
DVF ,
DVG ,
DVRC ,
DVTH ;

DECLARE INTEGER,

DFLT ,
DGMLM ,
DGSSM ,
DGST2 ,
DKMIR CONSTANT (163) ,
DKRTC0VF CONSTANT (8192) ,
DKRTCSEC CONSTANT (4063) ,
DKTD CONSTANT (14) ,
DLPRL ARRAY(3) CONSTANT (50794,60952,101562) ,
DLPTL ARKAY(3) ,
DLTTL ARRAY(12) ,
DQST2 ,
DTBID ,
DVACT ,
DVDGS ,
DVERT ,
DVHDA ,
DVHDB ,
DVLRC ,
DVMLD ,

HAL COMMON DATA DECLARATIONS

```
DVMLT,  
DVM05,  
DVM06,  
DVP,  
DVPTG,  
DVRE ARRAY(3),  
DVRTC,  
DVSST,  
DVID,  
DVTEX,  
DVTGB,  
DVTMM,  
DVTMR,  
DVTRB,  
DVTRR,  
DVTRS,  
DVTT1,  
DV2TG,  
EPTINDX,  
GST1M,  
ITEMP,  
VTOLD;  
DECLARE BIT(1),  
APSTAT ,  
ARSTAT ,  
CC1STAT,  
CSS1AT ,  
CTSTAT ,  
DFACQ ,  
DFDBF ,  
DFDTL ,  
DFIL1 ,  
DFIL2 ,  
DFIL3 ,  
DFPHC ,  
DFSMC ,  
DFTBCEP,  
DFTUP ,  
DFWV ,  
DFZER ,  
DKAPI ARRAY(4),  
DTSTAT ,  
DPSTAT ,  
DVSTAT ,  
EBSTAT ,  
HSSTAT ,  
IGSTAT ,  
MSSTAT ,  
NESTAT ,  
OGSTAT ,  
PASTAT ,  
PGSTAT ,  
PPSTAT ARRAY(3),  
RSSTAT ,  
SASTAT ,
```


HAL COMMON DATA DECLARATIONS

```

TB1STAT,
TB57STAT,
TCSTAT ,
TGSTAT ,
TTSTAT ,
T2STAT ARRAY (11);
DECLARE BIT(26),
DFILE,
DFMDI,
DVAC ARRAY(3),
DVASW,
DVDPM,
DVEMR,
DVICR,
DVIH ,
DVLDR,
DVMC4,
DVMC5,
DVMC6,
DVMC7,
VTMC,
BTEMP;
DECLARE BIT(26) CONSTANT,
MSKACCELA      (OCT'777700000'),
MSKACCELB      (OCT'000017776'),
MSKDCSCOMP     (OCT'774000000'),
MSKDCSDO       (OCT'000040000'),
MSKDCSER       (OCT'000077776'),
MSKDCSMC       (OCT'770000000'),
MSKDCSMODE     (OCT'000000020'),
MSKDCSSB       (OCT'004000000'),
MSKDCSTERM     (OCT'200000000'),
MSKDIN9        (OCT'000004000'),
MSKEMRLADR     (OCT'000001000'),
MSKERRORTAG    (OCT'000070000'),
MSKFMDREP      (OCT'000100000'),
MSKFPSCORD     (OCT'100000000'),
MSKFPSINT2     (OCT'000040000'),
MSKFPSISSA     (OCT'001000000'),
MSKGIMBALA     (OCT'377700000'),
MSKICRBG       (OCT'000000020'),
MSKICRCA       (OCT'000040000'),
MSKICRSSCB     (OCT'000010000'),
MSKICRSWG      (OCT'000002000'),
MSKINT         (OCT'157740000'),
MSKMC4AMF      (OCT'000000100'),
MSKMC4SSCB     (OCT'000001000'),
MSKMC54B1I     (OCT'000000100'),
MSKMC6D04      (OCT'000000400'),
MSKMC6LUI      (OCT'000010000'),
MSKMC6TB6A     (OCT'000002000'),
MSKMC6TB6B     (OCT'000000010'),
MSKMC6TB6C     (OCT'000000040'),
MSKMC68BRI     (OCT'400000000'),
MSKMC7HIG      (OCT'004000000'),

```

4

HAL COMMON DATA DECLARATIONS

```

MSKMC7LOG      (OCT'010000000'),
MSKMC7OMG      (OCT'020000000'),
MSKMC7T6D      (OCT'100000000'),
MSKRTC         (OCT'000037776'),
MSKRTCRESET    (OCT'007777540'),
MSKSCCO        (OCT'000100000'),
MSKSSACQU      (OCT'000000004'),
MSKSSCLS1      (OCT'000003770'),
MSKSSCLS3      (OCT'077774000'),
MSKSSCL1       (OCT'040000000'),
MSKSSCL3       (OCT'100000000'),
MSKSSDCS       (OCT'500000000'),
MSKSSDCT       (OCT'405400000'),
MSKSSSECSV     (OCT'002000000'),
MSKSSSECS1     (OCT'001000000'),
MSKSSGNSS      (OCT'040000000'),
MSKSSHIG       (OCT'100720000'),
MSKSSHS        (OCT'003770000'),
MSKSSLI        (OCT'000000002'),
MSKSSLOG       (OCT'100520000'),
MSKSSNSEND     (OCT'377777776'),
MSKSSOM0       (OCT'100070000'),
MSKSSREAD      (OCT'400000000'),
MSKSSRESET     (OCT'200000000'),
MSKSSSBHI      (OCT'010000000'),
MSKSSSBLO      (OCT'020000000'),
MSKSSSBOM      (OCT'004000000'),
MSKSSSCC       (OCT'100310000'),
MSKSSSIVB      (OCT'020230000'),
MSKSSSNA       (OCT'135770000'),
MSKSSSPEC      (OCT'200000000'),
MSKSSSSB       (OCT'174000000'),
MSKSSS4C0      (OCT'400000000'),
MSKSSS4C1      (OCT'000000400'),
MSKSSSTB6A     (OCT'000002000'),
MSKSSSTB6B     (OCT'000001000'),
MSKSSSTB6C     (OCT'100000000'),
MSKSSSTB6D     (OCT'000200000'),
MSKSSST3A      (OCT'000400000'),
MSKSSST6C      (OCT'004000000'),
MSKSSWV        (OCT'003000000'),
MSKSSWVC       (OCT'101050000'),
MSKSSWVO       (OCT'101450000'),
MSKSSZFSF      (OCT'002000000'),
MSKTMCO        (OCT'700000000'),
MSKTMCM1       (OCT'710000000'),
MSKTMCM2       (OCT'720000000'),
MSKTMCM3       (OCT'730000000'),
MSKTMCM4       (OCT'740000000'),
MSKT2INT       (OCT'100000000');

```

HAL UTILITY ROUTINES

```

UTR00: PROGRAM; /*TELEMETRY DELAY FOR MODE REG SETTING OF 70*/
E      .
      VTMC = MSKTMCO;
      CALL UTR0; /*PERFORM DELAY AS REQUIRED*/
CLOSE  UTR00;
UTR01: PROGRAM; /*TELEMETRY DELAY FOR MODE REG SETTING OF 71*/
E      .
      VTMC = MSKTMCO;
      CALL UTR0; /*PERFORM DELAY AS REQUIRED*/
CLOSE  UTR01;
UTR02: PROGRAM; /*TELEMETRY DELAY FOR MODE REG SETTING OF 72*/
E      .
      VTMC = MSKTMCO;
      CALL UTR0; /*PERFORM DELAY AS REQUIRED*/
CLOSE  UTR02;
UTR03: PROGRAM; /*TELEMETRY DELAY FOR MODE REG SETTING OF 73*/
E      .
      VTMC = MSKTMCO;
      CALL UTR0; /*PERFORM DELAY AS REQUIRED*/
CLOSE  UTR03;
UTR04: PROGRAM; /*TELEMETRY DELAY FOR MODE REG SETTING OF 74*/
E      .
      VTMC = MSKTMCO;
      CALL UTR0; /*PERFORM DELAY AS REQUIRED*/
CLOSE  UTR04;
UTR05: PROGRAM; /*TELEMETRY DELAY FOR LEVEL 0*/
      DECLARE INTEGER,
          KTELBIAS CONSTANT (2),
          VTIM;
TR00:  /*INHIBIT ALL INTERRUPTS EXCEPT TCL*/;
      READ (CLOCK) VTIM;
      IF BIT (VTIM - DVTD) >= DKTD THEN GO TO TR05;
S      14 TO 26
C      RELEASE PREVIOUSLY ENABLED INTERRUPTS
      GO TO TR00;
E
TR05:  WRITE (MODREG) VTMC;
      DVTD = VTIM + KTELBIAS;
CLOSE  UTR05;
UTR30: PROGRAM; /*TELEMETRY DELAY FOR INTERRUPT LEVEL 3*/
      DECLARE INTEGER,
          KTELBIAS CONSTANT (2),
          VTIM;
TR35:  READ (CLOCK) VTIM;
      IF BIT (VTIM - DVTD) < DKTD THEN GO TO TR35;
S      14 TO 26
F
      WRITE (MODREG) MSKTMCO;
      DVTD = VTIM + KTELBIAS;
CLOSE  UTR30;
UTR24: PROGRAM; /*TELEMETRY DELAY FOR INTERRUPT LEVEL 2*/
      DECLARE INTEGER,
          KTELBIAS CONSTANT (2),
          VTIM;
TR20:  /*INHIBIT ALL INTERRUPTS EXCEPT TLC*/;

```

HAL UTILITY ROUTINES

```
      READ (CLOCK) VTIM;
      IF BIT      (VTIM - DVTD) >= DKTD  THEN GO TO TR25;
S      14 TO 26
C      ENABLE TIMER 1 INTERRUPT
      GO TO TR20;
E
TR25:  WRITE (MODREG) MSKTC4;
      DVTD = VTIM + KTELBIAS;
CLOSE  UTR24;
```

HAL KERNEL 1 INITIALIZATION

```

EGP0: PROGRAM; /*SYSTEM INITIALIZATION*/
C
C   INHIBIT ALL INTERRUPTS
C
S   READ(XACC) VOAC ;
      1
S   READ(YACC) VOAC ;
      2
S   READ(ZACC VOAC ;
      3
E   READ(CLOCK) DVACT;
      .
      IF DFMDI AND MSKFMDREP = 0
      THEN DO;
          SCHEDULE CLOCK SYNC ON T1INT;
          WRITE (TIM1) 1;
          ENABLE TIMER 1 INTERRUPT
          WAIT FOR T1INT;
      END;
      DVRTC,DVTEX,VPPOT = DVACT;
      DVTMM,DVTRR,DVERT,DVTGR,DVTRS,DVTMR,DTBID,VTD = 0;
      .
      DFIL1, DFIL2, DFIL3 = TRUE;
      CALL EGP1; /*ACTIVATE INTERRUPT PROCESSOR*/
C
C   ENABLE TLC INTERRUPT
C
      .
      FGNC = FALSE;
      DVSST = 1.E10;
      DVMLT, DVMLD = DKMIR;
      CALL EGP15; /*SCHEDULE FIRST TIMER 1 FUNCTION*/
      DVP = 1;
      CALL GP002; /*PASS CONTROL TO PHASE ACTIVATION ROUTINE*/
CLOCK SYNC: TASK; /*RESPONSE TO TIMER 1 INTERRUPT*/
      READ (CLOCK) DVACT;
C
C   INHIBIT TIMER 1 INTERRUPT
C
      CLOSE CLOCK SYNC;
      CLOSE EGP0;
MPA00: PROGRAM; /*PHASE TERMINATION*/
E
      .
      DFPHC = TRUE;
C
C   INHIBIT ALL INTERRUPTS
C
      .
      WRITE(TIM2) MSKRTC; /*LOAD TIMER 2 WITH A LARGE VALUE*/
E
      WRITE(ISR) MSKT2INT; /*RESET ANY PENDING TIMER 2 INTERRUPT*/
E
      .
      DVIS1 = DVIH AND NOT MSKT2INT;
F
      .
      DFIL1, DFIL2 = TRUE;

```

HAL KERNEL 1 INITIALIZATION

```

C
C   ENABLE TLC AND TIMER 1 INTERRUPTS
C
CALL GP002; /*ACTIVATE THE NEXT MISSION PHASE*/
CLOSE MPA00;
GP002:  PROGRAM; /*MISSION PHASE CONTROL AND ACTIVATION ROUTINE*/
        DECLARE FGNC BIT(1);
        DECLARE VTD INTEGER;
GP0020: IF DVP > 4 THEN : /*HALT WHEN EXECUTION IS COMPLETED*/
E
        IF DKAPI
S
            DVP-1
            THEN DO;
                CALL EGP20; /*START PHASE TIME REFERENCE*/
                DO CASE DVP;
                    GO TO INP13;
                    GO TO INP24;
                    GO TO INP13;
                    GO TO INP24;
                END;
            END;
        ELSE DO;
            DVP = DVP + 1;
            GO TO GP0020;
        END;
E
INP13:  .ARSTAT, .APSTAT, .DPSTAT, .NESTAT, .CC1STAT, .MSSTAT, .EBSTAT = TRUE;
E
        .SASTAT, .DVSTAT, .TCSTAT, .PASTAT, .TTSTAT, .IGSTAT, .HSSTAT, .OGSTAT, .TGSTAT,
E
        .RSSTAT, .CSSTAT, .TB1STAT, .TB57STAT, .PGSTAT = FALSE;
        [DLPTL] = 0;
E
        [.PPSTAT] = FALSE;
E
        [.T2STAT] = FALSE;
E
        .T2STAT = TRUE;
S
        1
        CALL MIN00; /*PERFORM PHASE 1/3 APPLIC PGM INIT (NOT CODED)*/
        CALL EGP18; /*SCHEDULE NEXT TIMER 2 FUNCTION*/
E
        .DFIL1, .DFIL2, .DFIL3, .DFPHC = FALSE;
C
        RELEASE PREVIOUSLY ENABLED INTERRUPTS
C
        CALL NONINTSEQ1; /*PASS CONTROL TO PHASE 1/3 NON-INTERRUPT SEQ*/
E
INP24:  .CTSTAT, .DTSTAT = FALSE;
        [DLPTL] = 0;
E
        [.PPSTAT] = TRUE;
E
        [.T2STAT] = BIT(0,1,1,1,0,0,1,1,1,1,1);
        CALL MIN10; /*PERFORM PHASE 2/4 APPLIC PGM INIT (NOT CODED)*/

```

HAL KERNEL 1 INITIALIZATION

E
C
C
C

```
CALL EGP18; /*SCHEDULE NEXT TIMER 2 FUNCTION*/  
DFIL1, DFIL2, DFIL3, DFPHC = FALSE;  
RELEASE PREVIOUSLY ENABLED INTERRUPTS  
CALL NONINTSEQ2; /*PASS CONTROL TO PHASE 2/4 NON-INTERRUPT SEQ*/  
CLOSE GP002;
```

HAL KERNEL 2 INTERRUPT PROCESSING

```

EGP1:  PROGRAM; /*INTERRUPT PROCESSOR*/
        SCHEDULE EGPTLC ON TLCINT;
        SCHEDULE EGPT1 ON T1INT;
        SCHEDULE EGPT2 ON T2INT;
        SCHEDULE EGPEX2 ON EX2INT;
        SCHEDULE EGPEX4 ON EX4INT;
        SCHEDULE EGPEX5 ON EX5INT;
        SCHEDULE EGPEX6 ON EX6INT;
        SCHEDULE EGPEX8 ON EX8INT;
        RETURN; /* EGP1 */

C
EGPTLC: TASK; /* TLC INTERRUPT PROCESSOR */
C
C      INHIBIT ALL INTERRUPTS EXCEPT TLC
C
        READ(CLOCK) DVTEX;
        DFIL2, DFIL3 = TRUE;
        CALL MTS00; /*PROCESS TLC INTERRUPT*/
C      THE TLC APPLICATION PROGRAM DOES NOT RETURN CONTROL
CLOSE EGPTLC;
C
EGPT1:  TASK; /* TIMER 1 INTERRUPT PROCESSOR */
C
C      INHIBIT ALL INTERRUPTS EXCEPT TLC
C
        READ(CLOCK) DVTT1;
        DFIL3 = TRUE;
        DO CASE GST1M;
            CALL MML00; /*PROCESS MINOR LOOP FOR FLIGHT SIM*/
            CALL MML20; /*PROCESS NORMAL MINOR LOOP*/
            CALL MSS00; /*PROCESS SWITCH SELECTOR*/
        END;
        CALL EGP15; /*SCHEDULE NEXT TIMER 1 FUNCTION*/
        DFIL3 = FALSE;

C
C      RELEASE PREVIOUSLY ENABLED INTERRUPTS
C
CLOSE EGPT1;
C
EGPT2:  TASK; /* TIMER 2 INTERRUPT PROCESSOR */
C
C      INHIBIT TIMER 2 INTERRUPT
C
        DFIL1 = TRUE;
        DO CASE DGST2;
            CALL MUM00; /*TIME UPDATE (NOT CODED)*/
            CALL MLR10; /*LADDER RAMP PROCESSOR (NOT CODED)*/
            CALL MEP00; /*EVENTS PROCESSOR */
            CALL MTT10; /*TIME TILT GUIDANCE (NOT CODED)*/
            CALL MNU00; /*NAVIGATION UPDATE IMPLEMENT (NOT CODED)*/
            CALL MEE00; /*TIME BASE 8 ENABLE (NOT CODED)*/
            CALL MCM00; /*PHASE 2/4 CONTROL MODULE (NOT CODED)*/
            CALL MCM10; /*PHASE 2/4 CONTROL MODULE (NOT CODED)*/
            CALL MCM20; /*PHASE 2/4 CONTROL MODULE (NOT CODED)*/
            CALL MEPWM; /*WATER METHANOL ACTIVATE (NOT CODED)*/
        END;
    
```


HAL KERNEL 2 INTERRUPT PROCESSING

```

        CALL MER00; /*EXTRA ACCELEROMETER READ (NOT CODED)*/
END;
CALL EGP18; /*SCHEDULE NEXT TIMER 2 FUNCTION*/
DFIL1 = FALSE;
C
C
C
CLOSE EGPT2;
C
EGPEX2; TASK; /* EXTERNAL 2 INTERRUPT PROCESSOR */
C
C
C
INHIBIT ALL INTERRUPTS EXCEPT TLC
READ(CLOCK) DVTEX;
DFIL2, DFIL3 = TRUE;
CALL MDP28; /*SC INITIATION OF S2/S4B SEPARATION (NOT CODED)*/
DFIL2, DFIL3 = FALSE;
C
C
C
RELEASE PREVIOUSLY ENABLED INTERUPTS
C
CLOSE EGPEX2;
C
C
EGPEX4; TASK; /* EXTERNAL 4 INTERRUPT PROCESSOR */
C
C
C
INHIBIT ALL INTERRUPTS EXCEPT TLC
READ(CLOCK) DVTEX;
DFIL2, DFIL3 = TRUE;
CALL MTB50; /*S4B ENGINE OUT (NOT CODED)*/
DFIL2, DFIL3 = FALSE;
C
C
C
RELEASE PREVIOUSLY ENABLED INTERUPTS
C
CLOSE EGPEX4;
C
C
EGPEX5; TASK; /* EXTERNAL 5 INTERRUPT PROCESSOR */
C
C
C
INHIBIT ALL INTERRUPTS EXCEPT TLC
READ(CLOCK) DVTEX;
DFIL2, DFIL3 = TRUE;
CALL MTB30; /*S1C OUTBOARD ENGINE OUT (NOT CODED)*/
DFIL2, DFIL3 = FALSE;
C
C
C
RELEASE PREVIOUSLY ENABLED INTERUPTS
C
CLOSE EGPEX5;
C
C
EGPEX6; TASK; /* EXTERNAL 6 INTERRUPT PROCESSOR */
C
C
C
INHIBIT ALL INTERRUPTS EXCEPT TLC
READ(CLOCK) DVTEX;
DFIL2, DFIL3 = TRUE;
CALL MTB40; /*S2 PROPELLANT DEPLETION (NOT CODED)*/
DFIL2, DFIL3 = FALSE;

```

HAL KERNEL 2 INTERRUPT PROCESSING

```

C
C      RELEASE PREVIOUSLY ENABLED INTERUPTS
C
C      CLOSE      EGPEX6;
C
C      EGPEX8;    TASK;      /* EXTERNAL 8 INTERRUPT PROCESSOR */
C
C      INHIBIT ALL INTERRUPTS EXCEPT TLC
      READ(CLOCK) DVTEX;
      DFIL2, DFIL3 = TRUE;
      CALL MDS00; /*PROCESS DIGITAL COMMAND SYSTEM INPUT*/
      DFIL2, DFIL3 = FALSE;
C
C      RELEASE PREVIOUSLY ENABLED INTERUPTS
C
C      CLOSE      EGPEX8;
C
      CLOSE EGP1;
EGP15:  PROGRAM; /*TIMER 1 SCHEDULER*/
      DECLARE KT1BIAS INTEGER CONSTANT (2);
      READ(CLOCK) ITEMP;
      ITEMP = ITEMP - DVRTC;
      IF ITEMP < 0 THEN ITEMP = ITEMP + DKRTC0VF;
      ITEMP = DVTMM + ITEMP/4 + KT1BIAS;
      IF DVMLT <= ITEMP
          THEN DO;
              ITEMP = 1 ;
              GST1M = DGMLM;
          END;
      ELSE IF DVMLT <= DVSST
          THEN DO;
              ITEMP = 2 (DVMLT - ITEMP);
              GST1M = DGMLM;
          END;
      ELSE DO;
          GST1M = 3;
          IF DVSST <= ITEMP
              THEN ITEMP = 1;
          ELSE ITEMP = 2 (DVSST - ITEMP);
      END;
      WRITE(TIM1) ITEMP;
      CLOSE EGP15;
EGP18:  PROGRAM; /*TIMER 2 SCHEDULER*/
      DECLARE INTEGER CONSTANT, KT2BIAS (3),
          K49SEC (4063);
      DGST2 = 1;
      DV2TG = DVTMM + K49SEC;
      DO FOR I = 1 TO 11;
E
          IF NOT T2STAT THEN GO TO TS210;
S
          I
          IF DLTTL > DV2TG THEN GO TO TS210;
S
          I
          DGST2 = I + 1;

```

HAL KERNEL 2 INTERRUPT PROCESSING

```

                DV2TG = DLTTL ;
S                I
TS210:         END;
C
C                INHIBIT ALL INTERRUPTS EXCEPT T2 AND TLC
C
                IF DVT2G <= DVTMM THEN GO TO T2S20;
                READ(CLOCK) ITEMP;
                ITEMP = ITEMP - DVRTC;
                IF ITEMP < 0 THEN ITEMP = ITEMP + DKRTCQVF;
                ITEMP = (DV2TG - DVTMM + DVERT - ITEMP/4 -KT2BIAS)/2;
                IF ITEMP <=0 THEN
T2S20:         ITEMP = 1;
                WRITE(TIM2) ITEMP;
C
C                RELEASE PREVIOUSLY ENABLED INTERRUPTS
C
                CLOSE EGP18;
EGP20:         PROGRAM; /*SYSTEM TIME UPDATE ROUTINE*/
C
C                INHIBIT ALL INTERRUPTS EXCEPT TLC
C
                READ(CLOCK) ITEMP;
                ITEMP1 = ITEMP - DVRTC;
                DVERT = BIT      (ITEMP1);
S                25 TO 26
                IF ITEMP1 < 0 THEN ITEMP1 = ITEMP1 + DKRTCQVF;
                DVTMM = DVTMM + ITEMP1/4;
                DVRTC = ITEMP - DVERT;
                DVTRR = DVTMM - DVTMR;
E
                IF DFIL3
                THEN ;
E
                ELSE IF DFIL2
                THEN /*RELEASE TIMER 1 INTERRUPT*/ ;
E
                ELSE IF DFIL1
                THEN /*RELEASE PREVIOUSLY ENABLED
                INTERRUPTS EXCEPT TIMER 2*/ ;
                ELSE /*RELEASE PREV ENABLED INT*/;
                CLOSE EGP20;

```

HAL KERNEL 3 NON-INTERRUPT SEQUENCER

```

NONINTSEQ1: PROGRAM; /*NON-INTERRUPT SEQUENCER FOR PHASES 1 AND 3 */
E
NIS1:   IF .ARSTAT      THEN DO;
        CALL MAR00;      /*ACCELEROMETER READ*/
        CALL PERPROC; END;
E
        IF .SASTAT      THEN DO;
        CALL MSA00;      /*SIMULATED ACCEL      (NOT CODED)*/
        CALL PERPROC; END;
E
        IF .APSTAT      THEN DO;
        CALL MAPO0;      /*ACCELEROMETER PROCESSING*/
        CALL PERPROC; END;
E
        IF .DVSTAT      THEN DO;
        CALL MDV00;      /*F/M CALCULATIONS      (NOT CODED)*/
        CALL PERPROC; END;
E
        IF .DPSTAT      THEN DO;
        CALL MDPU0;      /*DISCRETE PROCESSOR    (NOT CODED)*/
        CALL PERPROC; END;
E
        IF .NESTAT      THEN DO;
        CALL MNE00;      /*BOOST NAVIGATION      (NOT CODED)*/
        CALL PERPROC; END;
E
        IF .TCSTAT      THEN DO;
        CALL MTCU0;      /*RESTART CALCULATIONS  (NOT CODED)*/
        CALL PERPROC; END;
E
        IF .PASTAT      THEN DO;
        CALL MPA00;      /*PHASE ACTIVATOR*/
        CALL PERPROC; END;
E
        IF .TTSTAT      THEN DO;
        CALL MIT00;      /*TIME TILT GUIDANCE    (NOT CODED)*/
        CALL PERPROC; END;
E
        IF .CC1STAT     THEN DO;
        CALL MCC10;      /*CHI COMPUTATIONS      (NOT CODED)*/
        CALL PERPROC; END;
E
        IF .IGSTAT      THEN DO;
        CALL MIG00;      /*ITERATIVE GUIDANCE MODE*/
        CALL PERPROC; END;
E
        IF .HSSTAT      THEN DO;
        CALL MHS00;      /*S4B CUTOFF PREDICTION (NOT CODED)*/
        CALL PERPROC; END;
E
        IF .OGSTAT      THEN DO;
        CALL MOG00;      /*ORBITAL GUIDANCE      (NOT CODED)*/
        CALL PERPROC; END;
E
        IF .TGSTAT      THEN DO;

```

HAL KERNEL 3 NON-INTERRUPT SEQUENCER

```

CALL MTG00;          /*TARGET UPDATE      (NOT CODED)*/
CALL PERPROC; END;
E
IF RSSTAT THEN DO;
CALL MRS00;          /*TIME-TO-GO TO RESTART (NOT CODED)*/
CALL PERPROC; END;
E
IF CSSTAT THEN DO;
CALL MCS00;          /*TIME BASE 6 CHECK      (NOT CODED)*/
CALL PERPROC; END;
E
IF TB1STAT THEN DO;
CALL MTR10;          /*TIME BASE 1            (NOT CODED)*/
CALL PERPROC; END;
E
IF TB57STAT THEN DO;
CALL MTB57;          /*TIME BASE 5/7          (NOT CODED)*/
CALL PERPROC; END;
E
IF MSSTAT THEN DO;
CALL MMS00;          /*MINOR LOOP SUPPORT     (NOT CODED)*/
CALL PERPROC; END;
E
IF PGSTAT THEN DO;
CALL MPG00;          /*SIM PLATFORM GIM ANGLE (NOT CODED)*/
CALL PERPROC; END;
E
IF EBSTAT THEN DO;
CALL MER00;          /*ETC/BTC                (NOT CODED)*/
CALL PERPROC; END;
GO TO NIS1;
CLOSE NONINTSEQ1;
NONINTSEQ2: PROGRAM; /*NON-INTERRUPT SEQUENCER FOR PHASES 2 AND 4 */
E
NIS2: IF CTSTAT THEN DO;
CALL MCT00; /* DATA COMPRESSION TELEMETRY (NOT CODED)*/
CALL PERPROC; END;
E
IF DTSTAT THEN
CALL MDT00; /*SECTOR DUMP TELEMETRY (NOT CODED)*/
CALL PERPROC; /*INSURE PERIODIC PROCESSOR GETS EXECUTED*/
GO TO NIS2;
CLOSE NONINTSEQ2;

```

HAL KERNEL 4 PERIODIC PROCESSOR

```

PERPROC: PROGRAM; /*PERIODIC PROCESSOR*/
DECLARE VPPOT INTEGER;
READ(CLOCK) ITEMP;
DVPTG = (ITEMP - VPPOT)/4;
VPPOT = ITEMP;
IF DVPTG < 0 THEN DVPTG = DVPTG + DKRTC0VF/4;
DO FOR I = 1 TO 3;
E
    IF NOT PPSTAT THEN GO TO PP20;
S
    DLPTL = DLPTL + DVPTG;
S
    IF DLPTL < DLPRL THEN GO TO PP20;
S
    DO CASE I;
        CALL MPC50; /*PERFORM 50 DATA COMPRESS(NOT CODED)*/
        CALL MPC60; /*PERFORM 60 DATA COMPRESS(NOT CODED)*/
        CALL MPC99; /*PERFORM 100 DATA COMPRESS(NOT CODED)*/
    END;
    DLPTL = 0;
S
    PP20: END;
CLOSE PERPROC;

```

HAL KERNEL 5 EVENTS PROCESSOR

```

MEP00; PROGRAM; /*EVENTS PROCESSOR (TIMER 2 ENTRY)*/
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
EVENTS PROCESSOR TABLE

ONLY A PORTION OF THE TABLE (THROUGH TIME BASE 3)
HAS BEEN CODED.

EACH ENTRY CONSISTS OF TWO WORDS:
1. AN INDEX IDENTIFYING THE APPLICATION MODULE TO
PERFORM PROCESSING FOR THE EVENT.
2. EVENT EXECUTION TIME (IN TENTHS OF A SECOND).

NOTE: AN ENTRY INDEX WITH A VALUE OF ZERO IS EITHER SET
DYNAMICALLY IN REAL TIME OR IS USED TO DISABLE
THE EVENTS PROCESSOR FOR THE REMAINDER OF A TIME
BASE
DECLARE EPTABLE ARRAY(2,131) INTEGER INITIAL
( 1 , 0 , /* START OF TIME BASE 0 TABLE */
0 , 160 ,
0 , 170 ,
0 , 175 ,
0 , 0 ,
2 , 0 , /*START OF TIME BASE 1 TABLE*/
3 , 10 ,
0 , 60 ,
4 , 90 ,
5 , 100 ,
0 , 140 ,
6 , 1347 ,
0 , 0 ,
7 , 0 , /*START OF TIME BASE 2 TABLE*/
0 , 0 ,
0 , 184 ,
0 , 275 ,
0 , 0 , /*START OF TIME BASE 3 TABLE*/
8 , 0 ,
9 , 0 ,
10 , 0 ,
0 , 0 ,
11 , 0 ,
12 , 0 ,
13 , 14 ,
14 , 44 ,
15 , 44 ,
16 , 67 ,
0 , 67 ,
17 , 67 ,
18 , 406 ,
19 , 406 ,
20 , 586 ,
21 , 606 ,
0 , 2990 ,
0 , 3550 ,
0 , 3885 ,
0 , 0);

```

HAL KERNEL 5 EVENTS PROCESSOR

```

E
EP00: IF DFTBCEP THEN
EP004A: DO;
E
      DFTBCEP = FALSE;
      GO TO EP02;
      END;
DO CASE EPTABLE
S      1,EPTINDX
      CALL LE285; /*SCHEDULE WATER METHANOL (NOT CODED)*/
      CALL LE25; /*TIME BASE 1 SETUP (NOT CODED)*/
      CALL LE30; /*COMMAND INIT OF YAW MANEUVER (NOT CODED)*/
      CALL LE35; /*COMMAND TERM OF YAW MANEUVER (NOT CODED)*/
      CALL LE40; /*SET ACCEL REASON TEST CONST (NOT CODED)*/
      CALL LE50; /*START TIME BASE 2 (NOT CODED)*/
      CALL LE55; /*DISABLE THRUST CONSTRAINT (NOT CODED)*/
      CALL LE75; /*TIME BASE 3 SETUP (NOT CODED)*/
      CALL LE70; /*SET ACCEL REASON TEST CONST (NOT CODED)*/
      CALL LE250; /*CHNG GIMB REASON TEST CONST (NOT CODED)*/
      CALL LE355; /*DISABLE TIME TILT (NOT CODED)*/
      CALL LE365; /*F/M UNCERT FOR THUST MISALIN (NOT CODED)*/
      CALL LE82; /*ENABLE DIN 22 AND INT 2 (NOT CODED)*/
      CALL LE100; /*SET ACCEL RACKUP PROFILE (NOT CODED)*/
      CALL LE95; /*SET ACCEL REASON TEST CONST (NOT CODED)*/
      CALL LE90; /*ENABLE DIN 19 (NOT CODED)*/
      CALL LE96; /*ENQUEUE F/M CALC, SMOOTHING (NOT CODED)*/
      CALL LE105; /*ENQUEUE IGM (NOT CODED)*/
      CALL LE115; /*SET MINOR LOOP PARAMETERS (NOT CODED)*/
      CALL LE111; /*SET SMC FLAG (NOT CODED)*/
      CALL LE110; /*ENQUEUE SMC (NOT CODED)*/
      END;
EP01: ;
C
C INHIBIT ALL INTERRUPTS EXCEPT TLC
C
E
      IF DFTBCEP THEN GO TO EP04A;
      EPTINDX = EPTINDX + 1;
      DQST2 = 4; /*SET INDEX FOR EP ENTRY IN T2 SCHED CONTROL TABLE*/
      IF EPTABLE
S      1,EPTINDX
      THEN DO;
      IF EPTABLE
S      2,EPTINDX
      THEN DO;
C
C RELEASE PREVIOUSLY ENABLED INTERRUPTS
C
      GO TO EP00;
      END;
      VTOLD = EPTABLE
S      2,EPTINDX
      ;
      DLTTL
S      = DVTMR + VTOLD DKRTCSEC/40;
      DQST2
S
      END;

```


HAL KERNEL 5 EVENTS PROCESSOR

```

ELSE DO;
E      .
      T2STAT      = FALSE;
S      DQST2
      IF NOT DFIL1
          THEN CALL EGP07; /*RESCHEDULE T2 (NOT CODED)*/
      END;
EP02:  ;
C      .
C      RELEASE PREVIOUSLY ENABLED INTERRUPTS
C      .
      CLOSE MEP00;
MEP05: PROGRAM; /*EVENTS PROCESSOR (TIME BASE CHANGE ENTRY)*/
      DECLARE ARRAY(10) EPTTBINDX INTEGER CONSTANT
          (1,6,14,19,39,56,72,94,108,111);
      EPTINDX = EPTTBINDX - 1;
S      DTRID+1
      CALL MEP10;
      CLOSE MEP05;
MEP10: PROGRAM; /*EVENTS PROCESSOR (RESCHEDULE ENTRY)*/
      EPTINDX = EPTINDX + 1;
      DQST2 = 4; /*SET INDEX FOR EP ENTRY IN T2 SCHED CONTROL TABLE*/
      IF EPTABLE
          THEN DO;
S      .
          1,EPTINDX
          THEN DO;
S      .
          VTOLD = EPTABLE
          .
          2,EPTINDX
          DLTTL      = DVTMR + VTOLD DKRTCSEC/40;
S      DQST2
E      .
          T2STAT      = TRUE;
S      DQST2
      END;
E      .
      ELSE T2STAT      = FALSE;
S      DQST2
      IF NOT DFIL1 THEN CALL EGP07; /*RESCHED TIMER2(NOT CODED)*/
      CLOSE MEP10;

```

```

MIG00: PROGRAM; /*ITERATIVE GUIDANCE MODE*/
DECLARE BIT(1) ,
CHI_BAR_STEERING,
PHASE,
REITERATE,
SMC_FLAG,
S4BURN;
DECLARE SCALAR,
COSTHETA,          DELTAL3,          DELTA2,
DPHII,             DPHIT,           EPSILON2,
EPSILON3,          GT,             J1,
J12,              J2,             J3,
J3P,
KCCT4 CONSTANT (1.53),
KCCT8 CONSTANT (1.55),
KMU CONSTANT (-.39860320E15),
KT CONSTANT (.48497964E-7),
K1,              K2,              K3,
K4,              LYP,             L1,
L12,             L2,             L3,
L3P,             PHII,            PHIT,
PHIT,           P1,             P12,
P2,             Q1,             Q12,
Q2,             R,              ROVEX3,
RT,             SINTHETA,        S1,
S12,           S2,             TAU1,
TAU2,          TAU3,            TC1,
THETAT,       TSTAR,           T1C,
T1I,          T2I,            T3I,
U1,           U12,           U2,
V,            VEX1,          VEX2,
VEX3,         VT;
DECLARE VECTOR,
DELTAVVP,     GS,             GV,
GVSTAR,       GVT,            RS,
RV,           RVT,            R4,
VS,           VV,             VVT,
V4;
DECLARE MATRIX,
MS4,
M4V;

```

C
C
C
C
C
C
C
C
C
C
C
E

DUE TO THE SIZE OF IGM, ONLY A SECTION OF IT HAS BEEN CODED. PART OF THE GUIDANCE COMPUTATIONS HAVE BEEN SELECTED TO DEMONSTRATE MATHEMATICAL OPERATIONS. THE PHASING PORTION OF IGM HAS NOT BEEN CODED SINCE SIMILAR CAPABILITIES ARE ILLUSTRATED BY OTHER KERNELS.

IG251 - IGM GUIDANCE PARAMETERS COMPUTATIONS

ROTATE POSITION AND VELOCITY INTO TARGET PLANE

```

IG253: *
R4 = MS4 RS;
CALL UTROO ; /*DELAY FOR TELEMETRY AS REQUIRED*/

```

HAL KERNEL 6 ITERATIVE GUIDANCE MODE

```

S      WRITE (TELX4) R4 ; /*TELEMETER X POSITION IN 4 SYSTEM*/
      1
      CALL UTR00 ; /*DELAY FOR TELEMETRY AS REQUIRED*/
S      WRITE (TELY4) R4 ; /*TELEMETER Y POSITION IN 4 SYSTEM*/
C      2
E      RELEASE INTERRUPTS DISABLED BY TELEMETRY DELAY ROUTINE
      *
      V4 = MS4.VS;
      CALL UTR00 ; /*DELAY FOR TELEMETRY AS REQUIRED*/
S      WRITE (TELZ4) R4 ; /*TELEMETER Z POSITION IN 4 SYSTEM*/
      3
      CALL UTR02 ; /*DELAY FOR TELEMETRY AS REQUIRED*/
S      WRITE (TELYD4)V4 ; /*TELEMETER Y VELOCITY IN 4 SYSTEM*/
C      2
C      RELEASE INTERRUPTS DISABLED BY TELEMETRY DELAY ROUTINE
C      CALCULATE RANGE ANGLE MEASURED IN ORBIT PLANE
IG254:  IF T2I = 0
      THEN DO;
      L12,J12,S12,Q12,P12,U12 = 0;
      GO TO IG259;
      END;
      IF T1I = 0
      THEN DO;
      L1, J1, S1, Q1, P1, U1 = 0;
      GO TO IG258;
      END;
      L1 = VEX1 LOG(TAU1/(TAU1 - T1I));
      J1 = L1 TAU1 - VEX1 T1I;
      S1 = L1 T1I - J1;
E      Q1 = S1 TAU1 - .5 VEX1 T1I2;
E      P1 = J1 TAU1 - .5 VEX1 T1I2;
E      U1 = Q1 TAU1 - VEX1 T1I /6.;
IG258:  L2 = VEX2 LOG(TAU2/(TAU2 - T2I));
      J2 = L2 TAU2 - VEX2 T2I;
      S2 = L2 T2I - J2;
E      Q2 = S2 TAU2 - .5 VEX2 T2I2;
E      P2 = J2 TAU2 - .5 VEX2 T2I2;
E      U2 = Q2 TAU2 - VEX2 T2I /6.;
      L12 = L1 + L2;
      J12 = J1 + J2 + L2 T1I;
      S12 = S1 - J2 + L12(T2I + TCI);
      Q12 = Q1 + Q2 + S2 T1I + J1 T2I;
      P12 = P1 + P2 + T1I (2 J2 + L2 T1I);
      U12 = U1 + U2 + T1I (2 Q2 + S2 T1I) + T2I P1;
IG259:  L3P = VEX3 LOG(TAU3/(TAU3 - T3I));
      LYP = L12 + L3P;
      J3P = L3P TAU3 - VEX3 T3I;

```

T1C = T1I + T2I + TCI;
 TSTAR = T1C + T3I;
 PHII = ATAN(R4₃, R4₁);

S
C
C
C
E

DETERMINE PHASE

E
E
E

IG260: IF PHASE THEN /*CALCULATE TERMINAL CONDITIONS*/
 IG262: DO;

E
E

SINTheta = $\frac{RS \cdot \sqrt{9}}{R \cdot V}$;

E

COSTHETA = SQRT(1 - SINTheta²);

DPHII = (V/R) COSTHETA;

DPHIT = (VT/RT) COS(THETA_T);

PHIIT = .5 TSTAR (DPHII + DPHIT);

PHIT = PHII + PHIIT;

CALL UTRO2; /*DELAY FOR TELEMETRY AS REQUIRED*/

WRITE (TELPHT) PHIT; /*TELEMETER TERMINAL RANGE ANGLE*/

C

RELEASE INTERRUPTS LOCKED BY TELEMETRY DELAY ROUTINE

IF TSTAR <= EPSILON3 THEN GO TO IG269;

E

CALL MIG30; /*CALC TERM RAD, VEL, FLT ANGLE (NOT CODED)*/

C

GT = - KMU/RT ;

CALL UTRO0; /*DELAY FOR TELEMETRY AS REQUIRED*/

WRITE (TELGT) GT; /*TELEMETER TERMINAL GRAVITY VECTOR*/

RELEASE INTERRUPTS LOCKED BY TELEMETRY DELAY ROUTINE

$\vec{GVT} = \text{VECTOR}(GT \cos(\text{THETA}_T), 0, GT \sin(\text{THETA}_T));$

$\vec{RVT} = \text{VECTOR}(RT \cos(\text{THETA}_T), 0, 0);$

IG269:

PHIT = PHIT - THETA_T;

END;

ELSE DO; /*CALCULATE INTERMEDIATE PARAMETERS*/

DELTA2 = V TSTAR - J3P + LYP T3I - ROVEX3((TAU1 - T1I) L1 + (TAU2 - T2I) L2 + (TAU3 - T3I) L3P) (LYP + V - VT);

PHIIT = KT (S12 + DELTA2); /*KT = COS(THETA_T)/RT*/

PHIT = PHII + PHIIT;

CALL UTRO2; /*DELAY FOR TELEMETRY AS REQUIRED*/

WRITE (TELPHT) PHIT; /*TELEMETER TERMINAL RANGE ANGLE*/

C

RELEASE INTERRUPTS LOCKED BY TELEMETRY DELAY ROUTINE

END;

C
C
C
E

ROTATE POSITION, VELOCITY, GRAVITY TO INJECTION SYSTEM

E

IG291: M4V = MATRIX(COS(PHIT), 0, SIN(PHIT),
 0, 1, 0,
 -SIN(PHIT), 0, COS(PHIT));

E

$\vec{RV} = M4V \vec{R4};$

E

$\vec{VV} = M4V \vec{V4};$

HAL KERNEL 6 ITERATIVE GUIDANCE MODE

```

E      * *
E      GV = M4V MS4 GS;
E      GVSTAR = .5(GVT + GV);
F      DELTAVVP = VVT - VV - TSTAR GVSTAR;
C
C      IG314 - CALCULATE TIME-TO-GO (NOT CODED)
C
E      IF REITERATE
E          THEN DO;
E              REITERATE = FALSE;
E              L3P = L3;
E              J3P = J3;
E              LYP = LYP + DELTAL3;
E              GO TO IG260;
E          END;
E      ELSE REITERATE = TRUE;
C
C      IG324 - COMPUTE CORRECTED VELOCITIES TO BE GAINED (NOT CODED)
C
C      IG326 - CALCULATE DESIRED PITCH AND YAW (NOT CODED)
C
E      IF CHI_BAR STEERING THEN GO TO IG350;
E      IF TSTAR >= EPSILON2 THEN GO TO IG360;
E
E      IF S4BURN
E          THEN DO;
E              DVMC5 = DVMC5 OR MSKMC5CBS;
E              DVMLR = 25 KCCT4;
E              DV1MR = .04/KCCT4;
E          END;
E      ELSE DO;
E              DVMC6 = DVMC6 OR MSKMC6CBS;
E              DVMLR = 25 KCCT8;
E              DV1MR = .04/KCCT8;
E          END;
E
E      IG340: CHI_BAR_STEERING = TRUE;
E      IG350: K1, K2, K3, K4 = 0;
E              GO TO IG440;
C
C      IG360: ;/*IG361 - COMPUTE INTERMEDIATE PARAMETERS (NOT CODED)*/
C
C      IG440. CALL UTROO ; /*DELAY FOR TELEMETRY AS REQUIRED*/
C              WRITE (TEL13I) T3I; /*TELEMETER T3I*/
C              RELEASE INTERRUPTS DISABLED BY TELEMETRY DELAY ROUTINE
C
C      IG446 - COMPUTE PITCH AND YAW IN 4-SYSTEM (NOT CODED)

```

HAL KERNEL 6 ITERATIVE GUIDANCE MODE

C
E

E

E

```
IF SMC_FLAG THEN CALL MSM00; /*COMPUTE SMC TERMS (NOT CODED)*/  
CALL MCC00; /*PERFORM CHI COMPUTATIONS (NOT CODED)*/  
  
IF DFILE AND MSKFPSINT2 = 0  
    THEN CALL EGP32(MSKSCC0); /*ENABLE INTERRUPT 2(NOT CODED)*/  
CLOSE MIG00;
```

HAL KERNEL 7 DIGITAL COMMAND SYSTEM

```

MDS00: PROGRAM; /*DIGITAL COMMAND SYSTEM*/
        DECLARE INTEGER,
            DCSDATACOUNT,
            DCSDATCT ARRAY(20) CONSTANT
                (0,1,35,2,2,3,3 0,35,8 0,6,0),
            DCSERLIM CONSTANT (7),
            DCSINDX,
            DCSMODE ARRAY (64) CONSTANT
                (5#0,8,2#0,1,2,3,4,5,2#0,14,6,0,7,2#0,19,3#0,9,0,15,
                17,8#0,13,4#0,18,10,11,12,2#0,16,15#0),
            VDSRC;
        DECLARE BIT(1),
            DCMSTAT ARRAY(20),
            FDSER,
            FDSPG,
            FDSRE,
            VDSSR;
        DECLARE BIT(26),
            DCSE04 CONSTANT (OCT'040000000'),
            DCSE10 CONSTANT (OCT'100000000'),
            DCSE14 CONSTANT (OCT'140000000'),
            DCSE20 CONSTANT (OCT'200000000'),
            DCSE24 CONSTANT (OCT'240000000'),
            DCSE44 CONSTANT (OCT'440000000'),
            DCSE60 CONSTANT (OCT'600000000'),
            DCSE64 CONSTANT (OCT'640000000'),
            DCSE74 CONSTANT (OCT'740000000'),
            DCSSCOD ARRAY(20) CONSTANT
                (OCT'000000000', OCT'100000000', OCT'110000000',
                OCT'120000000', OCT'130000000', OCT'140000000',
                OCT'200000000', OCT'220000000', OCT'050000000',
                OCT'310000000', OCT'770000000', OCT'770000000',
                OCT'770000000', OCT'450000000', OCT'170000000',
                OCT'330000000', OCT'600000000', OCT'340000000',
                OCT'520000000', OCT'250000000'),
            VDSE,
            VDS01;
        DECLARE ARRAY(35) VDSBL BIT(6);

C
C
C
E
        READ(DIR) BTEMP; /*READ DISCRETE INPUT REGISTER*/
F
        READ(DCS) VDS01; /*READ DCS INPUT REGISTER*/
E
        IF BTEMP AND MSKDCSMODE = 0 THEN GO TO DS60;
C
C
C
E
        PROCESS DCS MODE COMMAND
DS09:   IF VDS01          = NOT VDS01
S        1 TO 7          8 TO 14
        THEN DO;
E

```

HAL KERNEL 7 DIGITAL COMMAND SYSTEM

```

VDSER = DC SER10;
GO TO DS220;
END;
E
IF VDS01 AND MSKDCSSB = 0
THEN DO;
E
    VDSER = DC SER24;
    GO TO DS220;
END;
E
IF VDS01 AND MSKDCSMC = MSKDCSTERM THEN GO TO DS25;
E
IF NOT FDS EN
THEN DO;
E
    VDSER = DC SER20;
    GO TO DS220;
END;
E
IF DFDTL OR FDSPG
THEN DO;
E
    VDSER = DC SER64;
    GO TO DS220;
END;
E
DS20: FDS PG = TRUE;
DS25: ITEMP = INTEGER(VDS01
                    1 TO 6
                    );
DCSINDX = DCSMODE
          + 1;
          ITEMP+1
E
IF NOT DCSMSTAT
DCSINDX
THEN DO;
E
    FDSPG = FALSE;
E
    VDSER = DC SER74;
    GO TO DS220;
END;
C
TELEMETER STATUS CODE TWICE
CALL UTR24 ; /*DELAY FOR TELEMETRY AS REQUIRED*/
WRITE (TELDCSSC) DCSSTCOD
;
S
DCSINDX
CALL UTR24 ; /*DELAY FOR TELEMETRY AS REQUIRED*/
WRITE (TELDCSSC) DCSSTCOD
;
S
DCSINDX
C
RELEASE INTERRUPTS DISABLED BY TELEMETRY DELAY ROUTINE
CALL DS200; /*ISSUE CRP*/
DCSDATACOUNT = 0;
E
VDS SB = BIN'0';
GO TO DS100;

```



```

C
C      PROCESS DATA WORD
C
E
DS60:  IF FDSEN
        THEN DO;
        .
        VDSER = DCSER04;
        GO TO DS220;
        END;
E
        IF VDS01      = NOT VDS01
          1 TO 7      8 TO 14
        THEN DO;
        .
        VDSER = DCSER44;
        GO TO DS220;
        END;
E
        IF VDS01      = VDS08
          7
        THEN DO;
        .
        VDSER = DCSER60;
        GO TO DS220;
        END;
DS110: /*TELEMETER DATA WORD TWICE*/
        CALL UTR24 ; /*DELAY FOR TELEMETRY AS REQUIRED*/
E
        WRITE (TELDCSDW) VDS01;
        CALL UTR24 ; /*DELAY FOR TELEMETRY AS REQUIRED*/
E
        WRITE (TELDCSDW) VDS01;
        RELEASE INTERRUPTS DISABLED BY TELEMETRY DELAY ROUTINE
        CALL DS200; /*ISSUE CRP*/
E
        VDSBL          = VDS01      ;
        DCSDATACOUNT + 1      1 TO 6
E
        VDS08 = NOT VDS08;
        DCSDATACOUNT = DCSDATACOUNT + 1;
DS100: IF DCSDATACOUNT < DCSDATCT      THEN RETURN; /*MDS00*/
S
        DO CASE DCSSINDX;
        DO;
E
                FDSPO = FALSE;
                .
                VDSER = DCSER14;
                GO TO DS220;
        END;
        CALL DS260; /*TIME BASE UPDATE      (NOT CODED)*/
        DO;
                CALL DS330 ASSIGN(EXIT);/*NAV UPDATE      (NOT CODED)*/
                IF EXIT THEN GO TO DS235;
    
```

```

END;
DO;
    CALL DS380 ASSIGN(EXIT);/*GENERAL SS (NOT CODED)*/
    IF EXIT THEN GO TO DS220;
END;
CALL DS430; /*SECTOR DUMP (NOT CODED)*/
CALL DS470; /*TELEMETER SINGLE MEMORY LOC(NOT CODED)*/
CALL DS510; /*TERMINATE (NOT CODED)*/
CALL DS540; /*MANEUVER UPDATE (NOT CODED)*/
CALL DS550; /*MANEUVER INHIBIT (NOT CODED)*/
DO;
    CALL DS670 ASSIGN(EXIT);/*TARGET UPDATE(NOT CODED)*/
    IF EXIT THEN GO TO DS235;
END;
CALL DS700; /*ANTENNA TO OMNI (NOT CODED)*/
CALL DS720; /*ANTENNA TO LOW (NOT CODED)*/
CALL DS740; /*ANTENNA TO HIGH (NOT CODED)*/
CALL DS770; /*INHIBIT WATER CONTROL VALVE(NOT CODED)*/
CALL DS790; /*TIME BASE 8 ENABLE (NOT CODED)*/
CALL DS810; /*EXECUTE MANEUVER A (NOT CODED)*/
CALL DSR40; /*TD AND E ENABLE (NOT CODED)*/
CALL DS860; /*EXECUTE MANEUVER B (NOT CODED)*/
CALL DS900; /*S4R/IU LUNAR IMPACT (NOT CODED)*/
CALL DS960; /*ENABLE TB6D ALTERNATE SEQ (NOT CODED)*/
END;
GO TO DS530;

```

```

C
C PROCESS DCS ERROR CONDITION
C
DS220: VDSRC = VDSRC + 1;
      IF VDSRC < DCSERLIM
E
      THEN FDSRE = FALSE;
E
      ELSE FDSRE = TRUE;
E
      VDSER = VDSER OR BIT(VDSRC) OR VDS01 ;
S
      DS235: /*TELEMETER ERROR CODE TWICE*/
      CALL UTR24 ; /*DELAY FOR TELEMETRY AS REQUIRED*/
E
      WRITE (TELCSEC) VDSER;
      CALL UTR24 ; /*DELAY FOR TELEMETRY AS REQUIRED*/
E
      WRITE (TELCSEC) VDSER;
      RELEASE INTERRUPTS DISABLED BY TELEMETRY DELAY ROUTINE
E
      IF NOT FDSRE THEN RETURN; /*MDS00*/
DS530: VDSRC = 0;
E
      FDSER = TRUE;
E
      FDSPG = FALSE;
      RETURN; /*MDS00*/
DS200: PROCEDURE; /*ISSUE DCS COMMAND RESET PULSE*/

```

HAL KERNEL 7 DIGITAL COMMAND SYSTEM

C
C
C
E

INHIBIT ALL INTERRUPTS EXCEPT TLC

WRITE(DOS) MSKDCSDO; /*SET COMMAND RESET BIT IN DOR*/
WAIT /* 4.13 MILLI-SEC*/;

E

WRITE(DOR) MSKDCSDO; /*RESET COMMAND RESET BIT IN DOR*/

C
C
C

RELEASE PREVIOUSLY ENABLED INTERRUPTS

CLOSE DS200;
CLOSE MDS00;

HAL KERNEL 8 ACCELEROMETER PROCESSING

```

MAR00: PROGRAM; /*ACCELEROMETER READ ROUTINE*/
        DECLARE VOAC ARRAY(3) BIT(26);
        DECLARE SCALAR,
                VCCYA ,
                VCCZA ;

C      INHIBIT ALL INTERRUPTS EXCEPT TLC
C
C      READ(XACC) DVAC ;
S      1
        READ(YACC) DVAC ;
S      2
        READ(ZACC) DVAC ;
S      3
        READ(CLOCK) DVACT;
        CALL UTR00 ; /*DELAY FOR TELEMETRY AS REQUIRED*/
        WRITE (TELTI) DVTI; /*TELEMETER TIME OF CURRENT TIME BASE*/
        STEMP = DVTAS;
        ITEM = DVACT - DVRTC - DVERT;
        IF ITEM < 0 THEN ITEM = ITEM + DKRTC0VF;
        DVTAS = DVTMM DKRTCSEC + (DKRTCSEC/4) ITEM;
        DVTB = DVTAS - DVTI;
        DVDT = DVTAS - STEMP;
        CALL UTR00 ; /*DELAY FOR TELEMETRY AS REQUIRED*/
        WRITE (TELTB) DVTB; /*TELEMETER TIME IN CURRENT TIME BASE*/

F      DVMC4 = DVMC4 AND MSKRTCRESET;

C      RELEASE PREVIOUSLY ENABLED INTERRUPTS
C
C      CALL UTR00 ; /*DELAY FOR TELEMETRY AS REQUIRED*/
        WRITE (TELXAC) DVAC ; /*TELEMETER X ACCELEROMETER READING*/
S      1
        CALL UTR00 ; /*DELAY FOR TELEMETRY AS REQUIRED*/
        WRITE (TELYAC) DVAC ; /*TELEMETER Y ACCELEROMETER READING*/
S      2
C      RELEASE INTERRUPTS DISABLED BY TELEMETRY DELAY ROUTINE
        IF DKT1 = 0. THEN DVFMC = - DVG ;
S      1
                ELSE DO;
                        DVMAS = DVMAS - DVEOF DVMFR DVDT;
                        DVFMC = DVEOF DVFOR/DVMAS;
                END;

C      COMPUTE AVERAGE CHI'S FOR SMC CALCULATIONS
C
C      AR41: DVCA = (DVCC + VCCZA)/2.;
S      3
                VCCZA = DVCC ;
S      3
        DVCA = (DVCC + VCCYA)/2.;
S      2
                VCCYA = DVCC ;
S      2
        CALL UTR00 ; /*DELAY FOR TELEMETRY AS REQUIRED*/

```

HAL KERNEL 8 ACCELEROMETER PROCESSING

```

WRITE (TELZAC) DVAC ; /*TELEMETER Z ACCELEROMETER READING*/
3
RELEASE INTERRUPTS DISABLED BY TELEMETRY DELAY ROUTINE
COMPUTE CHANGES BETWEEN CURRENT AND PREVIOUS ACCELEROMETER READINGS
S
C
C
C
E
AR100:  DVDA = VECTOR( (DVAC      ) - VECTOR( (VOAC      ) );
          1 TO 12          1 TO 12
S
E
          DVDB = VECTOR( (DVAC      ) - VECTOR( (VOAC      ) );
          15 TO 26        15 TO 26
S
E
          (VOAC) = (DVAC);
          CALL UTR00 ; /*DELAY FOR TELEMETRY AS REQUIRED*/
          WRITE (TELRTC) DVACT; /*TELEMETER REAL TIME CLOCK AT ACCEL READ*/
          RELEASE INTERRUPTS DISABLED BY TELEMETRY DELAY ROUTINE
C
C
C
C
COMPUTE EXPECTED VELOCITY CHANGES
AR71:   DVD  = 20 DVDT COS(DVTH ) COS(DVTH );
S        1          2          3
S        DVD  = 20 DVDT SIN(DVTH );
S        2          3
S        DVD  = - 20 DVDT SIN(DVTH ) COS(DVTH );
S        3          2          3
          CALL UTR00 ; /*DELAY FOR TELEMETRY AS REQUIRED*/
          WRITE (TELTA) DVTA; /*TELEMETER MISSION ELAPSED TIME*/
          RELEASE INTERRUPTS DISABLED BY TELEMETRY DELAY ROUTINE
C
E
          DVF = DVFOM DVDB;
          CLOSE MAR00;
MAP00:  PROGRAM; /*ACCELEROMETER PROCESSING*/
          DECLARE SCALAR,
          DELTA,
          KSN2D CONSTANT (.0348994967), /*SIN 2 DEGREES*/
          VACZR,
          VPOV ARRAY(3);
          DECLARE ARRAY(3) BIT(26) CONSTANT,
          MSKAPDG (OCT'040000000',OCT'010000000',OCT'200000000'),
          MSKAPDF (OCT'000000010',OCT'000000200',OCT'000000020');
          DVVSQ = 0;
          VACZR = 20 DVFOM DVDT KSN2D;
          DO FOR I = 1 TO 3;
AP400:  IF ABS(DVDA - DVDB ) <= 2 THEN GO TO AP450;
S        I          I
          IF ABS(DVDA - DVF ) < ABS(DVDB - DVF ) THEN GO TO AP440;
S        I          I          I          I
F
          DVMC4 = DVMC4 OR MSKAPDG          CAT BIN'0';
S        I:2 TO 26
          DELTA = DVDB ;
S        I
          GO TO AP460;
E
          .          .          .

```

HAL KERNEL 8 ACCELEROMETER PROCESSING

```

AP440:      DVMC4 = DVMC4 OR MSKAPDG ;
S
AP450:      DELTA = DVDA ;
S
E
AP460:      IF ABS(DELTA) > 1 OR NOT DFZER OR ABS(DVF ) < VACZR
S
              THEN GO TO AP500;
E
AP470:      DVMC4 = DVMC4 OR MSKAPDF ;
S
E
AP530:      DVMC4 = DVMC4 OR MSKAPDG OR MSKAPDG          CAT BIN'0';
S
              I:2 TO 26
E
          DFMC = FALSE;
          DELTA = DVFC DVD ;
S
          GO TO AP520;
AP500:      IF DVF < 0
S
              I
              THEN DO:
S
                  IF DELTA < 1.5 DVF - DVDT DVRC OR
S
                      DELTA > .5 DVF + DVDT DVRC
S
                          THEN GO TO AP530;
S
                  END;
S
                  ELSE IF DELTA > 1.5 DVF + DVDT DVRC OR
S
                      DELTA < .5 DVF - DVDT DVRC
S
                          THEN GO TO AP530;
E
          DVVSQ = DVVSQ + DELTA ;
          VPOV = VPOV + DELTA;
S
          DVDM = .05 VPOV ;
S
          CALL UTR01; /*DELAY FOR TELEMETRY AS REQUIRED*/
          DO CASE I:
S
              WRITE (TELXDM) DVDM ;/*TELEMETER X MEASURED VELOCITY*/
S
              WRITE (TELYDM) DVDM ;/*TELEMETER Y MEASURED VELOCITY*/
S
              WRITE (TELZDM) DVDM ;/*TELEMETER Z MEASURED VELOCITY*/
S
          END;
C
          RELEASE INTERRUPTS LOCKED BY TELEMETRY DELAY ROUTINE
END;
CLOSE MAP00;

```

HAL KERNEL 9 MINOR LOOP

```

MML00: PROGRAM; /*FLIGHT SIMULATION MINOR LOOP*/
IF DVLRC = 0
E
    THEN DVCC = DVCC - DVDC;
    ELSE DVLRC = DVLRC - 1;
    CALL MML20; /*EXECUTE NORMAL MINOR LOOP*/
    CLOSE MML00;
MML20: PROGRAM; /*MINOR LOOP*/
    DECLARE BIT(26),
        VGR ARRAY(3),
        VMEMR,
        VMLET;
    DECLARE SCALAR,
        KCPBG CONSTANT (641.5839),
        VOCK;
    DECLARE ARRAY (3) SCALAR,
        VBUB,
        VCG ,
        VDEL,
        VML2,
        VSF ;
    DECLARE ARRAY(3) INTEGER,
        FBUG ,
        VCG0 ,
        VCG1 ,
        VCMND (3,3),
        VCOD ,
        VFIO ,
        VMLO ,
        VML1 ,
        VOLD ,
        VPGR ;
    DECLARE INTEGER,
        FBUGS,
        KMAXLAD CONSTANT (256),
        VCG10,
        VCG11,
        VIRE ;
E
    DVCC = DVCC + DVDC;
    IF FBUGS /= 0 THEN GO TO ML500;
ML001: DO FOR I = 3 TO 1 BY -1;
        DO CASE I;
ML201: DO;
E
        DVEMR = DVEMR OR VMEMR;
        DO CASE VFIO + 1;
S
E
            1
                VGR = VPGR ;/*USE INTERNAL X GIMBAL VALUE*/
S
E
            1 1
                READ(XGIM) VGR ;/*READ X GIMBAL*/
S
E
            1

```

HAL KERNEL 9 MINOR LOOP

```

S          READ(XBGIM) VGR ; /*READ X BACKUP GIMBAL*/
S          1
          END;
          END ML201;
ML101;     DO;
E          READ(EMR) VMEMR;
E          DVLDB = DVLDB - (VMEMR AND MSKEMRLADB);
S          DG CASE VFIO + 1;
E          2
          VGR = VPGR ; /*USE INTERNAL Y GIMBAL VALUE*/
S          2      2
E          READ(YGIM) VGR ; /*READ Y GIMBAL*/
S          2
E          READ(YBGIM) VGR ; /*READ Y VACKUP GIMBAL*/
S          2
          END;
          END ML101;
ML001A;    DO CASE VFIO + 1;
S          3
E          VGR = VPGR ; /*USE INTERNAL Z GIMBAL VALUE*/
S          3      3
E          READ(ZGIM) VGR ; /*READ Z GIMBAL*/
S          3
E          READ(ZBGIM) VGR ; /*READ Z BACKUP GIMBAL*/
S          3
          END ML001A;
          END;
E          IF VGR = 0 THEN GO TO ML020;
S          I:1
ML430;     IF DVDGS < 0 THEN GO TO ML432;
          IF DVDGS = 0 THEN GO TO ML020;
          GO TO ML637;
ML432;     CALL MDGUO ASSIGN(J); /*PROCESS DISAGREEMENT BIT(NOT CODED)*/
C          DISAGREEMENT BIT PROCESSING WILL RETURN ONE OF THE FOLLOWING FOR J:
C          J = 0 FOR INVALID DISAGREEMENT BIT
C          J = 1 FOR VALID GIMBAL, VALID DISAGREEMENT BIT
C          J = 2 INVALID GIMBAL, VALID DISAGREEMENT BIT
C
          IF J = 0 THEN GO TO ML020;
ML434;     DO CASE I;
ML4352;    DO CASE VFIO ;
S          1
E          READ(XGIM) BTEMP; /*RESTART Z COD COUNTER*/

```


HAL KERNEL 9 MINOR LOOP

```

                                READ(XBGIM) BTEMP; /*RESTART Z COD COUNTER*/
                                END ML4352;
ML4351: DO CASE VFIO ;
S                                     2
E                                     .
                                READ(YGIM) BTEMP; /*RESTART X COD COUNTER*/
E                                     .
                                READ(YBGIM) BTEMP; /*RESTART X COD COUNTER*/
                                END ML4351;
ML4350: DO CASE VFIO ;
S                                     3
E                                     .
                                READ(ZGIM) BTEMP; /*RESTART Y COD COUNTER*/
E                                     .
                                READ(ZBGIM) BTEMP; /*RESTART Y COD COUNTER*/
                                END ML4350;
                                END ML434;
                                IF J = 2 THEN GO TO ML637;
E
ML020: VCOD = VGR ;
S      I      I:2 TO 12
S      IF VCOD = 0 AND VOLD = 0 AND ABS(VDEL ) >= VOCK
S      I      I      I
      THEN GO TO ML631;
S      IF ABS(VCOD - VOLD ) < VML0 THEN GO TO ML040;
S      I      I      I
S      IF ABS(VCOD - VOLD ) + VML0 < VML1 THEN GO TO ML630;
S      I      I      I      I
S      IF VCOD < VOLD
S      I      I
      THEN VCG = VCG + VML2 ;
S      I      I      I
      ELSE VCG = VCG - VML2 ;
S      I      I      I
ML040: DVTH = VSF VCOD + VCG ;
S      I      I      I      I
S      VOLD = VCOD ;
S      I      I
S      VDEL = DVTH - DVCC ;
S      I      I      I
E      .
      DFDBF = TRUE;
      DO CASE I;
S      VCMND = DVA6 (VDEL + DVA3 VDEL );
S      1,1      1      2
S      VCMND = DVA1 VDEL + DVA2 VDEL );
S      2,1      2      3
S      VCMND = DVA5 VDEL - DVA4 VDEL );
S      3,1      3      2
      END;
      GO TO ML730;
E
ML630: VMLLET = I + 2;
E      GO TO ML632;
      .

```

HAL KERNEL 9 MINOR LOOP

```

ML631:      VMLET = I - 1;
E
ML632:      VMLET = VMLET          CAT BIN(11)'0' OR BIT(VCOD )
S              12 TO 26          I
              OR BIT          (VOLD ) CAT BIN(15)'0';
S              16 TO 26          I
E
          IF DVMC6 AND MSKMC6D04 = 0
          THEN DO;
          CALL UTR30; /*DELAY FOR TELEMETRY AS REQUIRED*/
E
          WRITE (TEMLER) VMLET; /*TELEMETEK ERROR MESSAGE*/
          END;
E
          IF NOT DFDHF THEN GO TO ML635;
          DVRE = DVRE + 1;
S              I
          IF DVRE < 0 THEN GO TO ML637;
S              I
          IF DVRE > 0 THEN GO TO ML636;
S              I
E
          VMLET = BIT(VCOD ) OR BIT          (VOLD ) CAT OCT'34000';
S              I          16 TO 26          I
          VFIO = 2; /*SET I/O FLAG FOR BACKUP GIMBAL*/
S              I
          IF VCG >= PI THEN VCG = PI - VBUB ;
S              I          I          I
              ELSE VCG = - VBUB ;
S              I          I          I
          VML2 = PI;
S              I
          IF DVTH >= PI THEN VOLD = (DVTH - PI) KCPBG;
S              I          I          I
              ELSE VOLD = DVTH KCPBG;
S              I          I
          VSF = 1/KCPBG;
S              I
          IF I = 3 THEN DO;
E
          WRITE(ICR) MSKICRBG; /*SET ICR TO SELECT BACKUP*/
E
          DVICR = DVICR OR MSKICRBG;
          END;
          FBUGS, FBUG = 2;
S              I
          VML0 = VCG10;
S              I
          VML1 = VCG11;
S              I
E
          IF DVMC6 AND MSKMC6D04 = 0
          THEN DO;
          CALL UTR30; /*DELAY FOR TELEMETRY AS REQUIRED*/
E

```

HAL KERNEL 9 MINOR LOOP

```

                                WRITE (TELMLER) VMLET; /*TELEMETER ERROR MESSAGE*/
                                END;
                                GO TO ML637;
ML635:  DVHDB = DVHDB - 1;
E
                                DFDBF = TRUE;
                                DVHDA = DVHDA + 1;
                                IF DVHDA < 0 THEN GO TO ML636;
E
                                WRITE(ICR) M$KICRSWG; /*SET ICR TO SWITCH GIMBAL ORDER*/
E
                                DVICR = DVICR OR M$KICRSWG;
E
                                DVMC4 = DVMC4 OR M$KMC4AMF;
                                DVDGS = 0;
E
ML636:  IF DVRE >= VIRE AND (DVMC6 AND M$KMC6D04) = 0
S
F
                                THEN CALL U0000(M$KGRF); /*SET GUIDANCE FAILURE DISC.*/
E
ML637:  DF$MC = FALSE;
                                GO TO ML760;
ML730:  IF ABS(VCMND I,1 ) > DVM06 THEN VCMND I,1 = DVM06;
S
                                IF ABS(VCMND I,1 - VCMND I,2 ) > DVM05
S
                                THEN VCMND I,1 = VCMND I,2 + DVM05;
S
                                VCMND I,1 = VCMND I,2 ;
S
                                IF VCMND I,2 < 0
S
                                THEN VCMND I,1 = KMAXLAD - VCMND I,1 ;
S
                                ELSE VCMND I,3 = VCMND I,1 ;
S
                                DO CASE I;
ML760:  DO;
ML260:  READ(DBG) ITEMP; /*START SPECIAL DOM BACKUP GIMBAL*/
                                WRITE(ZLAD) VCMND I,3 ; /*ISSUE YAW COMMAND*/
S
                                WRITE(XLAD) VCMND I,3 ; /*ISSUE ROLL COMMAND*/
S
                                END ML260;
ML160:  WRITE (YLAD) VCMND I,3 ; /*ISSUE PITCH COMMAND*/
S
ML060:  DO;
                                WRITE(ZLAD) VCMND I,3 ; /*ISSUE YAW COMMAND*/
S
E
                                IF DVLDR < 0 THEN WRITE(ICR) M$KICRCA;
                                ITEMP = DVTT1 - DVRTC;
                                IF ITEMP < 0 THEN ITEMP = ITEMP + DKRTC0VF;
                                DVMLT = DVTMM + DVMLD + ITEMP/4;

```

HAL KERNEL 9 MINOR LOOP

```

                END ML060;
            END ML760;
        END ML001;
        RETURN; /*MML20*/
ML500: DO FOR I = 1 TO 3:
            DO CASE FBUG + 1;
S                I
                GO TO ML530;
                DO;
S                    FBUG = 0;
                    I
S                    VML0 = VCG0 ;
                    I           I
S                    VML1 = VCG1 ;
                    I           I
S                END;
                FBUG = 1;
S                I
            END;
ML530: END;
        FRUGS = SUM([FBUG]);
        GO TO ML001;
        CLOSE MML20;

```

MS00: PROGRAM; /*SWITCH SELECTOR PROCESSING*/

C
C
C
C
C
C
C
C
C
C
C
C
C

SWITCH SELECTOR TABLE

THE SWITCH SELECTOR TABLE IS MADE UP OF A NUMBER OF SMALLER TABLES, ONE FOR EACH TIME BASE AND FOR EACH OF THE ALTERNATE SS SEQUENCES. THE SMALLER TABLES ARE ORGANIZED INTO ONE LARGE TABLE. HOWEVER, ONLY THE TIME BASE 1 TABLE HAS BEEN CODED.

EACH TABLE ENTRY REPRESENTS A SINGLE SS COMMAND AND CONSISTS OF TWO WORDS.

1. TIME OF SS ISSUANCE (IN TENTHS OF A SECOND).
2. SS STAGE AND ADDRESS.

DECLARE SSTABLE ARRAY(2,1000) INTEGER CONSTANT

(50 , OCT'000000000',
 60 , OCT'106500000',
 140 , OCT'026100000',
 198 , OCT'406440000',
 200 , OCT'405440000',
 202 , OCT'406340000',
 240 , OCT'025740000',
 270 , OCT'402100000',
 290 , OCT'027740000',
 300 , OCT'403040000',
 320 , OCT'401100000',
 495 , OCT'020100000',
 750 , OCT'000000000',
 900 , OCT'402100000',
 950 , OCT'401100000',
 953 , OCT'022640000',
 1050 , OCT'407640000',
 1151 , OCT'025740000',
 1198 , OCT'406740000',
 1200 , OCT'405740000',
 1201 , OCT'027740000',
 1300 , OCT'404040000',
 1324 , OCT'021640000',
 1336 , OCT'400700000',
 1338 , OCT'401700000',
 1344 , OCT'021740000',
 1346 , OCT'023740000',
 0'377777776', OCT'000000000');

DECLARE BIT(1),

FASE ,
 FBRNI ,
 FCLS4 ,
 FFBCH ,
 FHST ,
 FSSAC ,
 FSSIO ,
 FTADV ,
 FT60P ;

DECLARE INTEGER CONSTANT,

```

KCSSK      (203) ,
KSSB1     ( 18) ,
KSSB2     ( 26) ,
KSSB3     ( 17) ,
KSSB4     (  9) ,
KSSB5     ( 26) ,
KSSB6     ( 13) ,
KSSB7     ( 22) ,
KSSB8     ( 11) ,
KSSPB     ( 50) ,
KSS500MS  (508) ,
KSS500SEC (507937),
KSSINDEXALU    (410) ,
KSSINDEXESV   (386) ,
KSSINDEXES1   (384) ,
KSSINDEXGAIN  (405) ,
KSSINDEXGSS   (378) ,
KSSINDEXSHHI  (388) ,
KSSINDEXSHLO  (390) ,
KSSINDEXSBOM  (392) ,
KSSINDEXSIVA  (380) ,
KSSINDEXSIVH  (382) ,
KSSINDEXS4C1  (304) ,
KSSINDEXTB3A  (301) ,
KSSINDEXTB5A  (394) ,
KSSINDEXTB5B  (399) ,
KSSINDEXTB6A  (343) ,
KSSINDEXTB6B  (348) ,
KSSINDEXTB6C  (352) ,
KSSINDEXTB6D  (373) ;
DECLARE INTEGER,
SST1PTR ARRAY (8) CONSTANT (0, 28, 41, 84, 113, 151,
230, 281),
SST1PTR,
SST2PTR,
VATRR ,
VATR4 ,
VGBIA ,
VSC10 ,
VSC12 ,
VSC30 ,
VSC32 ,
VSSRT ,
VSSTM ,
VSSW ,
VSTGO ;
DECLARE BIT(26),
VASPI ,
VHSTW ,
VPSTG ,
VSCCA ,
VSC11 ,
VSC31 ,
VSNA ,
VSNA1 ,

```

```

VSSCA ,
VSSF8 ,
VSTG ;
DO CASE DGSSM:
GO TO MSS000;
GO TO MSS05;
GO TO MSS10;
GO TO MSS20;
GO TO MSS30;
GO TO MSS40;
GO TO MSS50;
GO TO MSS55;
GO TO MSS60;
GO TO MSS70;
GO TO MSS80;
END;
MSS000: ;
C
C INHIBIT ALL INTERRUPTS EXCEPT TLC
C
E .
E FASE = FALSE;
E IF .DVASW AND .MSKSSS4CO ^= 0
E THEN DO;
E .
E .DVASW = .DVASW AND .MSKSSWV;
E
E IF .VASPI AND .MSKSSS4CO ^= 0 THEN GO TO SS0060;
E CALL EGP08; /*RESCHEDULE TIMER 1 (NOT CODED)*/
E
E .
E .VASPI = .MSKSSS4CO;
E .SST1PTR = .KSSINDXSIVR;
E GO TO SS1050;
E
E END;
E IF .DVASW AND .MSKSSSPEC ^= 0
E THEN DO;
E .
E .DVASW = .DVASW AND .MSKSSWV;
E CALL EGP08; /*RESCHEDULE TIMER 1 (NOT CODED)*/
E
E .
E .VASPI = .MSKSSSPEC;
E .SST1PTR = .KSSINDXSIVA;
E GO TO SS1050;
E
E END;
F IF .DVASW AND .MSKSSTB6C ^= 0
E THEN DO;
E .
E .DVASW = .DVASW AND NOT .MSKSSTB6C;
E
E .
E .VASPI = .VASPI OR .MSKSST6C;
E
E .
E .DVMC6 = .DVMC6 OR .MSKMC6TB6C;

```

HAL KERNEL 10 SWITCH SELECTOR PROCESSING

```

SST1PTR = KSSINDXB6C;
CALL SSTUPD ASSIGN (VATRR); /*UPDATE SS TIME*/
GO TO SS1050;
END;
F
IF DVASW AND MSKSSCL91 = 0
  THEN DO;
E
  VSC10 = SST1PTR;
E
  VSC11 = VASPI;
  VSC12 = VATRR ;
E
  VASPI = MSKSSCL1;
  CALL SSTUPD ASSIGN (VATRR); /*UPDATE SS TIME*/
F
  FTADV = TRUE;
E
  IF DVASW AND MSKSSTB6A = 0
    THEN DO;
F
      DVASW = DVASW AND NOT MSKSSTB6A;
F
      DVMC6 = DVMC6 OR MSKMC6TB6A;
      SST1PTR = KSSINDXB6A;
E
    END;
F
  ELSE IF DVASW AND MSKSS4C1 = 0
    THEN DO;
F
      DVASW = DVASW AND NOT MSKSS4C1;
      SST1PTR = KSSINDXS4C1;
E
    END;
  ELSE DO;
F
      DVASW = DVASW AND NOT MSKSSTB6B;
E
      DVMC6 = DVMC6 OR MSKMC6TB6B;
      SST1PTR = KSSINDXB6R;
E
    END;
  GO TO SS1050;
END;
F
IF FSSAC THEN GO TO SS0060;
ELSE GO TO SS0000;
E
MSS05: FSSAC = FALSE;
E
SS0000: IF SSTABLE = MSKSSNSSEND THEN
S
SS0010: DO;
  CALL SSTUPD ASSIGN (VSTGO); /*UPDATE SS TIME*/
  VSTGO = VSSRT - VSTGO;
  IF VSTGO < KSS500MS THEN GO TO MSS30;
F
  IF DFTUP

```


HAL KERNEL 10 SWITCH SELECTOR PROCESSING

```

        THEN DO;
            DVTGB = DVTGB + VGR1A;
            VGB1A = 0;
E
            DFTUP = FALSE;
            GO TO SS0010;
        END;
E
        IF DVASW = 0 THEN GO TO SS0170;
        VSSTM = VSTGO + DVTRB - DVTGB - KCSSK;
        DVSST = VSSTM + DVTMR;
        DGSSM = 5; /*SET SS ENTRY INDEX FOR MSS30*/
        IF NOT DFIL3 THEN CALL EGP08; /*RESCHED T1(NOT CODED)*/
        GO TO SS0060;
        END;
E
SS0015: IF DVASW = 0 THEN GO TO SS0170;
        READ(CLOCK) ITEMP;
        IF ITEMP - DVRTC < 0
            THEN ITEMP = ITEMP - DVRTC + DKRTC0VF;
            ELSE ITEMP = ITEMP - DVRTC;
        DVSST = DVTMM + KSS500SEC + ITEMP/4;
        DGSSM = 2; /*SET SS ENTRY INDEX FOR MSS05*/
        IF NOT DFIL3 THEN CALL EGP08; /*RESCHED T1 (NOT CODED)*/
        GO TO SS0060;
E
SS0170: IF DVASW AND MSKSSCLS3 = 0
        THEN DO;
E
            IF DVASW AND MSKSSACQU = 0
            THEN DO;
E
                DVASW = DVASW AND NOT MSKSSACQU;
                SST2PTR = KSSINDXGAIN;
                CALL SSTUPD ASSIGN (VATR4); /*UPDATE SS T*/
            END;
            ELSE DO;
E
                IF DVASW AND MSKSSTB6D = 0
                THEN DO;
E
                    DVASW = DVASW AND NOT MSKSSTB6D;
                    SST2PTR = KSSINDXTB6D;
                    CALL SSTUPD ASSIGN(VATR4);
                    /*UPDATE SS TIME*/
                END;
                ELSE DO;
E
                    DVASW = DVASW AND NOT MSKSSLI;
                    SST2PTR = KSSINDXALU;
                    VATR4 = 0;
                END;
            END;
E
        FCLS4 = TRUE;

```

HAL KERNEL 10 SWITCH SELECTOR PROCESSING

```

E          .
E          FTADV = FALSE;
          .
          FHST = TRUE;
          CALL SS210; /*SET UP CLASS 4 ALTERNATE SEQUENCE*/
          GO TO SS0060;
E      END;
          .
E      IF VASPI = 0 THEN GO TO SS0060;
          VSC30 = SST1PTR;
E      .
          VSC31 = VASPI;
          VSC32 = VATRR;
E      .
          VASPI = MSKSSCL3;
          CALL SSTUPD ASSIGN (VATRR); /*UPDATE SS TIME*/
E      .
          FTADV = TRUE;
E      .
          FHST = TRUE;
E      IF DVASW AND MSKSSGNSS = 0
          THEN DO;
E      .
          .
          DVASW = DVASW AND NOT MSKSSGNSS;
          SST1PTR = KSSINDEXSS;
          GO TO SS0230;
E      END;
          .
E      IF DVASW AND MSKSSSBLO = 0
          THEN DO;
E      .
          .
          DVASW = DVASW AND NOT MSKSSSBLO;
          SST1PTR = KSSINDEXSBLO;
          GO TO SS0230;
E      END;
          .
E      IF DVASW AND MSKSSSBHI = 0
          THEN DO;
E      .
          .
          DVASW = DVASW AND NOT MSKSSSBHI;
          SST1PTR = KSSINDEXSBHI;
          GO TO SS0230;
E      END;
          .
E      IF DVASW AND MSKSSSBOM = 0
          THEN DO;
E      .
          .
          DVASW = DVASW AND NOT MSKSSSBOM;
          SST1PTR = KSSINDEXSBOM;
          GO TO SS0230;
E      END;
          .
E      IF DVASW AND MSKSSECSV = 0
          THEN DO;

```

HAL KERNEL 10 SWITCH SELECTOR PROCESSING

```

SST1PTR = KSSINDEXCSV;
GO TO SS0230;
END;
E
IF DVASW AND MSKSSECS1 = 0 THEN SST1PTR = KSSINDEXCS1;
ELSE DO;
SST1PTR = KSSINDXTB3A;
E
DVASW = DVASW AND NOT MSKSST3A;
END;
SS0230: CALL SS210; /*SET UP SS TABLE*/
GO TO SS0000;
E
SS0060: IF NOT FASE
THEN DO;
E
FASE = TRUE;
C
RELEASE PREVIOUSLY ENABLED INTERRUPTS
C
END;
RETURN; /*COMMON SS EXIT*/
E
MSS10: VASPI = BIN(26)'0';
VATRR = 0;
E
FCLS4 = FALSE;
E
DVASW = DVASW AND MSKSSWV;
CALL EGP08; /*RESCHEDULE TIMER 1 (NOT CODED)*/
E
FTADV = TRUE;
SST1PTR = SST1BPTR ;
S
DTBID
SS1050: CALL SS210; /*SET UP NEXT SS*/
E
IF FSSAC THEN GO TO MSS20;
VSSW = KSSB1;
F
FHST = TRUE;
GO TO SS0000;
E
MSS20: IF FSSIO THEN WRITE(SS) MSKSSRESET;
E
FHST = FALSE;
CALL SSTUPQ(KSSR8,2); /*SCHEDULE SS CHECK, MSS05*/
VSSW = KSSB5;
GO TO SS0060;
E
MSS30: FSSAC = TRUE;
E
VSNA , VSNA1 = (VSNA AND MSKSSNA) ;
S
1 TO 26 1 TO 26 1 TO 24
E
IF VSNA = 0

```

HAL KERNEL 10 SWITCH SELECTOR PROCESSING

```

E      THEN DO;
      FSSAC = FALSE;
      CALL SS201; /*ADVANCE TO NEXT SS*/
      GO TO SS0000;
      END;
E      VSTG = VSNA AND VPSTG;
E      FSSIO = VSTG  $\neq$  0;
      IF NOT FHST THEN GO TO SS4000;
      IF DFLT = 2 THEN GO TO SS4000;
E      READ (SSFB) BTEMP; /*READ SS FEEDBACK REGISTER*/
E      IF BTEMP AND MSKSSHS = 0 THEN GO TO SS4000;
      IF FSSIO THEN WRITE(SS) MSKSSRESET; /* ISSUE SS RESET*/
      CALL SSTUPQ(KSSB4,6);/*SCHEDULE STAGE/ADDRESS ISSUANCE MSS40*/
      VSSW = KSSB5;
      GO TO SS0060;
MSS40: WAIT ; /*DELAY BEFORE ISSUING STAGE AND ADDRESS*/
E      IF FSSIO THEN WRITE(SS) VSNA; /*ISSUE STAGE AND ADDRESS*/
      CALL SSTUPQ(VSSW,7);/*SCHEDULE ADDRESS VERIFICATION, MSS50 */
      WAIT ; /*DELAY FOR DOM TELEMETRY*/
      WRITE(DOM); /*OUTPUT SS AND DU REGISTERS VIA DOM TELEMETRY*/
      GO TO SS0060;
E      MSS50: VSCCA = VSNA CAT NOT VSNA CAT BIN(11)'0';
S      1 TO 7 8 TO 15
E      VSSCA = VSCCA AND MSKSSHS;
E      IF VSTG  $\neq$  0
E      THEN READ(SSFB) BTEMP; /*READ SS FEEDBACK REGISTER*/
E      ELSE BTEMP = VSCCA;
E      VSSFB = BTEMP AND MSKSSHS;
E      IF VSSFB  $\neq$  VSSCA THEN GO TO SS5540;
E      MSS55: IF VASPI AND MSKSS4CO  $\neq$  0
      THEN DO;
E      DFILE = DFILE OR MSKFPSISSA;
      DVSST = 1.E10;
      RETURN; /*MSS50, MSS55*/
      END;
      IF VSSRT = 0 THEN GO TO MSS60;
      CALL SSTUPD ASSIGN(DVTRB); /*UPDATE SS TIME*/
      IF VSSRT - DVTRB  $\leq$  KSSRB THEN GO TO MSS60;

```

HAL KERNEL 10 SWITCH SELECTOR PROCESSING

```

VSSTM = VSSRT - DVTGB - KSSRR;
DVSST = VSSTM + DVTMR;
DGSSM = 9; /*SET SS ENTRY INDEX FOR MSS60*/
IF NOT DFIL3 THEN CALL EPOB; /*RESCHEDULE T1 (NOT CODED)*/
RETURN; /*MSS50, MSS55*/
E
SS5540: IF VSSF3 = 0 AND VSSCA = MSKSSZFSF THEN GO TO MSS55;
E
IF FSSIO THEN WRITE(SS) MSKSSRESET; /*ISSUE RESET*/
CALL SSTUPQ(KSSB6,11); /*SCHEDULE COMP STAGE/ADDRESS, MSS80*/
ITEMP = 0;
DO FOR I = 8 TO 15;
E
    IF VSSF3 = VSSCA THEN ITEMP = ITEMP + 1;
S
    I
END;
IF ITEMP < 2 THEN RETURN; /*MSS50*/
E
DVMC4 = DVMC4 OR MSKMC4SSCB;
E
IF FFBCH
    THEN DO:
F
    FFBCH = FALSE;
E
    WRITE(ICR) MSKICRSSCB; /*SWITCH SS TO CHANNEL B*/
E
    DVICR = DVICR OR MSKICRSSCB;
    END;
CALL UTR30 ; /*DELAY FOR TELEMETRY AS REQUIRED*/
E
WRITE (TELSSF3) VSSF3; /*TELEMETER SS FEEDBACK*/
RETURN; /*MSS50*/
F
MSS60: BTEMP = VSTG OR MSKSSREAD;
E
IF FSSIO THEN WRITE(SS) BTEMP; /*ISSUE READ COMMAND*/
READ(CLOCK) ITEMP; /*GET TIME FOR SS TELEMETRY WORD*/
CALL SSTUPQ(KSSB2,10); /*SCHEDULE READ RESET, MSS70*/
F
BTEMP = VSNA CAT BIN'00' OR BIT(ITEMP) AND MSKRTC;
S
    3 TO 26
CALL UTR30 ; /*DELAY FOR TELEMETRY AS REQUIRED*/
E
WRITE (TELSSA) BTEMP; /*TELEMETER STAGE/ADDRESS AND READ TIME*/
E
IF NOT DFACQ
    THEN DO; /*COMPRESS DATA WHEN NOT OVER A STATION*/
E
    BTEMP = DVDCT OR MSKSSDCT;
F
    CALL MPC80(BTEMP); /*COMPRESS TIME AND TAG*/
E
    BTEMP = VSNA OR MSKSSDCS;
S
    1 TO 23

```

HAL KERNEL 10 SWITCH SELECTOR PROCESSING

```

E          CALL MPC80(BTEMP); /*COMPRESS STAGE AND ADDRESS*/
E          END;
E          IF VASPI AND MSKSSS4CO = 0
E              THEN DO;
E              VASPI = BIN(26)'0';
E              DFILE = DFILE OR MSKFPCORD;
E          END;
E          IF VSNA1 = MSKSSHIG THEN DVMC7 = DVMC7 OR MSKMC7HIG;
E          ELSE IF VSNA1 = MSKSSLOG THEN DVMC7 = DVMC7 OR MSKMC7LOG;
E          ELSE IF VSNA1 = MSKSSOMG THEN DVMC7 = DVMC7 OR MSKMC7OMG;
E              ELSE IF VSNA1 = MSKSSSIVB
E                  THEN DO;
E                      IF FBRNJ
E                          THEN DVMC5 = DVMC5 OR MSKMC54B1I;
E                          ELSE DVMC6 = DVMC6 OR MSKMC68BRI;
E                      END;
E          RETURN; /*MSS60*/
E
MSS70: IF FSSIO THEN WRITE(SS) 0; /*RESET READ COMMAND*/
CALL SSTUPQ(KSSR3,2); /*SCHEDULE HUNG STAGE TEST, MSS05*/
CALL SS201; /*ADVANCE TO NEXT SS*/
VSSW = KSSR1;
E
E          FHST = (VHSTW AND VSTG) = VSTG;
E
E          IF VSNA1 = MSKSSWVO
E              THEN DO;
E              DVASW = DVASW AND NOT MSKSSECS1;
E              DFWV = FALSE;
E          END;
E
E          ELSE IF VSNA1 = MSKSSWVC
E              THEN DO;
E              DVASW = DVASW AND NOT MSKSSECSV;
E              DFWV = TRUE;
E          END;
E
E          ELSE IF VSNA1 = MSKSSSCC
E              THEN DVDPM = DVDPM OR MSKDIN9;

```

HAL KERNEL 10 SWITCH SELECTOR PROCESSING

```

RETURN; /*MSS70*/
E
MSS80:  VSNA = VSCCA;
F
IF FSSIO THEN WRITE(SS) VSNA; /*ISSUE STAGE/COMPLEMENTED ADDR*/
CALL SSTUPQ(KSS87,8); /*SCHEDULE READ COMMAND, MSS55*/
WAIT      ; /*DELAY FOR DOM TELEMETRY*/
WRITE(DOM); /*OUTPUT SS AND DO REGISTERS VIA DOM TELEMETRY*/
RETURN; /*MSS80*/
SS201:  PROCEDURE; /*SS TABLE ADVANCE ROUTINE*/
E
IF FTADV
THEN SST1PTR = SST1PTR + 1;
ELSE SST2PTR = SST2PTR + 1;
CALL SS210; /*SET UP NEXT SWITCH SELECTOR*/
CLOSE SS201;
SS210:  PROCEDURE; /*SS SELECTION AND SETUP ROUTINE*/
E
IF FTADV THEN GO TO SS2020;
SS2160: IF SSTABLE >= 0 THEN GO TO SS2070;
S        1,SST2PTR
E
FCLS4 = FALSE;
E
DVMC6 = DVMC6 AND NOT MSKMC6LUI;
E
DVMC7 = DVMC7 AND NOT MSKMC7T6D;
GO TO SS2090;
SS2020: IF SSTABLE >= 0 THEN GO TO SS2030;
S        1,SST1PTR
E
IF VASPI AND MSKSSSPEC ^= 0
THEN DO;
E
    VASPI = MSKSSS4C0;
E
    DVASW = DVASW AND MSKSSWV;
    SST1PTR = KSSINDXSIVR;
    GO TO SS2020;
END;
E
IF VASPI AND MSKSSCL3 ^= 0
THEN DO;
    SST1PTR = VSC30;
F
    VASPI = VSC31;
    VATRR = VSC32;
    GO TO SS2020;
END;
E
IF VASPI AND MSKSSCL1 ^= 0
THEN DO;
F
    SST1PTR = VSC10;
    VASPI = VSC11;

```

HAL KERNEL 10 SWITCH SELECTOR PROCESSING

```

        VATRR = VSC12;
        GO TO SS2020;
    END;
E
    VASPI = FALSE;
    VATRR = 0;
F
    IF FT60P
    THEN DO;
E
        FT60P = FALSE;
        SST1PTR = KSSINDXTB5A;
    END;
    ELSE SST1PTR = KSSINDXTB5B;
    GO TO SS2020;
F
    SS2030: IF NOT FCLS4 THEN GO TO SS2040;
    SS2070: IF SSTABLE DKRTCSEC/40 + VATRR - KSS500MS >=
S
        1,SST1PTR
S
    SSTABLE DKRTCSEC/40 + VATR4
S
        1,SST2PTR
    THEN DO;
E
        FTADV = FALSE;
        VSSRT = SSTABLE DKRTCSEC/40 + VATR4;
S
        1,SST2PTR
E
        VSNA = BIT(SSTABLE );
S
        2,SST2PTR
    END;
    ELSE
SS2090: DO;
E
        FTADV = TRUE;
    SS2040: VSSRT = SSTABLE DKRTCSEC/40 + VATRR;
S
        1,SST1PTR
E
        VSNA = BIT(SSTABLE );
S
        2,SST1PTR
    END;
E
    SS2050: VHSTW = VSNA AND MSKSSSR;
S
        1 TO 24
    CLOSE SS210;
    SSTUPD: PROCEDURE ASSIGN(TIME); /*SS TIME UPDATE ROUTINE*/
    DECLARE TIME INTEGER;
    READ(CLOCK) ITEMP;
    ITEMP = ITEMP - DVRTC;
    IF ITEMP < 0 THEN ITEMP = ITEMP + DKRTC0VF;
    TIME, DVTRB = DVTGB + DVTRR + ITEMP/4;
    CLOSE SSTUPD;
    SSTUPQ: PROCEDURE(BIAS,ID); /*UPDATE SS TIME AND SCHEDULE SS FUNCT*/
    DECLARE INTEGER, BIAS, ID;
    READ(CLOCK) ITEMP;
    ITEMP = ITEMP - DVRTC;

```


HAL KERNEL 10 SWITCH SELECTOR PROCESSING

```
IF ITEMP < 0 THEN ITEMP = ITEMP + DKRTC0VF;  
DVTRB = DVTGB + DVTRR + ITEMP/4;  
VSSTM = BIAS + DVTRR + ITEMP/4;  
DVSST = VSSTM + DVTMR;  
IF NOT DFIL3 THEN CALL EGP08; /*RESCHEDULE T1 (NOT CODED)*/  
CLOSE SSTUPQ;  
CLOSE MSS00;
```

```

TASKKEY: PROGRAM(PRIORITY,TSKID); /*ATM TASK KEYING ROUTINE*/
        DECLARE INTEGER,
                PRIORITY, /*PRIORITY LEVEL OF TASK BEING KEYED*/
                TSKID, /*IDENTIFICATION INDEX FOR TASK BEING KEYED*/
                I, /*OVERFLOW TABLE POINTER CHAIN INDEX*/
                J; /*OVERFLOW TABLE OPEN-SLOT INDEX*/

C
C PRIORITY CONTROL TABLE CONTAINS ONE ENTRY FOR EACH PRIORITY LEVEL.
C EACH ENTRY CONSISTS OF FIVE ITEMS.
C 1. EITHER THE TASK ID OR THE LOCATION OF THE NEXT
C EXECUTABLE INSTRUCTION OF THE TASK CURRENTLY ASSIGNED
C TO A GIVEN PRIORITY LEVEL. IN ORDER TO DISTINGUISH
C BETWEEN THE TWO THE ID WILL BE STORED AS A NEGATIVE
C VALUE. A VALUE OF ZERO WILL SIGNIFY THAT NO TASK IS
C CURRENTLY ASSIGNED TO THAT PRIORITY LEVEL.
C 2. TASK REGISTER CONTENTS (INITIALLY SET TO ZERO).
C 3. TASK REGISTER CONTENTS (INITIALLY SET TO ZERO).
C 4. TASK REGISTER CONTENTS (INITIALLY SET TO ZERO).
C 5. INDEX POINTER TO THE BEGINNING OF THE PRIORITY OVERFLOW
C TABLE CHAIN FOR THAT PRIORITY LEVEL. A VALUE OF ZERO
C INDICATES END OF CHAIN.
C
        DECLARE ARRAY(5,10) ATMPCT INTEGER;

C
C THE PRIORITY OVERFLOW TABLE IS USED FOR KEYING TASKS ON A
C PRIORITY LEVEL WHICH IS CURRENTLY ASSIGNED TO ANOTHER TASK.
C THE ENTRIES ARE NOT ALLOCATED TO A FIXED PRIORITY BUT ARE
C ASSIGNED DYNAMICALLY AS REQUIRED. ALL OVERFLOW ENTRIES FOR
C EACH PRIORITY LEVEL ARE CHAINED TOGETHER SUCH THAT THE TASKS
C CAN BE EXECUTED ON A FIRST-IN-FIRST-OUT BASIS. EACH ENTRY
C CONSISTS OF TWO ITEMS.
C 1. INDEX POINTER TO THE NEXT ENTRY IN THE CHAIN. A VALUE OF
C ZERO INDICATES END OF CHAIN.
C 2. TASK ID INDEX. A VALUE OF ZERO SIGNIFIES AN UNASSIGNED
C ENTRY.
C
        DECLARE ARRAY(2,25) ATMPOVFT INTEGER;

C
C INHIBIT ALL INTERRUPTS.
C
C IF THE REQUESTED PRIORITY LEVEL IS NOT CURRENTLY ASSIGNED,
C INITIALIZE THE ENTRY FOR THIS TASK.
C
        IF ATMPCT = 0
S          1,PRIORITY
          THEN DO;
S            ATMPCT = - TSKID;
S              1,PRIORITY
              [ATMPCT] = 0;
S              2 TO 4,PRIORITY
          END;

C OTHERWISE, SEARCH FOR THE END OF THE OVERFLOW POINTER CHAIN.
C
        ELSE DO;

```

HAL KERNEL 11 ATM TASK KEYING

```

S          I = ATPCT          ;
          5,PRIORITY
          IF I = 0 THEN
CHAIN_SEARCH:
S          IF ATPOVFT = 0
          1,I
          THEN DO;
          I = ATPOVFT ;
S          1,I
          GO TO CHAIN_SEARCH;
          END;

C
C  WHEN THE END OF THE OVERFLOW POINTER CHAIN HAS BEEN FOUND, SEARCH
C  FOR AN EMPTY SLOT IN THE OVERFLOW TABLE.
C
          DO FOR J = 1 TO 25;
          IF ATPOVFT = 0 THEN GO TO SLOT_FOUND;
S          2,J
          END;

C
C  FALLING THROUGH THE LOOP INDICATES A FULL OVERFLOW TABLE AND SHOULD
C  CAUSE AN ERROR HALT.
C
SLOT_FOUND:
C
C  ADD THIS ENTRY TO THE END OF THE OVERFLOW POINTER CHAIN AND STORE
C  THE TASK POINTER IN IT.
C
          IF I = 0
          THEN ATPOVFT = J;
          1,I
          ELSE ATPCT = J;
          5,PRIORITY
          ATPOVFT = 0;
          1,J
          ATPOVFT = TSKID;
          2,J
S          END;

C
C  RELEASE INTERRUPTS AS REQUIRED
C
          CLOSE TASKKEY;

```

PRECEDING PAGE BLANK NOT FILMED

CMS-2 COMMON DATA DECLARATIONS

```

COMPOOL SYS-DD      'COMMON DATA DECLARATIONS' $
  VRBL DVMLR      F                                $
  VRBL DV1MR      F                                $
  VRBL DKMIR      A 26D S 0 P 163D                $
  VRBL DKRTC0VF   A 26D S 0      P 8192D          $
  VRBL DKRTCSEC   A 26D S 10D   P 4063.492D      $
  VRBL DVA1       A 26D S 4                        $
  VRBL DVA2       A 26D S 4                        $
  VRBL DVA3       A 26D S 4                        $
  VRBL DVA4       A 26D S 4                        $
  VRBL DVA5       A 26D S 4                        $
  VRBL DVA6       A 26D S 4                        $
  VRBL DVMLD     A 26D S 0                        $
  VRBL DVMLT     A 26D S -2                       $
  VRBL DVM05     A 26D S 0                        $
  VRBL DVM06     A 26D S 0                        $
  VRBL DVRTC     A 26D S 0                        $
  VRBL DV SST    A 26D S -2                       $
  VRBL DVTGR     A 26D S -2                       $
  VRBL DVTMM     A 26D S -2                       $
  VRBL DVTMR     A 26D S -2                       $
  VRBL DVTRR     A 26D S -2                       $
  VRBL DVTRR     A 26D S -2                       $
  VRBL DVTT1     A 26D S 0                        $
  TABLE DVCC    V NONE 3 $
                FIELD F1 A 26D S 25D $
  LIKE-TABLE DVDC $
  LIKE-TABLE DVTH $
  END TABLE DVCC $
  VRBL DTBID     I 26D S                          $
  VRBL DVDGS     I 26D S                          $
  VRBL DVEMR     I 26D U                          $
  VRBL DVHDA     I 26D S                          $
  VRBL DVHDB     I 26D S                          $
  VRBL DVLDB     I 26D S                          $
  VRBL DVLRC     I 26D S                          $
  VRBL TEMP      I 26D S                          $
  TABLE DVRE    V NONE 3 $
                FIELD F2 I 26D S                  $
  END-TABLE DVRE $
  VRBL MSKABSLAD I 26D U P 1E3                    $
  VRBL MSKSSCL3  I 26D U P 1E10                   $
  VRBL MSKSSDCS  I 26D U P 5E10                   $
  VRBL MSKSSDCT  I 26D U P 4054E5                 $
  VRBL MSKSSHIG  I 26D U P 10072E4                $
  VRBL MSKSSLOG  I 26D U P 10052E4                $
  VRBL MSKSSNSEND I 26D U P 377777776            $
  VRBL MSKSSOMG  I 26D U P 10007E4                $
  VRBL MSKSSSCC  I 26D U P 10031E4                $
  VRBL MSKSSSIVR I 26D U P 02023E4                $
  VRBL MSKSSSPEC I 26D U P 4E10                   $
  VRBL MSKSS4CO  I 26D U P 2E10                   $
  VRBL MSKSSWVC  I 26D U P 10105E4                $
  VRBL MSKSSWVO  I 26D U P 10145E4                $
  VRBL DFACQ     S 'LOSS', 'GAIN'                 $

```

CMS-2 COMMON DATA DECLARATIONS

```

VRBL DFDBF S 'GOOD', 'FAILED' $
VRBL DFDTL S 'INPROG', 'NOTINPROG' $
VRBL DFLT S 'FLIGHT', 'SIM', 'REP' $
VRBL DFTUP S 'NO', 'YES' $
VRBL DFSMC S 'ENABLE', 'DISABLE' $
VRBL DFVW S 'CLOSE', 'OPEN' $
VRBL DGSSM S 'SS00','SS05','SS10','SS20','SS30'
           'SS40','SS50','SS55','SS60','SS70','SS80'$
VRBL DFIL1 B $
VRBL DFIL2 B $
VRBL DFIL3 B $

```

```

TABLE DVASW V 1 1 $
      FIELD S4CO B 0 25D $
      FIELD SPEC B 0 24D $
      FIELD TB6C B 0 23D $
      FIELD GNSS B 0 22D $
      FIELD SBLO B 0 21D $
      FIELD SBHI B 0 20D $
      FIELD SBOM B 0 19D $
      FIELD ECSV B 0 18D $
      FIELD ECS1 B 0 17D $
      FIELD T3A B 0 16D $
      FIELD TR6D B 0 15D $
      FIELD TB6A B 0 9D $
      FIELD TB6B B 0 8D $
      FIELD S4C1 B 0 7D $
      FIELD ACQU B 0 1D $
      FIELD LI B 0 0D $

```

END-TABLE DVASW \$

```

TABLE DVDPM V 1 1 $
      FIELD DIN24 B 0 25D $
      FIELD DIN23 B 0 24D $
      FIELD DIN22 B 0 23D $
      FIELD DIN21 B 0 22D $
      FIELD DIN20 B 0 21D $
      FIELD DIN19 B 0 20D $
      FIELD DIN18 B 0 19D $
      FIELD DIN17 B 0 18D $
      FIELD DIN16 B 0 17D $
      FIELD DIN15 B 0 16D $
      FIELD DIN14 B 0 15D $
      FIELD DIN13 B 0 14D $
      FIELD DIN12 B 0 13D $
      FIELD DIN11 B 0 12D $
      FIELD DIN10 B 0 11D $
      FIELD DIN9 B 0 10D $
      FIELD DIN8 B 0 9D $
      FIELD DIN7 B 0 8D $
      FIELD DIN6 B 0 7 $
      FIELD DIN5 B 0 6 $
      FIELD DIN4 B 0 5 $
      FIELD DIN3 B 0 4 $
      FIELD DIN2 B 0 3 $
      FIELD DIN1 B 0 2 $

```

END-TABLE DVDPM \$

CMS-2 COMMON DATA DECLARATIONS

```

TABLE DFILE      V 1 1 $
    FIELD COSS      B 0 24D $
    FIELD CORD      B 0 23D $
    FIELD S4CK      B 0 22D $
    FIELD SCGC      B 0 21D $
    FIELD T690      B 0 20D $
    FIELD S2E0      B 0 19D $
    FIELD T6BI      B 0 18D $
    FIELD ISSA      B 0 17D $
    FIELD NU        B 0 16D $
    FIELD EMTL      B 0 15D $
    FIELD TU        B 0 14D $
    FIELD INT2      B 0 13D $
    FIELD LIOG      B 0 12D $
    FIELD FPAT      B 0 11D $
END-TABLE DFILE $
TABLE DVICR      V 1 1 $
    FIELD CA        B 0 13D $
    FIELD SSCB      B 0 11D $
    FIELD SWG       B 0 9D $
    FIELD BG        B 0 3 $
END-TABLE DVICR $
TABLE DVMC4      V 1 1 $
    FIELD ZAC1      B 0 25D $
    FIELD ZAC2      B 0 24D $
    FIELD XAC1      B 0 23D $
    FIELD XAC2      B 0 22D $
    FIELD YAC1      B 0 21D $
    FIELD YAC2      B 0 20D $
    FIELD ZGA1      B 0 19D $
    FIELD ZGA2      B 0 18D $
    FIELD XGA1      B 0 17D $
    FIELD XGA2      B 0 16D $
    FIELD YGA1      B 0 15D $
    FIELD YGA2      B 0 14D $
    FIELD DG1       B 0 13D $
    FIELD DG2       B 0 12D $
    FIELD ACDG      B 0 11D $
    FIELD RCDG      B 0 10D $
    FIELD MLLB      B 0 9D $
    FIELD SSCB      B 0 8D $
    FIELD EAZT      B 0 7 $
    FIELD YAOF      B 0 6 $
    FIELD AMF       B 0 5 $
    FIELD BMF       B 0 4 $
    FIELD ZAOF      B 0 3 $
    FIELD XAOF      B 0 2 $
END-TABLE DVMC4 $
TABLE DVMC5      V 1 1 $
    FIELD TOMC      B 0 25D $
    FIELD T1MC      B 0 24D $
    FIELD ICGM      B 0 23D $
    FIELD RCMC      B 0 22D $
    FIELD ICTA      B 0 20D $
    FIELD T2MC      B 0 19D $

```

CMS-2 COMMON DATA DECLARATIONS

| | |
|-----------------|------------|
| FIELD T3MC | B 0 18D \$ |
| FIELD ICIO | B 0 17D \$ |
| FIELD IC00 | B 0 16D \$ |
| FIELD II9S | B 0 14D \$ |
| FIELD II1G | B 0 13D \$ |
| FIELD IIRC | B 0 12D \$ |
| FIELD T4MC | B 0 11D \$ |
| FIELD II00 | B 0 10D \$ |
| FIELD IIE0 | B 0 9D \$ |
| FIELD DI10 | B 0 8D \$ |
| FIELD ESTG | B 0 7 \$ |
| FIELD S4B1G | B 0 6 \$ |
| FIELD S4B1I | B 0 5 \$ |
| FIELD S4B3G | B 0 4 \$ |
| FIELD S4B7G | B 0 3 \$ |
| FIELD S4BCC | B 0 2 \$ |
| FIELD T5MC | B 0 1 \$ |
| FIELD T6MC | B 0 0 \$ |
| END-TABLE DVMC5 | \$ |
| TABLE DVMC6 | V 1 1 \$ |
| FIELD S8RRI | B 0 25D \$ |
| FIELD S8RTG | B 0 22D \$ |
| FIELD S8RCC | B 0 21D \$ |
| FIELD T7MC | B 0 19D \$ |
| FIELD SCCC | B 0 18D \$ |
| FIELD SD04 | B 0 17D \$ |
| FIELD SC4A | B 0 16D \$ |
| FIELD LUI | B 0 11D \$ |
| FIELD SMC | B 0 10D \$ |
| FIELD RMFA | B 0 9D \$ |
| FIELD TLC | B 0 8D \$ |
| FIELD D04 | B 0 7 \$ |
| FIELD TLI2 | B 0 6 \$ |
| FIELD TLI1 | B 0 5 \$ |
| FIELD T6C | B 0 4 \$ |
| FIELD SCTG | B 0 3 \$ |
| FIELD RMFB | B 0 2 \$ |
| FIELD PABT | B 0 0 \$ |
| END-TABLE DVMC6 | \$ |
| TABLE DVMC7 | V 1 1 \$ |
| FIELD DCSI | B 0 25D \$ |
| FIELD T8EN | B 0 24D \$ |
| FIELD COMM | B 0 21D \$ |
| FIELD LOG | B 0 20D \$ |
| FIELD HIG | B 0 19D \$ |
| FIELD OMG | B 0 18D \$ |
| FIELD NUMC | B 0 17D \$ |
| FIELD TBU | B 0 16D \$ |
| FIELD MTLR | B 0 15D \$ |
| FIELD MIH | B 0 14D \$ |
| FIELD MSC | B 0 13D \$ |
| FIELD TDE | B 0 12D \$ |
| FIELD TARU | B 0 11D \$ |
| FIELD T8RS | B 0 10D \$ |
| FIELD SCLR | B 0 9D \$ |

CMS-2 COMMON DATA DECLARATIONS

| | | | | |
|------------|---|---|----|----|
| FIELD SCAH | B | 0 | 8D | \$ |
| FIELD WCVL | B | 0 | 7 | \$ |
| FIELD M1I | B | 0 | 6 | \$ |
| FIELD M2I | B | 0 | 5 | \$ |
| FIELD M3I | B | 0 | 4 | \$ |
| FIELD M4I | B | 0 | 3 | \$ |
| FIELD M5I | B | 0 | 2 | \$ |
| FIELD M6I | B | 0 | 1 | \$ |
| FIELD M7I | B | 0 | 0 | \$ |

END-TABLE DVMC7 \$
END-SYS-DD COMPOOL \$

MIG00 SYS-PROC 'ITERATIVE GUIDANCE MODE' S

LOC-DD S

```

VRBL CHIBARSTEER S 'INPROG', 'NOTINPROG' S
VRBL PHASE S 'BURN1', 'BURN2' S
VRBL REITERATE S 'YES', 'NO' S
VRBL SMCFLAG S 'NOCALC', 'CALCULATE' S
VRBL S4BURN S 'BURN1', 'BURN2' S
VRBL (COSTHETA, DELTAL3, DELTA2,
      DPHII, DPHIT, EPSILON2,
      EPSILON3, GT, J1,
      J12, J2, J3,
      J3P, K1, K2,
      K3, K4, LYP,
      L1, L12, L2,
      L3, L3P, PHII,
      PHIIT, PHIT, P1,
      P12, P2, Q1,
      Q12, Q2, R,
      ROVEX3, RT, SINTHETA,
      S1, S12, S2,
      TAU1, TAU2, TAU3,
      TCI, T1I, T2I,
      T3I, U1, U12,
      U2, V, VEX1,
      VEX2, VEX3, VT) F S
VRBL KCCT4 F P 1.53D S
VRBL KCCT8 F P 1.55D S
VRBL KMU F P -.39860320E15D S
VRBL KT F P .48497964E-7D S

```

TABLE DELTAVVP V NONE 3 S

FIELD F1 F S

```

LIKE-TABLE GS S
LIKE-TABLE GV S
LIKE-TABLE GVSTAR S
LIKE-TABLE GVT S
LIKE-TABLE G1 S
LIKE-TABLE RS S
LIKE-TABLE RV S
LIKE-TABLE RVT S
LIKE-TABLE R4 S
LIKE-TABLE VS S
LIKE-TABLE VV S
LIKE-TABLE VVT S
LIKE-TABLE V4 S

```

END-TABLE DELTAVVP S

TABLE MS4 A 1 3,3 S

FIELD F2 F S

```

LIKE-TABLE M4V S
END-TABLE MS4 S
(EXTREF) FUNCTION ATAN(ARG) S
(EXTREF) FUNCTION COS(ARG) S
(EXTREF) FUNCTION LOG(ARG) S
(EXTREF) FUNCTION SIN(ARG) S
(EXTREF) FUNCTION SQRT(ARG) S
(EXTREF) PROCEDURE EGP32 INPUT MASK S

```

CMS-2 KERNEL 6 ITERATIVE GUIDANCE MODE

```

(EXTREF) PROCEDURE MATMPY INPUT MATRIX,VEC1 OUTPUT VEC2 $
(EXTREF) PROCEDURE MCM00 $
(EXTREF) PROCEDURE MSM00 $
END-LOC-DD $
PROCEDURE MIG00 $
COMMENT $
COMMENT DUE TO THE SIZE OF IGM, ONLY A SECTION OF IT HAS $
COMMENT BEEN CODED. PART OF THE GUIDANCE COMPUTATIONS HAVE $
COMMENT BEEN SELECTED TO DEMONSTRATE MATHEMATICAL OPERA- $
COMMENT TIONS. THE PHASING PORTION OF IGM HAS NOT BEEN $
COMMENT CODED SINCE SIMILAR CAPABILITIES ARE ILLUSTRATED $
COMMENT BY OTHER KERNELS. $
COMMENT $
COMMENT IG251 - IGM GUIDANCE PARAMETERS COMPUTATIONS $
COMMENT $
COMMENT ROTATE POSITION AND VELOCITY INTO TARGET PLANE $
COMMENT $
IG253. MATMPY INPUT CORAD(MS4), CORAD(RS) OUTPUT CORAD(R4)$
UTR00 $ 'DELAY FOR TELEMETRY AS REQUIRED' $
COMMENT TELEMETER X POSITION IN 4 SYSTEM, R4(0) $
UTR00 $ 'DELAY FOR TELEMETRY AS REQUIRED' $
COMMENT TELEMETER Y POSITION IN 4 SYSTEM, R4(1) $
COMMENT RELEASE INTERRUPTS DISABLED BY TELEM DELAY ROUTINE $
MATMPY INPUT CORAD(MS4), CORAD(VS) OUTPUT CORAD(V4)$
UTR00 $ 'DELAY FOR TELEMETRY AS REQUIRED' $
COMMENT TELEMETER Z POSITION IN 4 SYSTEM, R4(2) $
UTR02 $ 'DELAY FOR TELEMETRY AS REQUIRED' $
COMMENT TELEMETER Y VELOCITY IN 4 SYSTEM, V4(1) $
COMMENT RELEASE INTERRUPTS DISABLED BY TELEM DELAY ROUTINE $
COMMENT $
COMMENT CALCULATE RANGE ANGLE MEASURED IN ORBIT PLANE $
COMMENT $
IG254. IF T2I EQ 0
      THEN SET L12,J12,S12,Q12,P12,U12 TO 0
      THEN GOTO IG259 $
IF T1I EQ 0
      THEN SET L1,J1,S1,Q1,P1,U1 TO 0
      THEN GOTO IG258 $
SET L1 TO VEX1*LOG(TAU1/(TAU1 - T1I)) $
SET J1 TO L1*TAU1 - VEX1*T1I $
SET S1 TO L1*T1I - J1 $
SET Q1 TO S1*TAU1 - .5D*VEX1*T1I**2 $
SET P1 TO J1*TAU1 - .5D*VEX1*T1I**2 $
SET U1 TO Q1*TAU1 - VEX1*T1I**3/6 $
IG258. SET L2 TO VEX2*LOG(TAU2/(TAU2 - T2I)) $
SET J2 TO L2*TAU2 - VEX2*T2I $
SET S2 TO L2*T2I - J2 $
SET Q2 TO S2*TAU2 - .5D*VEX2*T2I**2 $
SET P2 TO J2*TAU2 - .5D*VEX2*T2I**2 $
SET U2 TO Q2*TAU2 - VEX2*T2I**3/6 $
SET L12 TO L1 + L2 $
SET J12 TO J1 + J2 + L2*T1I $
SET S12 TO S1 - J2 + L12*(T2I + TCI) $
SET Q12 TO Q1 + Q2 + S2*T1I + J1*T2I $
SET P12 TO P1 + P2 + T1I*(2*J2 + L2*T1I) $

```

CMS-2 KERNEL 6 ITERATIVE GUIDANCE MODE

```

IG259. SET U12 TO U1 + U2 + T1I*(2*Q2 + S2*T1I) + T2I*P1 $
        SET L3P TO VEX3*LOG(TAU3/(TAU3 - T3I)) $
        SET LYP TO L12 + L3P $
        SET J3P TO L3P*TAU3 - VEX3*T3I $
        SET T1C TO T1I + T2I + TCI $
        SET TSTAR TO T1C + T3I $
        SET PHII TO ATAN(R4(2)/R4(0)) $

COMMENT $
COMMENT DETERMINE PHASE $
COMMENT $
IG260. IF PHASE EQ 'BURN2' THEN GOTO IG262 $
        SET DELTA2 TO V*TSTAR - J3P + LYP*T3I - ROVEX3*((TAU1
            - T1I)*L1 + (TAU2 - T2I)*L2 + (TAU3 - T3I)
            *L3P)*(LYP + V - VT) $
        SET PHIIT TO KT*(S12 + DELTA2) $
        SET PHIT TO PHII + PHIIT $
        UTR02 $ 'DELAY FOR TELEMETRY AS REQUIRED' $
        COMMENT TELEMETER TERMINAL RANGE ANGLE, PHIT $
        COMMENT RELEASE INTERRUPTS DISABLED BY TELEM DELAY ROUTINE $
        GOTO IG291 $
IG262. SET SINTHETA TO (RS(0)*VS(0) + RS(1)*VS(1) + RS(2)*VS(2))
        /(R*V) $
        SET COSTHETA TO SQRT(1 - SINTHETA**2) $
        SET DPHII TO V/R*COSTHETA $
        SET DPHIT TO VT/RT*COS(THETAT) $
        SET PHIIT TO .5D*(DPHII + DPHIT)*TSTAR $
        SET PHIT TO PHII + PHIIT $
        UTR02 $ 'DELAY FOR TELEMETRY AS REQUIRED' $
        COMMENT TELEMETER TERMINAL RANGE ANGLE, PHIT $
        COMMENT RELEASE INTERRUPTS DISABLED BY TELEM DELAY ROUTINE $
        IF TSTAR LTEQ EPSILON3 THEN GOTO IG269 $
        MIG30 $ 'CALC TERM RAD, VEL, FLT ANGLE(NOT CODED)' $
        SET GT TO - KMU/RT**2 $
        UTR00 $ 'DELAY FOR TELEMETRY AS REQUIRED' $
        COMMENT TELEMETER TERMINAL GRAVITY VECTOR, GT $
        COMMENT RELEASE INTERRUPTS DISABLED BY TELEM DELAY ROUTINE $
        SET GVT(0) TO GT*COS(THETAT) $
        SET GVT(1) TO 0 $
        SET GVT(2) TO GT*SIN(THETAT) $
        SET RVT(0) TO RT*COS(THETAT) $
        SET RVT(1), RVT(2) TO 0 $
        SET PHIT = PHIT - THETAT $

COMMENT $
COMMENT ROTATE POSITION, VELOCITY, GRAVITY TO INJECTION SYSTEM' $
COMMENT $
IG291. SET M4V(0), M4V(8) TO COS(PHIT) $
        SET M4V(2) TO SIN(PHIT) $
        SET M4V(6) TO - SIN(PHIT) $
        SET M4V(1), M4V(3), M4V(5), M4V(7) TO 0 $
        SET M4V(4) TO 1 $
        MATMPY INPUT CORAD(M4V), CORAD(R4) OUTPUT CORAD(RV) $
        MATMPY INPUT CORAD(M4V), CORAD(V4) OUTPUT CORAD(VV) $
        MATMPY INPUT CORAD(MS4), CORAD(G8) OUTPUT CORAD(G1) $
        MATMPY INPUT CORAD(M4V), CORAD(G1) OUTPUT CORAD(GV) $
IG293. VARY I FROM 0 THRU 2 $

```

```

        SET GVSTAR(I) TO .5*(GVT(I) + GV(I)) $
        SET DELTAVVP(I) TO VVT(I) - VV(I) - TSTAR*GVSTAR(I)$
    END IG293 $

COMMENT
COMMENT IG314 - CALCULATE TIME TO GO (NOT CODED) $
COMMENT $

    IF REITERATE EQ 'YES'
        THEN SET REITERATE TO 'NO'
        THEN SET L3P TO L3
        THEN SET J3P TO J3
        THEN SET LYP TO LYP + DELTAL3
        THEN GOTO IG260 $
    SET REITERATE TO 'YES' $

COMMENT
COMMENT IG324 - COMPUTE CORRECTED VEL TO BE GAINED(NOT CODED)$
COMMENT $
COMMENT IG326 - CALCULATE DESIRED PITCH AND YAW (NOT CODED)$
COMMENT $

    IF CHIBARSTEER EQ 'INPROG' THEN GOTO IG350 $
    IF TSTAR GTEQ EPSILON2 THEN GOTO IG360 $
    IF S4RURN EQ 'BURN1'
        THEN SET DVMC5(0,CBS) TO 1
        THEN SET DVMLR TO 25D*KCCT4
        THEN SET DV1MR TO .04D/KCCT4
        THEN GOTO IG340 $
    SET DVMC6(0,CBS) TO 1 $
    SET DVMLR TO 25D*KCCT8 $
    SET DV1MR TO .04D/KCCT8 $
IG340. SET CHIBARSTEER TO 'INPROG' $
IG350. SET K1, K2, K3, K4 TO 0 $
        GOTO IG440 $

COMMENT
COMMENT IG360. ''IG361 - COMPUTE INTERMEDIATE PARAMETERS(NOT CODED)'' $
COMMENT $
IG440. UTR00 $ ''DELAY FOR TELEMETRY AS REQUIRED''
COMMENT TELEMETER T3I $
COMMENT RELEASE INTERRUPTS DISABLED BY TELEM DELAY ROUTINE $
COMMENT $
COMMENT IG446 - COMPUTE PITCH AND YAW IN 4-SYSTEM(NOT CODED)$
COMMENT $

    IF SMCFLAG EQ 'CALCULATE'
        THEN MSM00 $ ''COMPUTE SMC TERMS (NOT CODED)''
    MCM00 $ ''PERFORM CHI COMPUTATIONS (NOT CODED)''
    IF DFILE(0,INT2)
        THEN EGP32 INPUT MSKSCCO $ ''ENABLE INTERRUPT 2''
    RETURN $ ''MIG00''

END-PROC MIG00 $
END-SYS-PROC MIG00 $

```

CMS-2 KERNEL 7 DIGITAL COMMAND SYSTEM

```

MDS00  SYS-PROC  'DIGITAL COMMAND SYSTEM' $
LOC-DD  $
SWITCH DCSRET (DCSRETURN) $
    'NORMAL', DS530 $
    'ERROR1', DS220 $
    'ERROR2', DS235 $
END-SWITCH DCSRET $
P-SWITCH DCS $
P DS105 $ 'ERROR PATH'
P DS260 $ 'TIME BASE UPDATE (NOT CODED)'
P DS330 $ 'NAVIGATION UPDATE (NOT CODED)'
P DS380 $ 'GENERALIZED SS (NOT CODED)'
P DS430 $ 'SECTOR DUMP (NOT CODED)'
P DS470 $ 'SINGLE MEM LOC TELEM (NOT CODED)'
P DS510 $ 'TERMINATE (NOT CODED)'
P DS540 $ 'MANEUVER UPDATE (NOT CODED)'
P DS550 $ 'MANEUVER INHIBIT (NOT CODED)'
P DS670 $ 'TARGET UPDATE (NOT CODED)'
P DS700 $ 'ANTENNAE TO OMNI (NOT CODED)'
P DS720 $ 'ANTENNAE TO LOW (NOT CODED)'
P DS740 $ 'ANTENNAE TO HIGH (NOT CODED)'
P DS770 $ 'INHIBIT WATER CONTROL (NOT CODED)'
P DS790 $ 'TIME BASE B ENABLE (NOT CODED)'
P DS810 $ 'EXECUTE MANEUVER A (NOT CODED)'
P DS840 $ 'TD AND E ENABLE (NOT CODED)'
P DS860 $ 'EXECUTE MANEUVER B (NOT CODED)'
P DS900 $ 'S4B/IU LUNAR IMPACT (NOT CODED)'
P DS960 $ 'ENABLE TR6D ALT SEQ (NOT CODED)'
END-P-SW DCS $
VRBL DCSDATACOUNT I 26D $
VRBL DCSERLIM I 26D P 7 $
VRBL DCSER04 I 26D P 040000000 $
VRBL DCSER10 I 26D P 100000000 $
VRBL DCSER14 I 26D P 140000000 $
VRBL DCSER20 I 26D P 200000000 $
VRBL DCSER24 I 26D P 240000000 $
VRBL DCSER44 I 26D P 440000000 $
VRBL DCSER60 I 26D P 600000000 $
VRBL DCSER64 I 26D P 640000000 $
VRBL DCSER74 I 26D P 740000000 $
VRBL DCSINDX I 26D $
VRBL DCSRETURN S 'NORMAL', 'ERROR1', 'ERROR2' $
VRBL FDSER S 'MODE', 'DATA' $
VRBL FDSPG S 'INPROG', 'NOTINPROG' $
VRBL FDSRE S 'TERM', 'NOTERM' $
VRBL VDSER I 26D $
VRBL VDSRC I 26D $
VRBL VDSSB B $
VRBL VDS01 I 26D $
TABLE DCSDATCT V 1 20D $
    FIELD F1 I 26D S $
LIKE-TABLE DCSSTCOD $
END-TABLE DCSDATCT $
TABLE DCSMODE V 1 64D $
    FIELD F2 I 26D S $

```

CMS-2 KERNEL 7 DIGITAL COMMAND SYSTEM

```

END-TABLE DCSMODE $
TABLE DCSMSTAT V 1 20D $
FIELD F3 S 'ACTIVE', 'INACTIVE' $
END-TABLE DCSMSTAT $
TABLE VDSBL V 1 35D $
FIELD F4 I 26D U $
END-TABLE VDSBL $
DCSSTCOD DATA 000000000 $
DATA 100000000 $
DATA 110000000 $
DATA 120000000 $
DATA 130000000 $
DATA 140000000 $
DATA 200000000 $
DATA 220000000 $
DATA 050000000 $
DATA 310000000 $
DATA 770000000 $
DATA 770000000 $
DATA 770000000 $
DATA 450000000 $
DATA 170000000 $
DATA 330000000 $
DATA 600000000 $
DATA 340000000 $
DATA 520000000 $
DATA 250000000 $
DCSDATCT DATA 0 $
DATA 1 $
DATA 35D $
DATA 2 $
DATA 2 $
DATA 3 $
DATA 0 $
DATA 0 $
DATA 0 $
DATA 35D $
DATA 0 $
DATA 0 $
DATA 0 $
DATA 0 $
DATA 0 $
DATA 0 $
DATA 0 $
DATA 0 $
DATA 0 $
DATA 6 $
DATA 0 $
DCSMODE DATA 0 $
DATA 0 $
DATA 0 $
DATA 0 $
DATA 0 $
DATA 8D $
DATA 0 $
DATA 0 $

```


CMS-2 KERNEL 7 DIGITAL COMMAND SYSTEM

```

          DATA      0      $
COM EQUALS 0        $
LEN EQUALS 6        $
MSG EQUALS 12D     $
MLEN EQUALS 14D    $
SEQ EQUALS 6        $
TERM EQUALS 20000000 $
END-LOC-DD $
PROCEDURE MDS00 $
COMMENT RELEASE PREVIOUSLY ENABLED INTERRUPTS $
COMMENT READ DISCRETE INPUT REG INTO TEMP $
COMMENT READ DIGITAL COMMAND SYSTEM INPUT INTO VDS01 $
        IF BIT(22D)(TEMP) EQ 0 THEN GOTO DS60 $
COMMENT $
COMMENT PROCESS DCS MODE COMMAND $
COMMENT $
DS09.   VARY I FROM COM THRU LEN $
        IF BIT(I)(VDS01) EQ BIT(I+7)(VDS01)
            THEN SET VDSER TO DCSE10
            THEN GOTO DS220 $
        END DS09 $
        IF BIT(SEQ)(VDS01) EQ 1
            THEN SET VDSER TO DCSE24
            THEN GOTO DS220 $
        IF BIT(COM,LEN)(VDS01) EQ TERM THEN GOTO DS25 $
        IF FDSEN EQ 'DATA'
            THEN SET VDSER TO DCSE20
            THEN GOTO DS220 $
        IF DFDTL EQ 'INPROG' OR FDSPG EQ 'INPROG'
            THEN SET VDSER TO DCSE64
            THEN GOTO DS220 $
DS20.   SET FDSPG TO 'INPROG' $
DS25.   SET DCSINDX TO DCSEMODE(BIT(COM,LEN)(VDS01)) $
        IF DCSMSTAT(DCSINDX) EQ 'INACTIVE'
            THEN SET FDSPG TO 'NOTINPROG'
            THEN SET VDSER TO DCSE74
            THEN GOTO DS220 $
COMMENT TELEMETER STATUS CODE TWICE $
        UTR24 $ 'DELAY FOR TELEMETRY AS REQUIRED' $
COMMENT TELEMETER DCSSTCOD(DCSINDX) $
        UTR24 $ 'DELAY FOR TELEMETRY AS REQUIRED' $
COMMENT TELEMETER DCSSTCOD(DCSINDX) $
COMMENT RELEASE INTERRUPTS DISABLED BY TELEM DELAY ROUTINE $
        DS200 $ 'ISSUE CRP' $
        SET DCSDATACOUNT TO 0 $
        SET VDSSB TO 0 $
        GOTO DS100 $
COMMENT $
COMMENT PROCESS DATA WORD $
COMMENT $
DS60.   IF FDSEN EQ 'MODE'
            THEN SET VDSER TO DCSE04
            THEN GOTO DS220 $
DS61.   VARY I FROM COM THRU LEN $
        IF BIT(I)(VDS01) EQ BIT(I+7)(VDS01)

```



```

        THEN SET VDSER TO DC SER44
        THEN GOTO DS220 $
END DS61 $
IF BIT(SEQ)(VDS01) NOT VDSSB
    THEN SET VDSER TO DC SER60
    THEN GOTO DS220 $
DS110.  'TELEMETER DATA WORD TWICE'
COMMENT UTR24 $ 'DELAY FOR TELEMETRY AS REQUIRED'
TELEMETER VDS01 $
COMMENT UTR24 $ 'DELAY FOR TELEMETRY AS REQUIRED'
TELEMETER VDS01 $
COMMENT RELEASE INTERRUPTS DISABLED BY TELEM DELAY ROUTINE $
COMMENT DS200 $ 'ISSUE CRP'
SET VDSBL(DCS DATA COUNT) TO BIT(COM,LEN)(VDS01) $
SET VDSSB TO COMP VDSSB $
DS100.  SET DCS DATA COUNT TO DCS DATA COUNT + 1 $
IF DCS DATA COUNT LT DCS DATCT(DCS INDX) THEN RETURN $
SET DCS RETURN TO 'NORMAL' $
DCS USING DCS INDX INVALID DS105 $
GOTO DCS RET DCS RETURN $
DS105.  SET FDSPG TO 'NOT IN PROG' $
SET VDSER TO DC SER14 $
COMMENT $
COMMENT PROCESS DCS ERROR CONDITION $
COMMENT $
DS220.  SET VDSRC TO VDSRC + 1 $
SET FDSRE TO 'TERM' $
IF VDSRC LT DC SER LIM THEN SET VDSRC TO 'NOTERM' $
SET VDSER TO VDSER + VDSRC $
SET BIT(MESG,MLEN)(VDSER) TO BIT(COM,MLEN)(VDS01) $
DS235.  'TELEMETER ERROR CODE TWICE'
COMMENT UTR24 $ 'DELAY FOR TELEMETRY AS REQUIRED'
TELEMETER VDSER $
COMMENT UTR24 $ 'DELAY FOR TELEMETRY AS REQUIRED'
TELEMETER VDSER $
COMMENT RELEASE INTERRUPTS DISABLED BY TELEM DELAY ROUTINE $
COMMENT IF FDSRE EQ 'NOTERM' THEN RETURN $ 'MDS00'
DS530.  SET VDSRC TO 0 $
SET FDSER TO 'MODE' $
SET FDSPG TO 'NOT IN PROG' $
RETURN $ 'MDS00'
END-PROC MDS00 $
PROCEDURE DS200 $ 'ISSUE DCS COMMAND RESET PULSE'
COMMENT $
COMMENT INHIBIT ALL INTERRUPTS EXCEPT TLC $
COMMENT $
COMMENT ISSUE COMMAND RESET PULSE $
COMMENT $
COMMENT DELAY 4.13 MS $
COMMENT $
COMMENT RESET THE COMMAND RESET PULSE $
COMMENT $
COMMENT RELEASE PREVIOUSLY ENABLED INTERRUPTS $
COMMENT $
RETURN $ 'DS200'

```

CMS-2 KERNEL 7 DIGITAL COMMAND SYSTEM

END-PROC DS200\$
END-SYS-PROC MDS00 \$

```

MML00 SYS-PROC 'MINOR LOOP' $
LOC-DD $
(EXTREF) PROCEDURE MDG00 OUTPUT J EXIT ERR $
SWITCH MLSW1 ML201, ML101, ML001A $
SWITCH MLSW2 ML4352, ML4351, ML4350 $
SWITCH MLSW3 ML245, ML145, ML045 $
SWITCH MLSW4 ML260, ML160, ML060 $
VRBL, FBUGS I 26D S $
VRBL, KCPRG A 26D S 14D P 2016D $
VRBL, VCG10 I 12D S $
VRBL, VCG11 I 12D S $
VRBL, VIRE I 26D S $
VRBL, VMEMR I 26D U $
VRBL, VOCK A 26D S 25D $
TABLE FRUG V NONE 3 $
FIELD F1 I 26D S $
LIKE-TABLE VGR 3 $
LIKE-TABLE VPGR 3 $
END-TABLE FBUG $
TABLE VCOD V NONE 3 $
FIELD F2 I 12D S $
LIKE-TABLE VOLD 3 $
LIKE-TABLE VCG0 3 $
LIKE-TABLE VCG1 3 $
LIKE-TABLE VMLO 3 $
LIKE-TABLE VML1 3 $
END-TABLE VCOD $
TABLE VCG V NONE 3 $
FIELD F3 A 26D S 25D $
LIKE-TABLE VDEL 3 $
LIKE-TABLE VBUB 3 $
LIKE-TABLE VML2 3 $
END-TABLE VCG $
TABLE VSF V NONE 3 $
FIELD F4 A 26D S 35D $
END-TABLE VSF $
TABLE VFIO V NONE 3 $
FIELD F5 S 'NORMAL', 'BACKUP', 'DUMMY' $
END-TABLE VFIO $
TABLE VMLET V DENSE 1 $
FIELD OLD I 12D S 0 25D $
FIELD TAG I 3D S 0 13D $
FIELD COD I 11D S 0 10D $
END-TABLE VMLET $
TABLE VCMND A 1 3,3 $
FIELD F6 I 26D S $
END-TABLE VMLET $
ERTAG EQUALS 7 $
PI EQUALS 40000000 $
END-LOC-DD $
PROCEDURE MML00 $ 'FLIGHT SIMULATION MINOR LOOP'
IF DVLRC EQ 0
THEN SET DVCC TO DVCC - DVDC
THEN GOTO MLO $
SET DVLRC TO DVLRC - 1 $

```

CMS-2 KERNEL 9 MINOR LOOP

```

ML0.      MML20 $ 'EXECUTE NORMAL MINOR LOOP''
          RETURN $ 'MML00''
END-PROC MML00 $
(EXTDEF) PROCEDURE MML20 $ 'NORMAL MINOR LOOP''
          SET DVCC TO DVCC + DVDC $
          IF FRUGS NOT 0 THEN GOTO ML500 $
ML001.    VARY I FROM 2 THRU 0 BY -1 $
          GOTO MLSW1 I $
ML001A.   IF VFIO(2) EQ 'NORMAL'
          THEN 'READ Z GIMBAL INTO VGR(2)''
          THEN GOTO ML004 $
          IF VFIO(2) EQ 'BACKUP'
          THEN 'READ Z BACKUP GIMBAL INTO VGR(2)''
          THEN GOTO ML004 $
          SET VGR(2) TO VPGR(2) $
          GOTO ML004 $
ML101.    'READ ERROR MONITOR REGISTER INTO VMEMR''S
          SET DVLDB TO DVLDB - BIT(17D)(VMEMR) $
          IF VFIO(1) EQ 'NORMAL'
          THEN 'READ Y GIMBAL INTO VGR(1)''
          THEN GOTO ML004 $
          IF VFIO(1) EQ 'BACKUP'
          THEN 'READ Y BACKUP GIMBAL INTO VGR(1)''
          THEN GOTO ML004 $
          SET VGR(1) TO VPGR(1) $
          GOTO ML004 $
ML201.    VARY J FROM 0 THRU 250 $
          IF VMEMR(0,J) EQ 1 THEN SET DVEMR(0,J) TO 1S
          END ML201 $
          IF VFIO(0) EQ 'NORMAL'
          THEN 'READ X GIMBAL INTO VGR(0)''
          THEN GOTO ML004 $
          IF VFIO(0) EQ 'BACKUP'
          THEN 'READ X BACKUP GIMBAL INTO VGR(0)''
          THEN GOTO ML004 $
          SET VGR(0) TO VPGR(0) $
ML004.    IF BIT(0)(VGR(I)) EQ 0 THEN GOTO ML020 $
          IF DVDGS LT 0 THEN GOTO ML432 $
          IF ID DVDGS EQ 0 THEN GOTO ML020 $
          GOTO ML637 $
ML432.    MDG00 OUTPUT J EXIT ML434 $ 'PROCESS
          DISAGREEMENT BIT (NOT CODED)''
          COMMENT DISAGREEMENT BIT PROCESSING WILL TAKE A NORMALS $
          COMMENT RETURN IF THE DISAGREEMENT BIT IS FOUND TO BE $
          COMMENT INVALID, OTHERWISE, IT WILL TAKE THE ERROR $
          COMMENT EXIT TO ML434 AND SET J=0 IF THE GIMBAL IS $
          COMMENT 'VALID OR J=1 IF THE GIMBAL IS NOT VALID. $
          COMMENT $
          GOTO ML020 $
ML434.    GOTO MLSW2 I $
ML4350.   IF VFIO(2) EQ 'NORMAL'
          THEN 'READ Z GIMBAL, RESTART Y COD COUNTER''
          THEN GOTO ML450 $
          'READ Z BACKUP GIMBAL, RESTART Y COD COUNTER''S

```

```

GOTO ML450 $
ML4351. IF VFIO(1) EQ 'NORMAL'
        THEN 'READ Y GIMBAL, RESTART X COD COUNTER'
        THEN GOTO ML450 $
        'READ Y BACKUP GIMBAL, RESTART X COD COUNTER'
GOTO ML450 $
ML4352. IF VFIO(0) EQ 'NORMAL'
        THEN 'READ X GIMBAL, RESTART Z COD COUNTER'
        THEN GOTO ML450 $
        'READ X BACKUP GIMBAL, RESTART Z COD COUNTER'
ML450. IF J EQ 1 THEN GOTO ML637 $
ML020. SET VCOD(I) TO BIT(1,11D)(VGR(I)) $
        IF VCOD(I) EQ 0 AND VOLD(I) EQ 0 AND ABS(VDEL(I))
            GTEQ VOCK THEN GOTO ML631 $
        IF ABS(VCOD(I) - VOLD(I)) LT VML0(I)
            THEN GOTO ML040 $
        IF ABS(VCOD(I) - VOLD(I)) + VML0(I) LT VML1(I)
            THEN GOTO ML630 $
        IF VCOD(I) LT VOLD(I)
            THEN SET VCG(I) TO VCG(I) + VML2(I)
            THEN GOTO ML040 $
        SET VCG(I) TO VCG(I) - VML2(I) $
ML040. SET DVTH(I) TO VSF(I)*VCOD(I) + VCG(I) $
        SET VOLD(I) TO VCOD(I) $
        SET VDEL(I) TO DVTH(I) - DVCC(I) $
        SET DFDRF TO 'GOOD' $
        GOTO MLSW3 I $
ML045. SET VCMND(2,0) TO DVA5*VDEL(2) - DVA4*VDEL(1) $
        GOTO ML730 $
ML145. SET VCMND(1,0) TO DVA1*VDEL(1) + DVA2*VDEL(2) $
        GOTO ML730 $
ML245. SET VCMND(0,0) TO DVA6*(VDEL(0) + DVA3*VDEL(1)) $
        GOTO ML730 $
ML630. SET VMLET(0,TAG) TO I + 3 $
        GOTO ML632 $
ML631. SET VMLET(0,TAG) TO I $
ML632. SET VMLET(0,COD) TO VCOD(I) $
        SET VMLET(0,OLD) TO VOLD(I) $
        IF COMP DVMC6(0,D04)
            THEN UTR30 'DELAY FOR TELEMETRY AS REQUIRED'
            THEN 'TELEMETER ERROR MESSAGE' $
        IF DFDBF EQ 'FAILED' THEN GOTO ML635 $
        SET DVRE(I) TO DVRE(I) + 1 $
        IF DVRE(I) LT 0 THEN GOTO ML637 $
        IF DVRE(I) GT 0 THEN GOTO ML636 $
        SET VMLET(0,TAG) TO ERTAG $
        SET VMLET(0,COD) TO VCOD(I) $
        SET VMLET(0,OLD) TO VOLD(I) $
        SET VFIO(I) TO 'BACKUP' $
        IF VCG(I) GTEQ PI
            THEN SET VCG(I) TO PI - VRUB(I)
            THEN GOTO ML633 $
        SET VCG(I) TO - VRUB(I) $
ML633. SET VML2(I) TO PI $
        IF DVTH(I) GTEQ PI

```

```

        THEN SET VOLD(I) TO (DVTH(I) - PI)*KCPBG
        THEN GOTO ML634 $
ML634.  SET VOLD(I) TO DVTH(I)*KCPBG $
        SET VSF(I) TO 1/KCPBG $
        IF I = 2
            THEN ''SET ICR TO SELECT BACKUP GIMBAL''
            THEN SET DVICR(0,BG) TO 1 $
        SET FBUGS, FBUG(I) TO 2 $
        SET VMLO(I) TO VCG10 $
        SET VML1(I) TO VCG11 $
        IF COMP DVMC6(0,DO4)
            THEN UTR30 ''DELAY FOR TELEMETRY AS REQUIRED''
            THEN ''TELEMETER ERROR MESSAGE'' $
        GOTO ML637 $
ML635.  SET DVHDB TO DVHDB - 1 $
        SET DVHDA TO DVHDA + 1 $
        SET DFDRF TO 'GOOD' $
        IF DVHDA LT 0 THEN GOTO ML636 $
COMMENT SET ICR TO SWITCH GIMBAL ORDER $
        SET DVICR(0,SWG) TO 1 $
        SET DVMC4(0,AMF) TO 1 $
        SET DVDGS TO 0 $
ML636.  IF DVRE(I) GTEQ VIRE AND COMP DVMC6(0,DO4)
            THEN UD000 INPUT MSKGRF $ ''SET GUIDANCE
            REFERENCE FAILURE DISC(NOT CODED)''
ML637.  SET DFSMC TO 'DISABLE' $
        GOTO ML760 $
ML730.  IF ABS(VCMND(I,0)) GR DVM06
            THEN SET VCMND(I,0) TO DVM06 $
        IF ABS(VCMND(I,0) - VCMND(I,1)) GR DVM05
            THEN SET VCMND(I,0) TO VCMND(I,1) + DVM05 $
        SET VCMND(I,1) TO VCMND(I,0) $
        IF VCMND(I,0) LT 0
            THEN SET VCMND(I,2) TO MSKABSLAD - VCMND(I,0)
            THEN GOTO ML760 $
        SET VCMND(I,2) TO VCMND(I,0) $
ML760.  GOTO MLSW4 I $
ML060.  ''ISSUE YAW COMMAND FROM VCMND(2,2)'' $
        IF DVLDB LT 0 THEN ''SET ICR TO SELECT CONV A'' $
        SET DVTT1 TO DVTT1 - DVRTC $
        IF DVTT1 LT 0 THEN SET DVTT1 TO DVTT1 + DKRTC0VF $
        SET DVMLT TO DVTMM + DVMLD + DVTT1 $
        GOTO ML900 $
ML160.  ''ISSUE PITCH COMMAND FROM VCMND(1,2)'' $
        GOTO ML900 $
ML260.  ''START SPECIAL DOM BACKUP GIMBAL'' $
        ''ISSUE YAW COMMAND FROM VCMND(2,2)'' $
        ''ISSUE ROLL COMMAND FROM VCMND(0,2)'' $
ML900.  END ML00 $
        RETURN $ ''MML20''
ML500.  VARY I FROM 0 THRU 2 $
        IF FBUG(I) EQ 0 THEN GOTO ML530 $
        IF FBUG(I) EQ 1
            THEN SET FBUG(I) TO 0
            THEN SET VMLO(I) TO VCG0(I)

```

CMS-2 KERNEL 9 MINOR LOOP

```
                THEN SET VML1(I) TO VCB1(I)
                THEN GOTO ML530 $
                SET FBUG(I) TO 1 $
ML530.          END ML500 $
                SET FBUGS TO FBUG(0) + FBUG(1) + FBUG(2) $
                GOTO ML001 $
END-PROC MML20 $
END-SYS-PROC MML00 $
```

MSS00 SYS-PROC 'SWITCH SELECTOR PROCESSOR' \$
 LOC-DDS

(EXTREF) PROCEDURE EGP08 \$

SWITCH SS (DGSSM)\$

'SS00', SS00 \$ 'SS ALTERNATE SEQUENCE CHECK'
 'SS05', MSS05 \$ 'SS NORMAL CHECK'
 'SS10', MSS10 \$ 'SS TIME BASE SET INITIALIZE'
 'SS20', MSS20 \$ 'SS FORCED RESET'
 'SS30', MSS30 \$ 'SS HUNG STAGE TEST'
 'SS40', MSS40 \$ 'SS STAGE/ADDRESS ISSUANCE'
 'SS50', MSS50 \$ 'SS VERIFY ADDRESS'
 'SS55', MSS55 \$ 'SS READ TIME CHECK'
 'SS60', MSS60 \$ 'SS READ ISSUANCE'
 'SS70', MSS70 \$ 'SS RESET'
 'SS80', MSS80 \$ 'SS COMPLEMENT STAGE/ADDRESS'

END-SWITCH SS \$

VRBL RIAS A 26D S -2 \$
 VRBL ID I 26D U \$
 VRBL TIME A 26D S -2 \$
 VRBL FASE S 'NORMAL', 'ALTERNATE' \$
 VRBL FBRNI S 'FIRST', 'SECOND' \$
 VRBL FCLS4 S 'NOTINPROG', 'INPROG' \$
 VRBL FFBCH S 'CHANA', 'CHANB' \$
 VRBL FHST S 'NOTEST', 'TEST' \$
 VRBL FSSIO B \$
 VRBL FSSAC S 'INACTIVE', 'ACTIVE' \$
 VRBL FTADV S 'NORMAL', 'CLASS4' \$
 VRBL FT60P S 'PASS1', 'PASS2' \$
 VRBL KCSSK 26D A -2 P 812.69840D \$
 VRBL KSSB1 26D A -2 P 70.142856D \$
 VRBL KSSB2 26D A -2 P 103.58730D \$
 VRBL KSSB3 26D A -2 P 66.206348D \$
 VRBL KSSB4 26D A -2 P 35.825396D \$
 VRBL KSSB5 26D A -2 P 102.65079D \$
 VRBL KSSB6 26D A -2 P 50.825396D \$
 VRBL KSSB7 26D A -2 P 87.460316D \$
 VRBL KSSB8 26D A -2 P 43.825396D \$
 VRBL KSSRB 26D A -2 P 201.17460D \$
 VRBL KSS500MS 26D A -2 P 2031.7460D \$
 VRBL KSS500SEC 26D A -2 P 2031746.0D \$
 VRBL (SST1PTR,
 SST2PTR,
 VHSTW,
 VPSTG,
 VSCCA,
 VSC10,
 VSC11,
 VSC30,
 VSC31,
 VSNA,
 VSNA1,
 VSSCA,
 VSSFR,
 VSTG) I 26D U \$
 VRBL (VATRR,

VATR4,
 VGBIA,
 VSC12,
 VSC32,
 VSSRT,
 VSSTM,
 VSSW,

VSTGO) A 26D S -2 \$

TABLE VASPI V 1 1 \$
 FIELD S4CO B 0 25D \$
 FIELD SPEC B 0 24D \$
 FIELD CL3 B 0 23D \$
 FIELD CL1 B 0 22D \$
 FIELD T6C R 0 19D \$

END-TABLE VASPI \$

TABLE SSTTRPTR V 1 8D \$
 FIELD F1 I 26D U \$

END-TABLE SSTTRPTR \$

SSTTBPTR DATA CORAD(SSTTB1) \$
 DATA CORAD(SSTTR2) \$
 DATA CORAD(SSTTR3) \$
 DATA CORAD(SSTTR4) \$
 DATA CORAD(SSTTR5) \$
 DATA 0 \$
 DATA CORAD(SSTTB7) \$
 DATA CORAD(SSTTB8) \$

COMMENT SWITCH SELECTOR TABLE \$
 COMMENT \$
 COMMENT \$
 COMMENT THE SWITCH SELECTOR TABLE IS MADE UP OF A NUMBER OF \$
 COMMENT SMALLER TABLES, ONE FOR EACH TIME BASE AND FOR EACH \$
 COMMENT OF THE ALTERNATE SS SEQUENCES. HOWEVER, ONLY THE \$
 COMMENT TIME BASE 1 TABLE HAS BEEN CODED. \$
 COMMENT \$
 COMMENT EACH TABLE ENTRY REPRESENTS A SINGLE SS COMMAND AND \$
 COMMENT CONSISTS OF TWO WORDS. \$
 COMMENT 1. TIME OF ISSUANCE. \$
 COMMENT 2. SS STAGE AND ADDRESS. \$
 COMMENT \$

TABLE SSTABLE V 2 ZERO \$
 FIELD TIME A 26D S 10D 0 \$
 FIELD STAGADD I 15D U 1 25D \$

END-TABLE SSTABLE \$

TABLE SSTTB1 V 2 28D \$
 FIELD TIME A 26D S 10D 0 \$
 FIELD STAGADD I 15D U 1 25D \$

END-TABLE SSTTB1 \$

SSTTB1 DATA 5.0 \$ DATA 0 \$
 DATA 6.0 \$ DATA 10650 \$
 DATA 14.0D \$ DATA 02610 \$
 DATA 19.8D \$ DATA 40644 \$
 DATA 20.0D \$ DATA 40544 \$
 DATA 20.2D \$ DATA 40634 \$
 DATA 24.0D \$ DATA 02574 \$
 DATA 27.0D \$ DATA 40210 \$

| | | | | | |
|------|-----------|----|------|-------|----|
| DATA | 29.0D | \$ | DATA | 02774 | \$ |
| DATA | 30.0D | \$ | DATA | 40304 | \$ |
| DATA | 32.0D | \$ | DATA | 40110 | \$ |
| DATA | 49.5D | \$ | DATA | 02010 | \$ |
| DATA | 75.0D | \$ | DATA | 00000 | \$ |
| DATA | 90.0D | \$ | DATA | 40210 | \$ |
| DATA | 95.0D | \$ | DATA | 40110 | \$ |
| DATA | 95.3D | \$ | DATA | 02264 | \$ |
| DATA | 105.0D | \$ | DATA | 40764 | \$ |
| DATA | 115.1D | \$ | DATA | 02574 | \$ |
| DATA | 119.8D | \$ | DATA | 40674 | \$ |
| DATA | 120.0D | \$ | DATA | 40574 | \$ |
| DATA | 120.1D | \$ | DATA | 02774 | \$ |
| DATA | 130.0D | \$ | DATA | 40404 | \$ |
| DATA | 132.4D | \$ | DATA | 02164 | \$ |
| DATA | 133.6D | \$ | DATA | 40070 | \$ |
| DATA | 133.8D | \$ | DATA | 40170 | \$ |
| DATA | 134.4D | \$ | DATA | 02174 | \$ |
| DATA | 134.6D | \$ | DATA | 02374 | \$ |
| DATA | 7777.7774 | \$ | DATA | 00000 | \$ |

```

END-LOC-DD $
PROCEDURE MSS00 $
  GOTO SS DGSSM $
SS00.  'INHIBIT ALL INTERRUPTS EXCEPT TLC' $
  SET FASE TO 'ALTERNATE' $
  IF COMP DVASW(0,S4C0) THEN GOTO SS001 $
  SET BIT(0,7)(DVASW(0)),BIT(9D,17D)(DVASW(0)) TO 0 $
  IF VASPI(0,S4C0) THEN GOTO SS0060 $
  EGPO8 $ 'RESCHEDULE TIMER 1 (NOT CODED)' $
  SET VASPI(0) TO MSKSSS4C0 $
  SET SST1PTR TO CORAD(SSTSIVB) $
  GOTO SS1050 $
SS001. IF DVASW(0,SPEC)
  THEN SET BIT(0,7)(DVASW(0)),
  BIT(9D,17D)(DVASW(0)) TO 0
  THEN EGPO8 'RESCHEDULE TIMER 1 (NOT CODED)' $
  THEN SET VASPI(0) TO MSKSSSPEC
  THEN SET SST1PTR TO CORAD(SSTSIVA)
  THEN GOTO SS1050 $
  IF DVASW(0,TB6C)
  THEN SET DVASW(0,TB6C) TO 0
  THEN SET VASPI(0,T6C) TO 1
  THEN SET DVMC6(0,TB6B) TO 1
  THEN SET SST1PTR TO CORAD(SSTTB6C)
  THEN SSTUPD OUTPUT VATRR 'UPDATE SS TIME'
  THEN GOTO SS1050 $
  IF COMP(DVASW(0,TB6A) OR DVASW(0,TB6B) OR DVASW(0,S4C1))
  THEN GOTO SS002 $
  SET VSC10 TO SST1PTR $
  SET VSC11 TO VASPI(0) $
  SET VSC12 TO VATRR $
  SET VASPI(0) TO MSKSSCL1 $
  SSTUPD OUTPUT VATRR $ 'UPDATE SS TIME'
  SET FTADV TO 'NORMAL' $
  IF DVASW(0,TB6A)

```

```

        THEN SET DVASW(0,TB6A) TO 0
        THEN SET DVMC6(0,TB6A) TO 1
        THEN SET SST1PTR TO CORAD(SSTTB6A)
        THEN GOTO SS1050 $
    IF DVASW(0,S4C1)
        THEN SET DVASW(0,S4C1) TO 0
        THEN SET SST1PTR TO CORAD(SSTS4C1)
        THEN GOTO SS1050 $
    SET DVASW(0,TB6B) TO 0 $
    SET DVMC6(0,TB6B) TO 1 $
    SET SST1PTR TO CORAD(SSTTB6B) $
    GOTO SS1050 $
SS002.  IF FSSAC EQ 'ACTIVE'      THEN GOTO SS0060 $
        GOTO SS0000 $
MSS05.  SET FSSAC TO 'INACTIVE' $
SS0000. IF SSTABLE(0,0)SST1PTR EQ MSKSSNSSEND THEN GOTO SS0015$
SS0010. SSTUPD OUTPUT VSTGO $ 'UPDATE SS TIME'
        SET VSTGO TO VSSRT - VSTGO $
        IF VSTGO LT K99500MS THEN GOTO MSS30 $
        IF DFTUP EQ 'YES'
            THEN SET DVTGB TO DVTGB + VGBIA
            THEN SET VGBIA TO 0
            THEN SET DFTUP TO 'NO'
            THEN GOTO SS0010 $
        IF DVASW(0) NOT 0 THEN GOTO SS0170 $
        SET VSSTM TO VSTGO + DVTRB - DVTGB - KCSSK $
        SET DVSST TO VSSTM + DVTMR $
        SET DGSSM TO 'SS30' $
        IF COMP DFIL3 THEN EGPOB $ 'RESCHED T1(UNCODED)''
        GOTO SS0060 $
SS0015. IF DVASW(0) NOT 0 THEN GOTO SS0170 $
COMMENT READ REAL TIME CLOCK INTO TEMP $
        SET TEMP TO TEMP - DVRTC $
        IF TEMP LT 0 THEN SET TEMP TO TEMP + DKRTC0VF $
        SET DVSST TO DVTMM + K99500SEC + TEMP..0 $
        SET DGSSM TO 'SS05' $
        IF COMP DFIL3 THEN EGPOB $ 'RESCHED T1(NOT CODED)''
        GOTO SS0060 $
SS0170. IF DVASW(0,GNSS) OR DVASW(0,SBLO) OR DVASW(0,SBHI)
        OR DVASW(0,SBOM) OR DVASW(0,ECSV) OR DVASW(0,ECS1)
        OR DVASW(0,T3A) THEN GOTO SS004 $
        IF DVASW(0,ACQU)
            THEN SET DVASW(0,ACQU) TO 0
            THEN SET SST2PTR TO CORAD(SSTGAIN)
            THEN SSTUPD OUTPUT VATR4 'UPDATE SS TIME''
            THEN GOTO SS005 $
        IF DVASW(0,TB6D)
            THEN SET DVASW(0,TB6D) TO 0
            THEN SET SST2PTR TO CORAD(SSTTB6D)
            THEN SSTUPD OUTPUT VATR4 'UPDATE SS TIME''
            THEN GOTO SS005 $
        SET DVASW(0,LI) TO 0 $
        SET VATR4 TO 0 $
SS005.  SET FCLS4 TO 'INPROG' $ SET FHST TO 'TEST' $
        SET FTADV TO 'CLASS4' $

```

CMS-2 KERNEL 10 SWITCH SELECTOR PROCESSING

```

SS210 $ 'SET UP CLASS 4 ALTERNATE SEQUENCE'
GOTO SS0000 $
SS004. IF VASPI(0) NOT 0 THEN GOTO SS0060 $
SET VSC30 TO SST1PTR $
SET VSC31 TO VASPI(0) $
SET VSC32 TO VATRR $
SET VASPI(0) TO MSKSSCL3 $
SSTUPD OUTPUT VATRR $ 'UPDATE SS TIME'
SET FTADV TO 'NORMAL' $ SET FHST TO 'TEST' $
IF DVASW(0,GNSS)
    THEN SET DVASW(0,GNSS) TO 0
    THEN SET SST1PTR TO CORAD(SSTGSS)
    THEN GOTO SS0230 $
IF DVASW(0,SBLO)
    THEN SET DVASW(0,SBLO) TO 0
    THEN SET SST1PTR TO CORAD(SSTSRLO)
    THEN GOTO SS0230 $
IF DVASW(0,SRHI)
    THEN SET DVASW(0,SRHI) TO 0
    THEN SET SST1PTR TO CORAD(SSTSRHI)
    THEN GOTO SS0230 $
IF DVASW(0,SBOM)
    THEN SET DVASW(0,SBOM) TO 0
    THEN SET SST1PTR TO CORAD(SSTSBOM)
    THEN GOTO SS0230 $
IF DVASW(0,ECSV)
    THEN SET SST1PTR TO CORAD(SSTECSV)
    THEN GOTO SS0230 $
IF DVASW(0,ECS1)
    THEN SET SST1PTR TO CORAD(SSTECS1)
    THEN GOTO SS0230 $
SET DVASW(0,T3A) TO 0 $
SET SST1PTR TO CORAD(SSTTB3A) $
SS0230. SS210 $ 'SET UP SS TABLE'
GOTO SS0000 $
SS0060. IF FASE EQ 'ALTERNATE'
    THEN SET FASE TO 'NORMAL'
    THEN 'RELEASE PREVIOUSLY ENABLED INTERRUPTS'$
RETURN '$COMMON SS EXIT'
MSS10. SET VASPI(0) TO 0 $
SET VATRR TO 0 $
SET FCLS4 TO 'NOTINPROG' $
SET RIT(0,7)(DVASW(0)), RIT(9D,17D)(DVASW(0)) TO 0 $
EGP08 $ 'RESCHEDULE TIMER 1 (NOT CODED)'
SET FTADV TO 'NORMAL' $
SET SST1PTR TO SSTBPTR(OTRID - 1) $
SS1050. SS210 $ 'SET UP SS TABLE'
IF FSSAC EQ 'ACTIVE' THEN GOTO MSS20 $
SET VSSW TO KSSB1 $
SET FHST TO 'TEST' $
GOTO SS0000 $
MSS20. IF FSSIO THEN 'ISSUE FORCED RESET' $
SET FHST TO 'NOTEST' $
SSTUPD INPUT KSSB8, 'SS05' $ 'SCHEDULE SS CHECK'
SET VSSW TO KSSB5 $

```

```

MSS30.  GOTO SS0060 $
        SET FSSAC TO 'ACTIVE' $
        SET BIT(2)(VSNA1) TO BIT(0) VSNA $
        SET BIT(4,3)(VSNA1) TO BIT(2,3) VSNA $
        SET BIT(8,7)(VSNA1) TO BIT(6,7) VSNA $
        SET VSNA TO VSNA1 $
        IF VSNA EQ 0
            THEN SET FSSAC TO 'INACTIVE'
            THEN SS201 $ 'ADVANCE SS TABLE'
            THEN GOTO SS0000 $
        SET BIT(2)(VSTG) TO BIT(2)(VSNA) AND BIT(2)(VPSTG) $
        SET BIT(4)(VSTG) TO BIT(4)(VSNA) AND BIT(4)(VPSTG) $
        SET BIT(5)(VSTG) TO BIT(5)(VSNA) AND BIT(5)(VPSTG) $
        SET BIT(6)(VSTG) TO BIT(6)(VSNA) AND BIT(6)(VPSTG) $
        SET FSSIO TO VSTG NOT 0 $
        IF FHST EQ 'NOTEST' THEN GOTO SS4000 $
        IF DFLT EQ 'REP' THEN GOTO SS4000$
COMMENT  READ SS FEEDBACK REGISTER INTO TEMP $
        IF BIT(7,8)(TEMP) EQ 0 THEN GOTO SS4000 $
        IF FSSIO THEN 'ISSUE SS RESET'$
        SSTUPQ INPUT KSSB4, 'SS40' $ 'SCHEDULE STAGE/ADDRESS'
        SET VSSW TO KSSB5 $
        GOTO SS0060 $
MSS40.  'DELAY BEFORE ISSUING STAGE AND ADDRESS'$
SS4000. IF FSSIO THEN 'ISSUE STAGE AND ADDRESS FROM VSNA'$
        SSTUPQ INPUT VSSW, 'SS50' $ 'SCHEDULE ADDRESS VERIF'
COMMENT  DELAY FOR DOM TELEMETRY $
COMMENT  OUTPUT SS AND DO REGISTERS VIA DOM TELEMETRY $
        GOTO SS0060 $
MSS50.  SET VSCCA TO VSNA $
        SET VSSCA TO 0 $
SS006.  VARY I FROM 7 THRU 14D $
        SET BIT(I)(VSCCA), BIT(I)(VSSCA) TO
            COMP(BIT(I)(VSNA)) $
        END SS006 $
        IF VSTG NOT 0 THEN 'READ SS FEEDBACK REG INTO TEMP'
            THEN SET VSSFB TO 0
            THEN SET BIT(7,8D)(VSSFB) TO BIT(7,8D)(TEMP)
            GOTO SS007 $
        SET VSSFR TO VSSCA $
SS007.  IF VSSFB NOT VSSCA THEN GOTO SS5540 $
MSS55.  IF VASPI(0,S4C0)
            THEN SET DFILE(0,ISSA) TO 1
            THEN SET DVSST TO 1E10D
            THEN RETURN $ 'MSS50, MSS55'
        IF VSSRT EQ 0 THEN GOTO MSS60 $
        SSTUPD OUTPUT DVTRB $ 'UPDATE SS TIME'
        IF VSSRT - DVTRB LTEQ KSSRB THEN GOTO MSS60 $
        SET VSSTM TO VSSRT - DVTGB - KSSRB $
        SET DVSST TO VSSTM + DVTMR $
        SET DGSSM TO 'SS60' $
        IF COMP DFIL3 THEN EGPOB$ 'RESCHED TIMER 1(NOT CODED)'
        RETURN $ 'MSS50, MSS55'
SS5540. IF VSSFB EQ 0 AND BIT(7)(VSSCA) THEN GOTO MSS55 $
        IF FSSIO THEN 'ISSUE SS RESET' $

```

CMS-2 KERNEL 10 SWITCH SELECTOR PROCESSING

```

SSTUPQ INPUT KSSB6, 'SS80' $ 'SCHED COMP STAGE/ADD'
SET TEMP TO 0 $
SS008. VARY I FROM 7 THRU 14D $
        IF BIT(I)(VSSCA) NOT BIT(I)(VSSFB)
          THEN SET TEMP TO TEMP + 1 $
END SS008 $
IF TEMP LT 2 THEN RETURN $ 'MSS50, MSS55'
SET DVMC4(0,SSCB) TO 1 $
IF FFBCH EQ 'CHANA'
  THEN SET FFBCH TO 'CHANB'
  THEN 'SET SS CHAN B BIT IN INTERNAL CONTROL REG'
  THEN SET DVICR(0,SSCB) TO 1 $
UTR30 $ 'DELAY FOR TELEMETRY AS REQUIRED'
COMMENT TELEMETER SS FEED BACK, VSSFB $
MSS60. RETURN $ 'MSS50, MSS55'
SET TEMP TO VSTG $
SET BIT(0)(TEMP) TO 1 $
IF FSSIO THEN 'ISSUE READ COMMAND FROM TEMP' $
COMMENT READ CLOCK INTO TEMP $
SSTUPQ INPUT KSSB2, 'SS70' $ 'SCHEDULE READ RESET'
SET TEMP TO 0 + BIT(13D,13D)(TEMP) $
SET BIT(0,13D)(TEMP) TO BIT(2,13)(VSNA) $
UTR30 $ 'DELAY FOR TELEMETRY AS REQUIRED'
COMMENT TELEMETER STAGE/ADDRESS AND READ TIME, TEMP $
IF DFACQ NOT 'GAIN' 'COMPRESS DATA BETWEEN STATIONS'
  THEN MPC80 INPUT DVDCT + MSKSSDCT 'COMP TIME/TAG'
  THEN MPC80 INPUT MSKSSDCS + BIT(0,23)(VSNA) $
  'COMPRESS STAGE AND ADDRESS WITH TAG'
IF VASPI(0,S4C0)
  THEN SET VASPI(0) TO 0
  THEN SET DFILE(0,CORD) TO 1 $
IF VSNA1 EQ MSKSSHIG
  THEN SET DVMC7(0,HIG) TO 1
  THEN RETURN $ 'MSS60'
IF VSNA1 EQ MSKSSLOG
  THEN SET DVMC7(0,LOG) TO 1
  THEN RETURN $ 'MSS60'
IF VSNA1 EQ MSKSSOMG
  THEN SET DVMC7(0,OMG) TO 1
  THEN RETURN $ 'MSS60'
IF VSNA1 NOT MSKSSIVB THEN RETURN $ 'MSS60'
IF FBRI EQ 'FIRST'
  THEN DVMC5(0,S4R1I) TO 1
  THEN RETURN $ 'MSS60'
SET DVMC6(0,S8BRI) TO 1 $
RETURN $ 'MSS60'
MSS70. IF FSSIO THEN 'RESET READ COMMAND' $
SSTUPQ INPUT KSSB3, 'SS05' $ 'SCHEDULE HUNG STAGE TEST'
SS201 $ 'ADVANCE SS TABLE'
SET VSSW TO KSSB1 $
SET FHST TO 'TEST' $
IF BIT(2,5)(VHSTW) EQ BIT(2,5)(VSTG)
  THEN SET FHST TO 'NOTEST' $
IF VSNA1 EQ MSKSSWVO
  THEN SET DVASW(0,ECS1) TO 0

```

CMS-2 KERNEL 10 SWITCH SELECTOR PROCESSING

```

        THEN SET DFWV TO 'OPEN'
        THEN RETURN $ 'MSS70'
    IF VSNA1 EQ MSKSSWVC
        THEN SET DVASW(0,ECSV) TO 0
        THEN SET DFWV TO 'CLOSE'
        THEN RETURN $ 'MSS70'
    IF VSNA1 EQ MSKSSSCC
        THEN SET DVDPM(0,DIN9) TO 1 $
    RETURN $ 'MSS70'
MSS80.  SET VSNA TO VSCCA $
        IF FSSIO THEN 'ISSUE STAGE AND COMPLEMENTED ADDRESS'S
        SSTUPO INPUT K9B7, 'SS55'S 'SCHEDULE READ COMMAND'
COMMENT DELAY FOR DOM TELEMETRY $
COMMENT OUTPUT SS AND DO REGS VIA DOM TELEMETRY $
        RETURN $ 'MSS80'
END-PROC MSS00 $
PROCEDURE SS201 $ 'SS TABLE ADVANCE ROUTINE'
    IF FTADV EQ 'NORMAL'
        THEN SET SST1PTR TO SST1PTR + 1
        THEN GOTO SS009 $
    SET SST2PTR TO SST2PTR + 1 $
SS009.  SS210 $ 'SET UP NEXT SWITCH SELECTOR'
        RETURN $ 'SS201'
END-PROC SS201 $
PROCEDURE SS210 $ 'SS SELECTION AND SETUP ROUTINE'
    IF FTADV EQ 'NORMAL' THEN GOTO SS2020 $
SS2160. IF SSTABLE(0,0)SST2PTR GTEQ 0 THEN GOTO SS2070S
        SET FCLS4 TO 'NOTINPROG' $
        SET DVMC6(0,LUI) TO 0 $
        SET DVMC7(0,T6D) TO 0 $
        GOTO SS2090 $
SS2020. IF SSTABLE(0,0)SST1PTR GTEQ 0 THEN GOTO SS2030 $
        IF VASPI(0,SPEC)
            THEN SET VASPI(0) TO MSKSSS4C0
            THEN SET BIT(0,7)(DVASW(0)),BIT(9D,17D)(DVASW(0))
                TO 0
            THEN SET SST1PTR TO CORAD(SSTSIVB)
            THEN GOTO SS2020 $
        IF VASPI(0,CL3)
            THEN SET SST1PTR TO VSC30
            THEN SET VASPI(0) TO VSC31
            THEN SET VATRR TO VSC32
            THEN GOTO SS2020 $
        IF VASPI(0,CL1)
            THEN SET SST1PTR TO VSC10
            THEN SET VASPI(0) TO VSC11
            THEN SET VATRR TO VSC12
            THEN GOTO SS2020 $
        SET VASPI(0), VATRR TO 0 $
        IF FT60P EQ 'PASS1'
            THEN SET FT60P TO 'PASS2'
            THEN SET SST1PTR TO CORAD(SSTTB5A)
            THEN GOTO SS2020 $
        SET SST1PTR TO CORAD(SSTTB5B) $
        GOTO SS2020 $

```

```

SS2030. IF FCLS4 EQ 'NOTINPROG' THEN GOTO SS2040 $
SS2070. IF SSTABLE(0,0)SST1PTR * DKRTCSEC + VATRR = KSS500MS
      GTEQ SSTABLE(0,0)SST2PTR * DKRTCSEC + VATR4
      THEN SET FTADV TO 'CLASS4'
      THEN SET VSSRT TO SSTABLE(0,0)SST2PTR * DKRTCSEC
        + VATR4
      THEN SET VSNA TO SSTABLE(0,1)SST2PTR
      THEN GOTO SS2050 $
SS2090. SET FTADV TO 'NORMAL' $
SS2040. SET VSSRT TO SSTABLE(0,0)SST1PTR * DKRTCSEC + VATRR $
      SET VSNA TO SSTABLE(0,1)SST1PTR $
SS2050. SET BIT(2,5)(VHSTW) TO BIT(0,5)(VSNA) $
      RETURN $ 'SS210'
END-PROC SS210 $
PROCEDURE SSTUPD OUTPUT TIME $ 'SS TIME UPDATE ROUTINE'
COMMENT READ CLOCK INTO TEMP $
      SET TEMP TO TEMP - DVRTC $
      IF TEMP LT 0 THEN SET TEMP TO TEMP + DKRTC0VF $
      SET TIME, DVTRB TO DVTGB + DVTRR + TEMP..0 $
      RETURN $ 'SSTUPD'
END-PROC SSTUPD $
PROCEDURE SSTUPQ INPUT BIAS, ID $ 'SS SCHEDULER'
COMMENT READ CLOCK INTO TEMP $
      SET DGSSM TO 'ID' $
      SET TEMP TO TEMP - DVRTC $
      IF TEMP LT 0 THEN SET TEMP TO TEMP + DKRTC0VF $
      SET DVTRB TO DVTGB + DVTRR + TEMP..0 $
      SET VSSTM TO BIAS + DVTRR + TEMP..0 $
      SET DVSST TO VSSTM + DVTMR
      IF COMP DFIL3 THEN EOP08 $ 'RESCHEDULE T1(NOT CODED)'
      RETURN $ 'SSTUPQ'
END-PROC SSTUPQ $
END-SYS-PROC MSS00 $

```


CMS-2 KERNEL 11 ATM TASK KEYING

```

TASKKEY  SYS-PROC  'ATM TASK KEYING ROUTINE' $
LOC-DD  $
VRBL PRIORITY  I  10D S $ 'TASK PRIORITY LEVEL'
VRBL TSKPTR    I  16D U $ 'TASK POINTER (ADDRESS)'
VRBL I         I  10D S $ 'OVERFLOW TABLE CHAIN INDEX'
VRBL J         I  10D S $ 'OVERFLOW TABLE INDEX'

COMMENT
COMMENT  THE PRIORITY CONTROL TABLE CONTAINS ONE ENTRY FOR $
COMMENT  EACH PRIORITY LEVEL.  EACH ENTRY CONSISTS OF FIVE $
COMMENT  ITEMS. $
COMMENT    1. LOCATION POINTER TO THE NEXT EXECUTABLE $
COMMENT    INSTRUCTION OF THE TASK CURRENTLY ASSIGNED $
COMMENT    TO THAT PRIORITY LEVEL OR ZERO IF NO TASK $
COMMENT    IS CURRENTLY ASSIGNED. $
COMMENT    2. TASK REGISTER CONTENTS (INITIALLY ZERO). $
COMMENT    3. TASK REGISTER CONTENTS (INITIALLY ZERO). $
COMMENT    4. TASK REGISTER CONTENTS (INITIALLY ZERO). $
COMMENT    5. INDEX POINTER TO THE BEGINNING OF THE $
COMMENT    PRIORITY OVERFLOW TABLE CHAIN FOR THAT $
COMMENT    PRIORITY LEVEL.  A VALUE OF ZERO INDICATES $
COMMENT    END OF CHAIN. $
TABLE ATMPCT V NONE 10D $
FIELD ATMTSKPTR I 16D U $
FIELD ATMTSKREG1 I 26D U $
FIELD ATMTSKREG2 I 26D U $
FIELD ATMTSKREG3 I 26D U $
FIELD ATMVFPTTR I 16D U $
END-TABLE ATMPCT $

COMMENT
COMMENT  THE PRIORITY OVERFLOW TABLE IS USED FOR KEYING $
COMMENT  TASKS ON A PRIORITY LEVEL WHICH IS CURRENTLY $
COMMENT  ASSIGNED TO ANOTHER TASK.  THE ENTRIES ARE NOT $
COMMENT  ALLOCATED TO A FIXED PRIORITY BUT ARE ASSIGNED $
COMMENT  DYNAMICALLY AS REQUIRED.  ALL OVERFLOW ENTRIES FOR $
COMMENT  EACH PRIORITY LEVEL ARE CHAINED TOGETHER SUCH THAT $
COMMENT  THE TASKS CAN BE EXECUTED ON A FIRST-IN/FIRST-OUT $
COMMENT  BASIS.  EACH ENTRY CONSISTS OF TWO ITEMS. $
COMMENT    1. INDEX POINTER TO THE NEXT ENTRY IN THE $
COMMENT    CHAIN.  A VALUE OF ZERO INDICATES END OF $
COMMENT    CHAIN. $
COMMENT    2. POINTER TO THE BEGINNING OF THE TASK FOR $
COMMENT    THAT ENTRY.  A VALUE OF ZERO INDICATES THATS $
COMMENT    THE ENTRY IS NOT ASSIGNED TO ANY TASK. $
TABLE ATMPOVFT V NONE 26D $
FIELD ATMVFPTTR I 16D U $
FIELD ATMTSKPTR I 16D U $
END-TABLE ATMPOVFT $
END-LOC-DD $
PROCEDURE TASKKEY INPUT PRIORITY, TSKPTR $
COMMENT
COMMENT  INHIBIT ALL INTERRUPTS $
COMMENT
COMMENT  IF THE REQUESTED PRIORITY LEVEL IS NOT CURRENTLY $
COMMENT  ASSIGNED, INITIALIZE THE ENTRY FOR THIS TASK. $
COMMENT

```

CMS-2 KERNEL 11 ATM TASK KEYING

```

IF ATPCT(PRIORITY,ATMTSKPTR) EQ 0
    THEN SET ATPCT(PRIORITY,ATMTSKPTR) TO TSKPTR
    THEN SET ATPCT(PRIORITY,ATMTSKREG1),
             ATPCT(PRIORITY,ATMTSKREG2),
             ATPCT(PRIORITY,ATMTSKREG3) TO 0
    THEN GOTO FINI $
COMMENT
COMMENT OTHERWISE, SEARCH FOR THE END OF THE OVERFLOW
COMMENT POINTER CHAIN.
COMMENT
SET I TO ATPCT(PRIORITY,ATMOVFPTR) $
IF I EQ 0 THEN GOTO SLTSRCH $
CHNSRCH. IF ATPOVFT(I,ATMOVFPTR) NOT 0
          THEN SET I TO ATPOVFT(I,ATMOVFPTR)
          THEN GOTO CHNSRCH $
COMMENT
COMMENT WHEN THE END OF THE OVERFLOW POINTER CHAIN HAS BEEN
COMMENT FOUND, SEARCH FOR AN EMPTY SLOT IN THE OVERFLOW
COMMENT TABLE.
COMMENT
SLTSRCH. VARY J FROM 1 THRU 25D $
          IF ATPOVFT(J,ATMTSKPTR) EQ 0 THEN GOTO SLTFNDS
          END SLTSRCH $
          STOP $ 'HALT IF OVERFLOW TABLE IS FULL'
COMMENT
COMMENT ADD THE NEW ENTRY TO THE END OF THE OVERFLOW CHAIN
COMMENT AND STORE THE TASK POINTER IN IT.
COMMENT
SLTFND. IF I EQ 0
          THEN SET ATPCT(PRIORITY,ATMOVFPTR) TO J
          THEN GOTO AROUND $
          SET ATPOVFT(I,ATMOVFPTR) TO J $
AROUND. SET ATPOVFT(J,ATMOVFPTR) TO 0 $
          SET ATPOVFT(J,ATMTSKPTR) TO TSKPTR $
FINI.   'RELEASE INTERRUPTS AS REQUIRED'$
        RETURN $
END-PROC TASKKEY $
END-SYS-PROC TASKKEY $

```