

1. Report No. 953-6		2. Government Accession No.		3. Recipient's Catalog No.	
4. Title and Subtitle Actuator Program For Frontage Road Control			5. Report Date October 1974		
			6. Performing Organization Code		
7. Author(s) Byron White			8. Performing Organization Report No. 953-6		
9. Performing Organization Name and Address Texas Transportation Institute College Station, Texas 77843			10. Work Unit No.		
			11. Contract or Grant No.		
12. Sponsoring Agency Name and Address Federal Highway Administration Department of Transportation Washington, D. C. 20590			13. Type of Report and Period Covered		
			14. Sponsoring Agency Code		
15. Supplementary Notes Research Study Title: Dallas Corridor Study					
16. Abstract A computer program that will operate in minicomputers located at diamond interchanges along the frontage road of North Central Expressway in Dallas is described. The minicomputers, referred to in this report as actuators, will be part of a larger computer control system that will coordinate the operation of the frontage road signals. This report describes the actuator program both functionally (i.e., for the traffic engineer's information) and technically (i.e., for the computer specialist's information). The program logic, which is shown in detailed flow charts, may be useful in other computer controlled traffic applications.					
17. Key Words Actuator, minicomputer, program telemetry, task, frontage road system, diamond interchange, emergency controller			18. Distribution Statement		
19. Security Classif. (of this report) None		20. Security Classif. (of this page) None		21. No. of Pages	22. Price

ACTUATOR PROGRAM
FOR FRONTAGE ROAD CONTROL

Prepared by

Byron White
Systems Analyst

Texas Transportation Institute
Texas A&M University
College Station, Texas

October 1974

Prepared for
Federal Highway Administration
Department of Transportation
Contract No. DOT-11-7964

TABLE OF CONTENTS

	<u>PAGE NO.</u>
1.0 INTRODUCTION	1
1.1 Background	1
1.2 Scope of Report	1
1.3 Structure of Report	3
2.0 FUNCTIONAL DESCRIPTION OF ACTUATOR PROGRAM	4
2.1 Initialization and Restart	4
2.2 Task Control	4
2.3 Telemetry Control	4
2.4 Detector Processing	7
2.5 Signal Light Processing	9
2.6 Emergency Control	16
3.0 TECHNICAL DESCRIPTION OF ACTUATOR PROGRAM	21
3.1 Actuator Computer Hardware Description	21
3.2 Programming Problems, Constraints and Notes	23
3.3 Program Segments 0 and 8, Common Definitions	25
3.4 Program Segment 1, Common Subroutines	25
3.5 Program Segment 2, Initialization, Interrupt, Task Control	25
3.6 Program Segment 3, Magnetometer Processing (TASK0)	41
3.7 Program Segment 4, Detector Processing (TASK1)	42
3.8 Program Segment 5, Signal Light Processing (TASK2)	44
3.9 Program Segment 6, Emergency Control (TASK3)	54
3.10 Program Segment 7, IDLE TASK	57
4.0 PROGRAM TESTING	60
4.1 Program Configuration	60
4.2 Program Checkout Variables	61
4.3 Checkout Procedure	62
5.0 PROGRAM LOADING AND OPERATION	68
5.1 Program Preparation	68
5.2 Loading the Actuator	68
5.3 Pin Matrix Operation	70
5.4 Clearing a "Stall Alarm" Condition	73
6.0 REFERENCES	75
APPENDIX A - ACTUATOR PROGRAM FLOW CHARTS	
APPENDIX B - ACTUATOR PROGRAM "COMMON" DEFINITIONS	
APPENDIX C - INPUT/OUTPUT BIT ASSIGNMENTS	
APPENDIX D - TELEMETRY FORMAT BIT ASSIGNMENTS	
APPENDIX E - ACTUATOR PROGRAM SEGMENTS	
APPENDIX F - ACTUATOR PROGRAM LISTINGS	
APPENDIX G - SYSTEM CONFIGURATION PARAMETERS	

ABSTRACT

A computer program that will operate in minicomputers located at diamond interchanges along the frontage road of North Central Expressway in Dallas is described. The minicomputers, referred to in this report as actuators, will be part of a larger computer control system that will coordinate the operation of the frontage road signals. This report describes the actuator program both functionally (i.e., for the traffic engineer's information) and technically (i.e., for the computer specialist's information). The program logic, which is shown in detailed flow charts, may be useful in other computer controlled traffic applications.

DISCLAIMER

The contents of this report reflect the views of the author who is responsible for the facts and the accuracy of the data presented herein. The contents do not necessarily reflect the official views or policies of the Federal Highway Administration. This report does not constitute a standard, specification or regulation.

1.0 INTRODUCTION

1.1 Background

"Actuator" is the term applied to the minicomputers located in control cabinets at each frontage road intersection. Each actuator is a general purpose programmable computer known as a Nova 1210 manufactured by Data General Corporation. The actuators have three broad purposes:

1. To communicate the status of the intersection hardware to the network control computers.
2. To control the traffic signals as commanded by the network control computers.
3. To serve as a back-up controller in the event of a communications failure with the network control computers.

The use of an actuator at an intersection was described by Charles W. Blumentritt in Reference 6. Many of the concepts used in the current actuator program were derived from that report.

1.2 Scope of Report

This report describes the computer program for the frontage road actuators. No attempt is made to cover other parts of the frontage road system. Figure 1.1 shows the relationship of the actuators to other frontage road control system components. Reference 3 describes each element of the control system in detail.

The actuator program was constructed in a modular fashion that will permit its use in the future arterial intersection actuators with very minor changes. These changes will be to various tables that define detector inputs, signal light outputs, and emergency control logic.

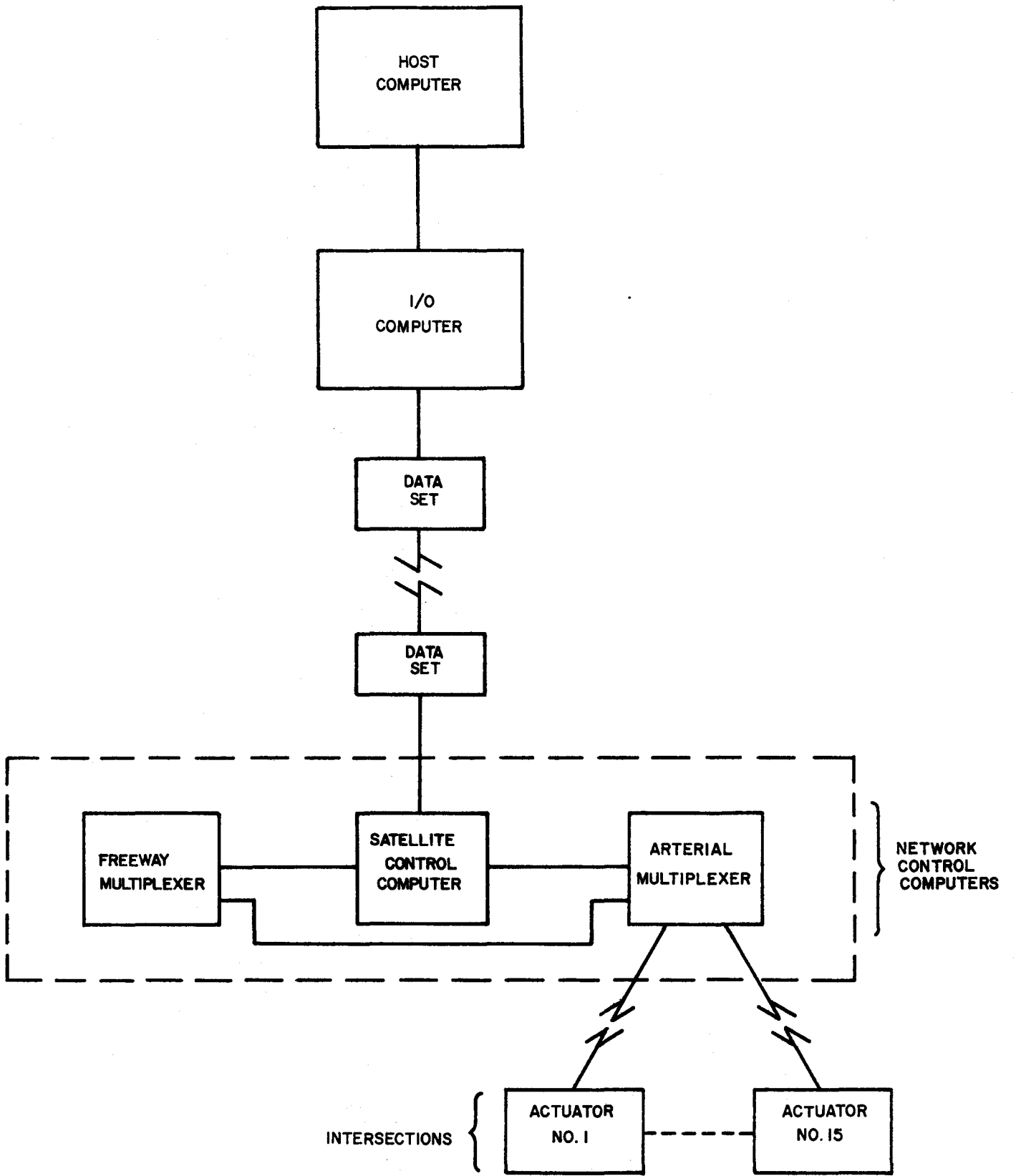


FIGURE 1.1
FRONTAGE ROAD CONTROL SYSTEM

1.3 Structure of Report

Section 2 is a functional description of the actuator program. That is, it tells what the program does but does not go into detail about how each function is accomplished. It is intended for all readers of this report.

Section 3 is a technical description of the actuator program. It is intended for the computer systems analyst or programmer who is interested in the details of how each specific task was accomplished.

Section 4 describes the testing procedure that was used to verify the various program functions.

Section 5 is a detailed description of the program loading and operating procedure for the actuator once it is installed in the field.

2.0 FUNCTIONAL DESCRIPTION OF ACTUATOR PROGRAM

Figure 2.1 is a block diagram of the inputs and outputs of the actuator. It is the function of the actuator program to react in a timely manner to all the inputs such that the proper outputs are produced. The following paragraphs describe the various subfunctions of the actuator program.

2.1 Initialization and Restart

When the program is first started or when power is restored after an electrical power failure, the program must be initialized. This consists of placing the intersection on flash, synchronizing the communications with the arterial multiplexer, setting program variables to their initial values, and initiating the various program tasks.

2.2 Task Control

The actuator program consists of several tasks, each with different priorities and execution rates. For example, speed detector processing is a high priority task that is executed every .01 second, while stop-line detector processing executes at a lower priority and less often. It is the job of task control to schedule each task for execution at the proper time and to oversee the execution of all tasks. It is the "operating system" for the actuator. The task controller is also responsible for resetting the stall alarm, a device that causes the intersection traffic lights to flash if it is not periodically pulsed by the actuator.

2.3 Telemetry Control

The telemetry control section of the actuator program is responsible for all communication with the arterial multiplexer computer.

Communication is over a voice-grade telephone line with a data set, consisting of a transmitter and receiver, at each end. Transmission and reception

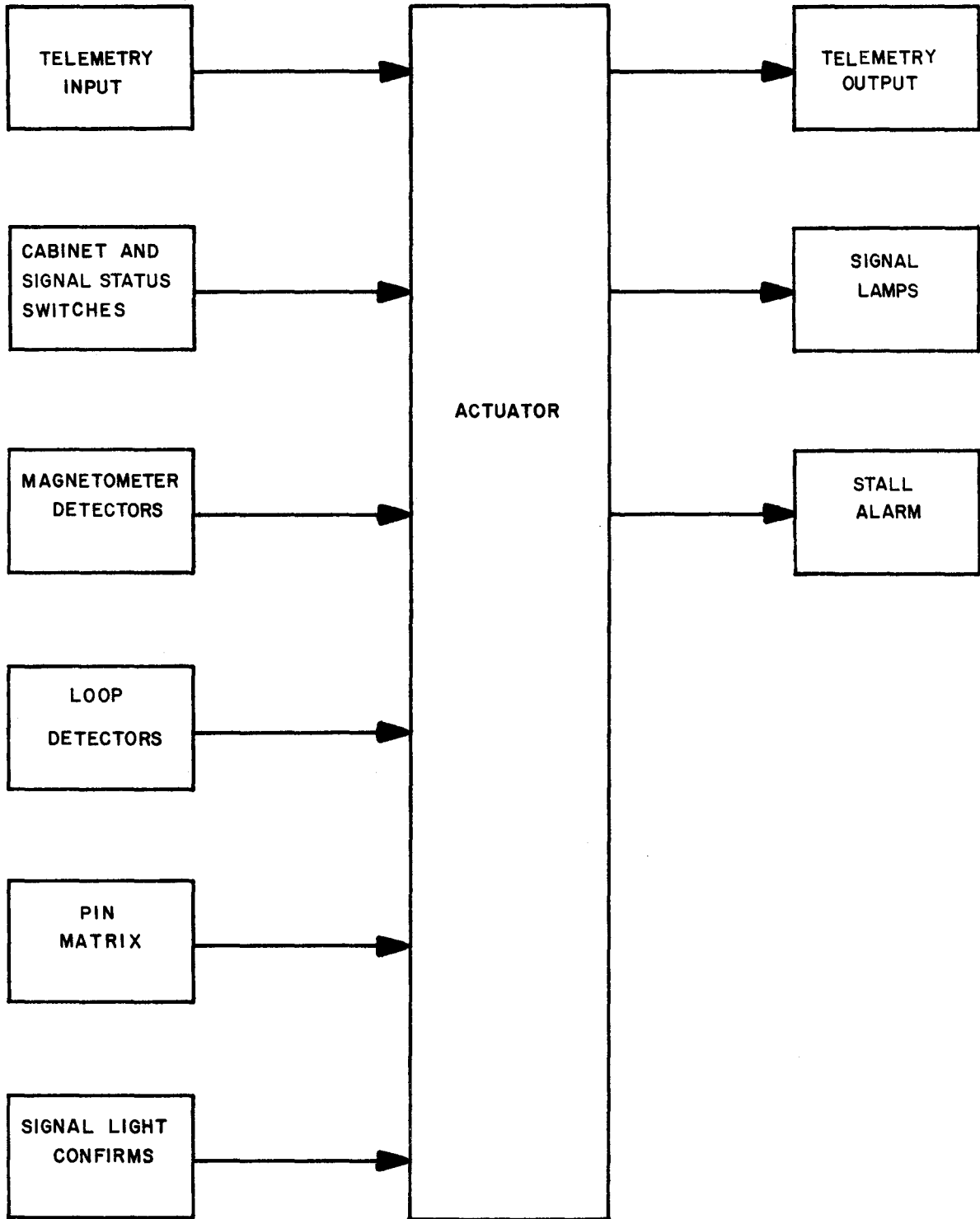


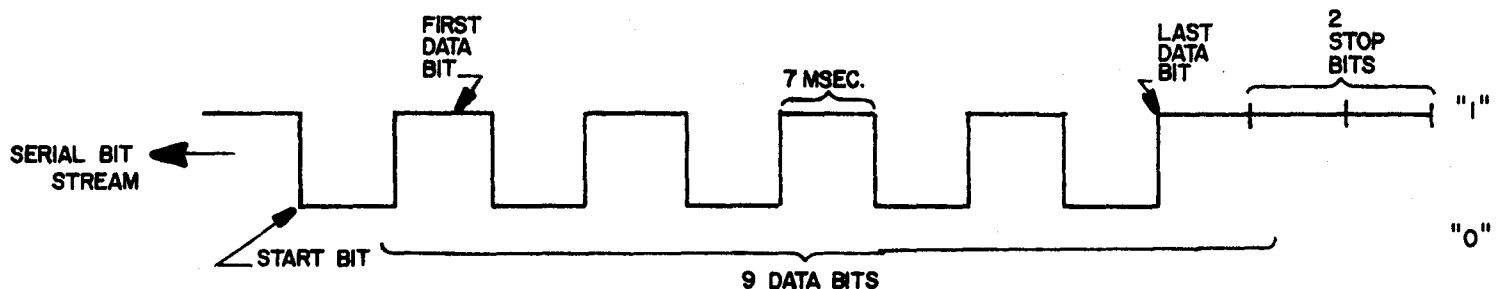
FIGURE 2.1

ACTUATOR INPUT/OUTPUT -GENERAL

by the actuator can be simultaneous, since its transmitter and receiver operate at different frequencies.

All data is transmitted and received in a serial fashion. The timing and packing/unpacking of each data bit is done under actuator program control. This allows for complete flexibility in the telemetry message format, size and speed.

The telemetry messages are sent and received in an asynchronous manner. Therefore, there must be a way for the receiving computer to distinguish the start of a message. This is done in the actuator by sending two consecutive stop bits ("1" bits) at the end of the message, and a start bit ("0" bit) at the start of the next message. A typical message might be diagrammed thusly:



Each message is 12 bits in length. Each bit is sent in 7 milliseconds (.007 second) for a message transfer time of 84 milliseconds, or about 12 messages per second.

The actuator receives telemetry data from the arterial multiplexer continuously. The input data is generally a command directing the actuator to perform certain functions. The most commonly sent command directs the actuator to place the traffic signals in one of the permitted configurations. Appendix D shows the meaning of all input bits.

The actuator sends telemetry data to the arterial multiplexer continuously. This output data consists of intersection status information, such as traffic

signal status, detector actuations, etc. Appendix D shows the meaning of all output bits.

The actuator checks each telemetry input bit for transmission errors, such as telephone line noise, and informs the network control computers if any are encountered. Also, each input message has a check or parity bit that should make the total number of one bits in the message an odd number. If the number of one bits is even, the entire message is ignored. For security purposes, the actuator will not act upon a command from the network control computers until two identical commands are received consecutively.

2.4 Detector Processing

As Figure 2.2 shows, there are three types of detectors used in the front-age road system. The method of processing each type is given in the following paragraphs.

2.4.1 Magnetometers

These detectors are located about 300 feet back from the intersection stop-line in each approach lane. Each time a vehicle activates one of these detectors, a volume bit is set in the telemetry output word for that detector. The time that the detector is occupied by each vehicle is used to update an average speed for each approach to the intersection. This new average speed is also sent via telemetry, but only if it is different from the previously sent average speed for that approach.

2.4.2 Loops

These detectors are located at the stop-lines for each approach to the intersection and in left-turn bays if present. A volume bit is set in the telemetry output word whenever a vehicle enters the loop. An occupancy counter is incremented every .05 seconds that the vehicle is in the loop. Whenever this

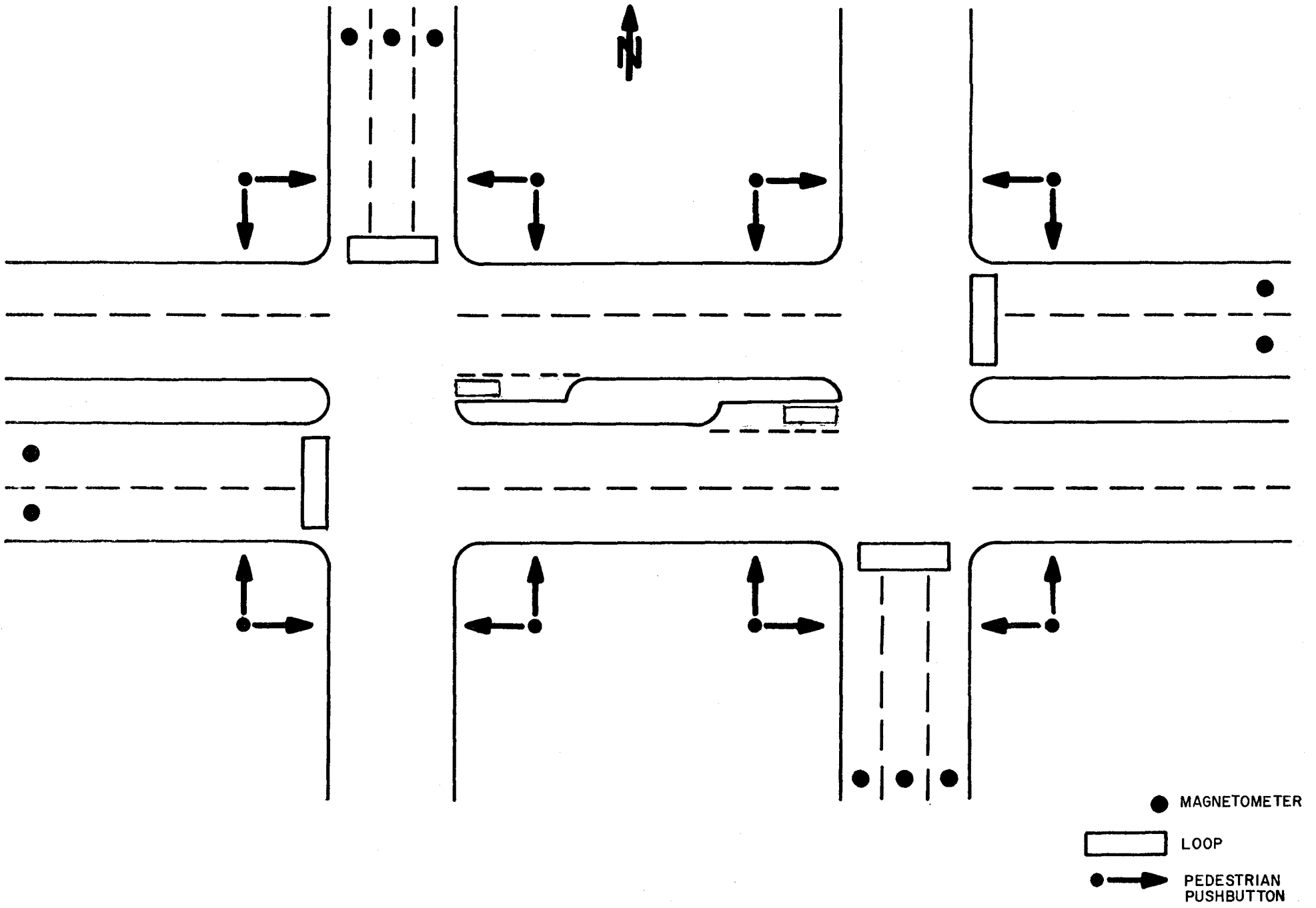


FIGURE 2.2

TYPICAL FRONTAGE ROAD INTERSECTION DETECTOR LOCATIONS

occupancy counter overflows (currently every 5 seconds), an occupancy bit is set for telemetry output.

2.4.3 Pedestrian Pushbuttons

Pedestrian pushbuttons are located on each corner of the frontage road intersection. Whenever a pedestrian activates the pushbutton, a volume bit is set in the telemetry output.

2.5 Signal Light Processing

This is the most critical task of the actuator program, since it directly affects the safety of both vehicular and pedestrian traffic. For this reason, much effort was expended in making this portion of the program extremely flexible and self-checking.

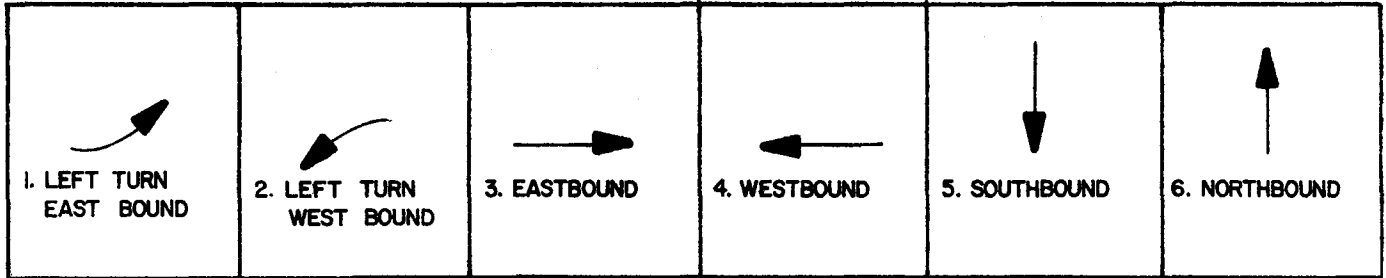
2.5.1 Signal Hardware

The signal hardware consists of:

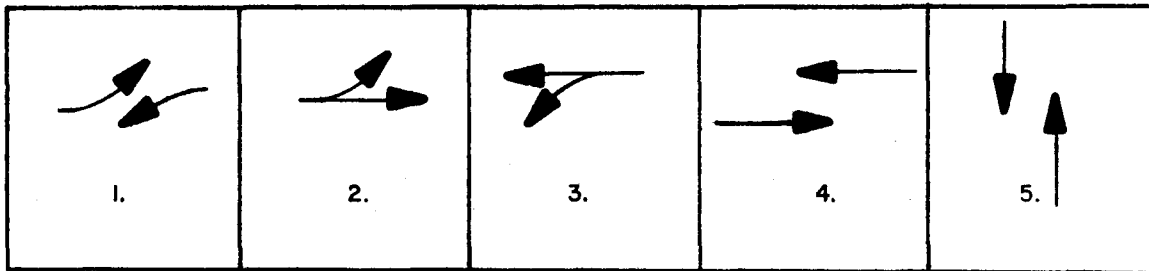
1. Red, green and amber lights.
2. Walk and don't walk lights.
3. Solid state switches that permit the actuator to switch AC power to the traffic lights.
4. Flash circuitry that allows the actuator to place the signals in flash mode.
5. Sensor circuits that allow the actuator to sense the on/off status of the switches that control the traffic lights.

2.5.2 "Phase" Versus "Movement Approach"

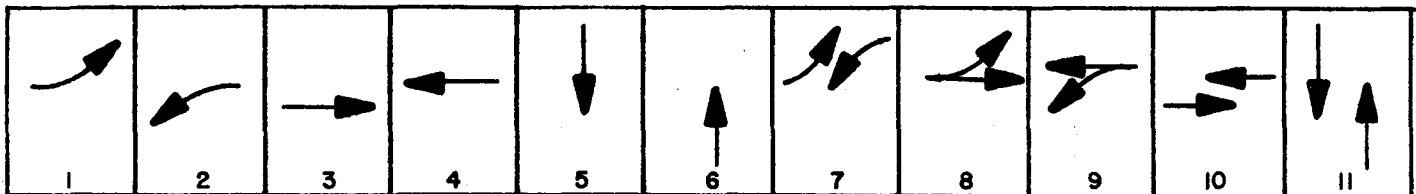
There are two generally used methods for describing the configuration of the traffic signals at an intersection. Assume a simple intersection that has traffic signals for the following traffic movements:



In the "phase" method, the intersection might be described as 5-phase because the following five signal configurations are allowed:







In the "movement" method, the intersection would be described as 6-movement since signals exist for six separate traffic movements as shown above. However, the following configurations would be allowed:



That is, the five valid phases as well as each separate movement alone would be allowed.





The network control computers must communicate to the actuator the desired state of the traffic signals. If the "phase" method were used, the commands might appear as follows:

<u>Time (Sec.)</u>	<u>Command</u>	<u>Desired Configuration</u>
0	Phase 1	
1	Phase 1	
-	Phase 1	
10	Phase 4	Ambers
-	Phase 4	Ambers
14	Phase 4	

The rules for the actuator to use under this "phase" method would be:

1. Remain in commanded phase as long as that phase number is being received.
2. When a different phase number is received, display the appropriate ambers for a preset time. Then display the greens for the new phase. Ignore new commands during amber display, since each phase must be retained for a minimum time period.

If the "movement" method were used the commands would be:

<u>Time (Sec.)</u>	<u>Command</u>	<u>Desired Configuration</u>
0	Movements 1 and 2	
1	Movements 1 and 2	
-	Movements 1 and 2	
10	Movements 3 and 4	Ambers
-	Movements 3 and 4	Ambers
14	Movements 3 and 4	

The rules for the actuator to use under this "movement" method would be:

1. Display the green movements given in the command unless they conflict.

2. When a different command is received, display ambers for the movements that existed in the old command but are not in the new command. Display green movements in the new command unless they conflict with the ambers.

The "movement" approach is by far the more flexible of the two methods, since it allows each possible green movement to be displayed separately or in conjunction with other valid green movements. The following example in Table 2.1 illustrates this flexibility. In the example, the movement method allowed the main street greens to be displayed 3 seconds earlier than the phase method. This is due to the fact that the phase method does not allow each movement to be controlled individually, and thus the amber required to go from phase 1 to phase 3 had to "time-out" before a new command could be accepted.

The "movement" method was therefore used in the actuator for the following reasons:



















1. Each movement can be controlled independently of all others.
2. Each movement may have its own times - minimum, maximum, extension, amber.
3. The intersection may be allowed to "rest" in all red. Then only a movement with demand may be displayed. This feature might be used during periods of light traffic.

2.5.3 Signal Light Confirm Checking

Hardware to allow the actuator to sense the on/off state of the solid state switches that control the traffic signals is a standard part of each intersection. This allows the actuator program to determine if the signal

TABLE 2.1

EXAMPLE: At $t = 0$, movements 1 and 2 are displayed. When demand ends on movement 1, it is desired to begin movement 4. When demand ends on movement 2, it is desired to begin movement 3. Assume 4 second ambers.

<u>Time (Sec.)</u>	<u>Situation</u>	<u>Movement Method</u>	<u>Phase Method</u>
0	Demand on movements 1 & 2	Command movements 1 & 2 	Command phase 1 
1	Demand ends on movement 1	Command movement 2 AMBER  	Command phase 3 AMBER  
2	Demand ends on movement 2	Command no movements AMBER  	Amber in progress Cannot accept new command
5	-	Command movement 4  	Command phase 4  
6	-	Command movements 3 & 4  	Amber in progress
9	-	Command movements 3 & 4  	Command phase 4  

lights are operating as commanded. The state of all signal lights is sampled every 0.1 second and the following action is taken if any error persists for two consecutive samples:

1. If any green is on when it should be off, put intersection on flash.
2. If any green is off when it should be on, send error indication via telemetry.
3. If any red is on when it should be off, send error indication via telemetry.
4. If any red is off when it should be on, put intersection on flash.
5. If any amber is on when it should be off, put intersection on flash.
6. If any amber is off when it should be on, send error indication via telemetry.

That is, any error that might lead a driver to believe he has the right-of-way when he doesn't will cause the intersection to flash.

2.5.4 Command Decoding and Changing Signals

The portion of the actuator program responsible for actually controlling the signal lights can accept commands from two sources. Under normal circumstances, commands are received from the telemetry input stream. These command codes are decoded according to the chart in Appendix D. However, under abnormal circumstances such as when communications via the telemetry link are interrupted or after an electrical power failure, signal commands are accepted from the emergency control section of the actuator program.

In either case, the commands are checked for validity. Any attempt to terminate a green movement before an absolute minimum time expires is invalid. Any attempt to turn on conflicting greens is of course invalid. After the command is checked, the signals are changed according to the rules given for the "movement" method described in 2.5.2. That is, ambers are displayed for a set time for each terminating green movement, and new greens are displayed as soon as possible without causing conflicts. Whenever a new green is displayed, the pedestrian WALK light controlled by that green is displayed for a set time.

Before any change is made to the signal lights, the following checks are made on the new outputs:

1. Will the outputs cause a valid (i.e., nonconflicting) green configuration to be displayed?
2. Will the outputs light one and only one color per traffic movement?

If either test is negative, the actuator program performs an abort, which causes the intersection signals to flash.

2.5.5 Other Functions of Signal Control

The signal control section of the actuator program is also responsible for:

1. Placing the intersection on flash if so commanded via telemetry input or by a manual flash switch on the intersection cabinet door.
2. Sending the intersection status to the network control computers via telemetry (i.e., green configuration, flash status).

3. Sending the status of various switches via telemetry (i.e., police-panel switches, cabinet door ajar, interconnect functions).
4. Informing the network control computers if the last command received was accepted or rejected.
5. Turning all signal lights dark if the signal shutdown switch on the police-panel is set.

2.6 Emergency Control

This section of the actuator program drives the signal control section in emergencies, i.e., when telemetry input commands are invalid or non-existent.

2.6.1 Flash Control

When the intersection is placed on flash for any reason, a flash timer is initialized to time-out the flash. Emergency control is responsible for counting-down this timer and terminating the flash mode at the proper time.

The various ways that intersection flash is initiated are:

1. Restart after a power failure. Flash is terminated after the restart flash time expires.
2. Signal confirm error. Flash is terminated after a minimum flash time expires. However, if such errors occur too frequently, the intersection will be placed on "permanent" flash until the program is reinitialized.
3. Manual flash switch in the cabinet is set. Intersection will flash for a minimum time, and then remain flashing until the switch is reset.

4. Telemetry input flash command. Flash will stay on for a minimum time, and then continue until telemetry input command indicates "no-flash."

After the flash is terminated, the emergency signal control mode is initiated.

2.6.2 Emergency Control Patch Panel

Located on the front panel of the actuator is a "patch panel" or "pin matrix" that can be read by the actuator. By inserting or removing pins from the panel, one can change the various times used by the emergency control section. Appendix C shows the bit assignments of the panel and Section 5 gives detailed instructions on how to use the panel. Briefly, the pin matrix allows the operator to vary the maximum, minimum and extension times for each phase used by emergency control. In addition, the actuator console data switches are used to change the amber times for each movement.

2.6.3 Emergency Control Logic and Phasing

The emergency control logic for the actuator is very simple for two reasons:

1. The actuator memory size is very small (2048 words) and a more sophisticated control algorithm would require more memory.
2. There was no need for a more sophisticated technique, since the emergency control logic will only be used as a back-up controller when the network computer communication link is broken.

The signal phasing selected was basically that described in Reference 2, and is commonly known as "TTI Phasing." In the actuator program, the phases are numbered as shown in Figure 2.3.

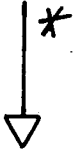
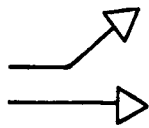

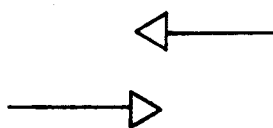
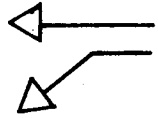
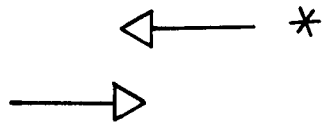
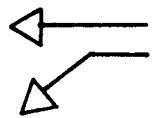

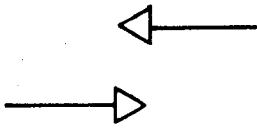

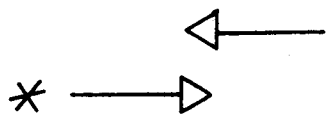
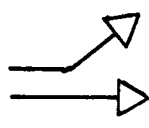
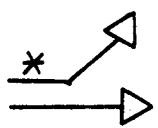
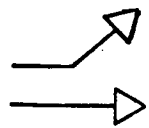
PHASE NO.	WEST SIDE ("A" SIDE)	EAST SIDE ("B" SIDE)
1		
2		
3		
4		
5		
6		
7		

FIGURE 2.3 EMERGENCY CONTROL PHASING

Phase 7 is used at only one intersection, Forest Lane, which has a two way frontage road on the west side. The rules used by the actuator in effecting emergency control are:

1. Whenever it is necessary to begin emergency control, begin with phase 1 if the intersection is on flash. Otherwise, begin with the phase that includes all movements that are currently green at the intersection. If no such phase exists, enter flash mode for the minimum time period.
2. The maximum, minimum, extension and amber times will be selected from the computer front panel. However, phases 2 and 5 will have maximum = minimum time with no extension allowed. The times for phase 7 at Forest Lane will be stored in the actuator.
3. Each phase will be displayed for the minimum time. Thereafter, each actuation of the appropriate stop-line loop detector (marked with an asterisk in the phasing diagrams) for that phase will cause the phase to be displayed for one extension interval from the instant of actuation. However, if after the minimum time has expired no actuation occurs during an extension interval, that phase will be terminated. The clearance interval will be displayed, after which the next sequential phase will begin.
4. As mentioned in Section 2.5.4, walk light processing is handled in the signal processing section. The emergency control section simply commands the signal control section to display the appropriate green lights for each phase.

Emergency control is invoked under the following circumstances:

1. After flash mode is terminated for any reason.
2. If a new valid telemetry command is not received for a 60 second period.
3. A command is received via telemetry to begin emergency control.

Emergency control will continue in effect until neither event listed in items 2 and 3 above is present and the current emergency control phase has reached termination.

3.0 TECHNICAL DESCRIPTION OF ACTUATOR PROGRAM

The flow charts in Appendix A have been carefully constructed to be an accurate description of the actuator program for frontage road control. Scattered throughout the flow charts are narrative comments that explain the function of a group of boxes. These comments can be perused for a general understanding of the program flow. This section should be used along with the flow charts and program listing in Appendix F to obtain detailed knowledge about the internal structure of the actuator program. It should be noted that the program logic described in this section and in the flow charts could be applied to any computer used for traffic control, not necessarily Nova computers.

3.1 Actuator Computer Hardware Description

The specifications for the actuator and related I/O equipment are given in Reference 3. The computer hardware characteristics are listed here for convenience.

1. The actuator is a general purpose minicomputer manufactured by Data General Corporation. It is commonly referred to as a Nova 1210 and is described in Reference 1.
2. The CPU has a general instruction set common to all of the Nova computers.
3. It is equipped with a 16-bit word, 2K (2048 words) read/write core memory.
4. The memory cycle time is 1.2 microseconds. Arithmetic, logical and branch instructions require 1.35 microseconds, memory reference with accumulator 2.55 microseconds, memory

increment and decrement 3.15 microseconds, and I/O instructions either 2.55 or 3.15 microseconds.

5. A real-time clock is provided that is capable of generating interrupts at any of the following program selectable frequencies: AC line frequency (60 Hz), 10 Hz, 100 Hz, or 1000 Hz.
6. Power failure protection is provided. This insures that an electrical power failure does not affect memory. It also causes a processor interrupt when power begins failing, and causes an automatic restart at location 0 when power is restored.
7. Four 16-bit accumulators are provided. They are referred to as AC0, AC1, AC2 and AC3. The last two, AC2 and AC3, can also be used as index registers.
8. Page 0, which is the first 256 words of memory, can be directly addressed from anywhere in memory. There are three other direct addressing modes that allow an instruction to address 256 locations relative to 1) the program counter, 2) AC2 or 3) AC3.
9. Any memory location can be indirectly addressed from anywhere in memory.
10. Each actuator has a standard front console. This console contains data entry and function switches that permit manual operation of the computer. Also provided is an automatic program load switch which is used to initially load a bootstrap loader.

11. A plug is provided for connecting a portable paper tape reader, which is used to load programs into the actuator. in conjunction with the front console.
12. An 8x8 "pin matrix" is located on the actuator front panel. Grounding pins can be inserted into any of the 64 slots for external parameter programming. The actuator program can input the matrix as four 16-bit words. They are used to select signal light timing values.
13. Digital input/output registers are provided as shown in Appendix C. They are used to interface the actuator with the intersection hardware.
14. A telecommunications interface is provided between the actuator and a voice-grade telephone line. This allows a 1-bit input and 1-bit output for the actuator to communicate with the network control computers.
15. A stall alarm is provided that must be pulsed periodically, or the intersection flasher in the cabinet will be activated and any outputs by the actuator to the signal lights will be ignored. The actuator resets this alarm by doing an output to address 57_8 . Once the stall alarm is activated, a manual reset must be done before the actuator can operate the signals.

3.2 Programming Problems, Constraints and Notes

The main problem in programming the actuator was that only 2K words of memory were available. However, since the processor speed was much faster

than necessary, it was possible to make extensive use of subroutines and program loops to conserve core. This allowed the program to contain all the required functions for communicating with the network control computers, as well as a traffic responsive emergency (back-up) traffic control section. Currently the program uses all but 128 words of memory. Timing tests have shown that only about one-third of the available time (i.e., CPU speed) is required under the most extreme conditions.

Another constraint was the the program should be flexible enough for use at many types of intersections with a minimum amount of alteration. This necessitated the "task" method that is described later. Each separate program function, such as telecommunications, speed detector processing, emergency control, etc., was coded independently from other tasks. Also, all detector bit assignments, emergency control phasing, and other intersection-related data were defined in tables. Thus, for conversion to another type intersection, such as an 8-phase, all that is required is to change these tables and the intersection-dependent tasks.

A problem arose as the result of having a Nova 1200 computer with an ASR 33 teletypewriter as the only program preparation and checkout device available. With its low speed (10 characters/second) paper tape reader, punch and printer, the ASR 33 was not suited for long input tapes. It was, therefore, necessary to break the actuator program up into several different segments that could be updated and assembled independently. The program segments are shown in Appendix E. The Data General Nova Paper Tape Editor program was used to prepare and later make changes to the tapes and punch new tapes. The Data General Nova Extended Assembler was used to assemble the individual tape segments after editing. Both of these programs are

described in Reference 4. All programming described in this report was done in Nova Assembly Language.

3.3 Program Segments 0 and 8, Common Definitions

Program segment 0, as seen in the listings, defines all "common" data and variables, most of which are in page 0. Generally, this is data that is referenced by more than one task or program segment. In some cases, it is an array that must be indexed using AC2 or AC3. Every element defined in this segment is listed as an entry point. All other program segments are prefixed during assembly by segment 8 which defines all the "common" data as externals. Thus, all program segments have access to the "common" data in segment 0. Appendix B describes all "common" data in alphabetical order. Program segment 8 also contains the definitions of input/output device codes used by the various tasks.

There are various parameters located throughout the actuator program that might be described as system configuration parameters. These define such things as the number of tasks, the number of bits in a telemetry message, the minimum intersection flash time, etc. These parameters as well as their location in the program are described in Appendix G should it ever be necessary to alter them.

3.4 Program Segment 1, Common Subroutines

These are useful subroutines that can be called indirectly through a transfer vector in page 0. The method of call, calling arguments and register usage, are given in the program listings. The function of each subroutine is described in detail. Many of these subroutines are reentrant, which means that multiple tasks can use them independently.

3.5 Program Segment 2, Initialization, Interrupt, Task Control

3.5.1 Initialization Section

The initialization section, entry point INITR, is entered after an initial program load or when power is restored after an electrical power failure. This section is responsible for assuring that the restart is carried out in an orderly manner. First a 5 second delay loop is entered to allow the real time clock to stabilize. During this loop the signals are placed on flash. All tasks are initialized to start at their original addresses, the telemetry section is initialized, and the real-time clock is restarted at 1000 Hz. A wait loop is then entered until the first real-time clock interrupt occurs.

3.5.2 Interrupt Section

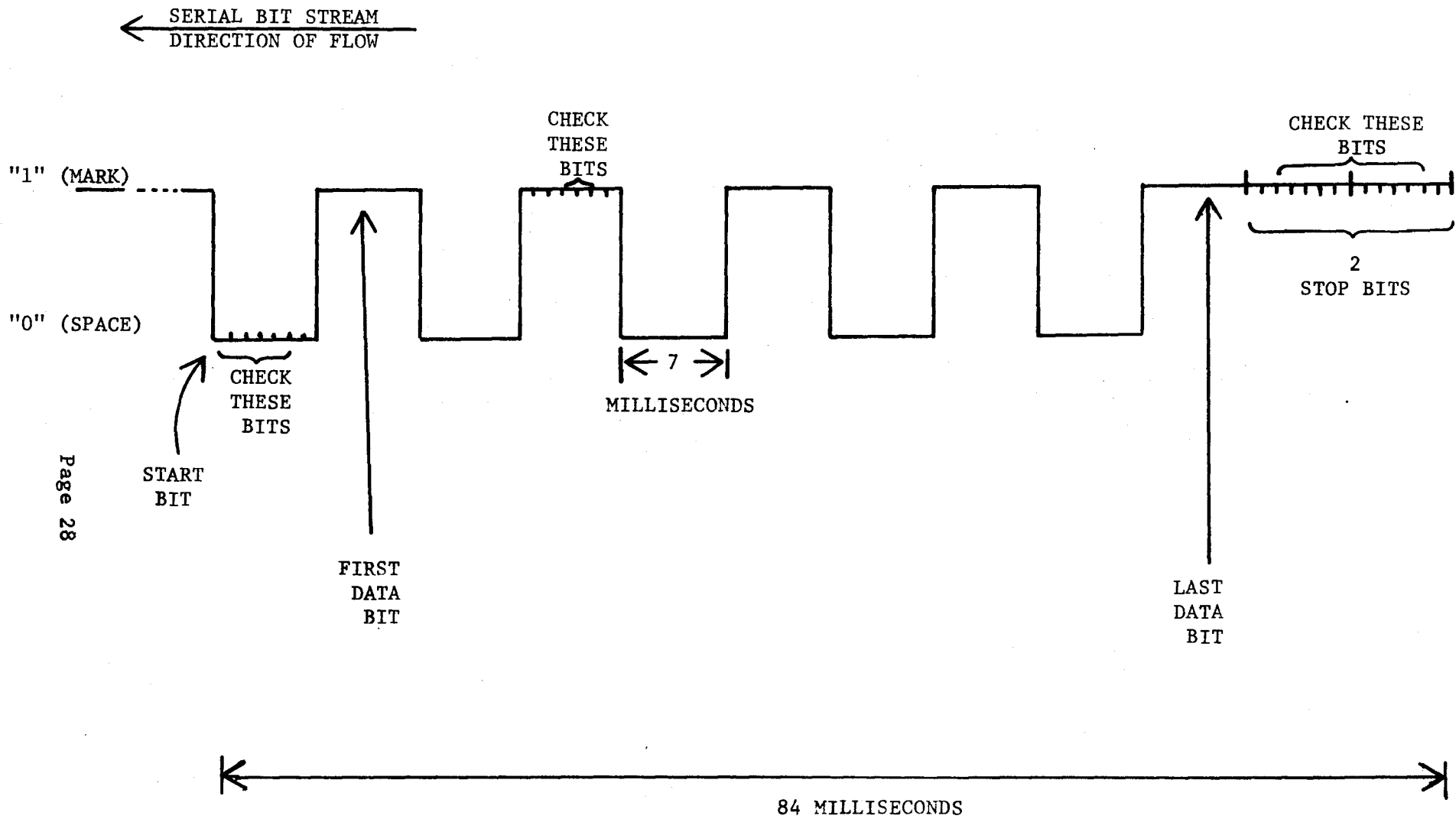
The interrupt section, entry point INTRP, is entered whenever any interrupt occurs since its address is stored in location 1. The hardware stores the contents of the program counter (PC) in location 0 when an interrupt occurs. The interrupt routine tests the device flags to determine what type of interrupt occurred and takes action as follows:

1. Power failure interrupt - A jump to the restart routine (INITR) is placed in location 0. The intersection signals are placed on flash. Then a CPU halt is executed to await the power failure.
2. Undefined interrupt - If the interrupt is not caused by power failure or the real-time clock, the device code on the interrupt line is read and inserted into an instruction that clears that device's busy/done flags. Interrupt is enabled and the return is made to the interrupted PC.
3. Real-time clock interrupt - This section is executed every millisecond when the done flag is set for the real-time clock.

Two main functions are handled in this routine, telemetry processing and task control.

a. Telemetry processing - There is a one bit input and one bit output for two way communication with the network control computers. These bits operate the data set receiver and transmitter which are connected to a two wire telephone line. Full duplex operation, i.e., simultaneous transmission/reception, is accomplished by using separate frequencies for transmitting and receiving. This one data bit interface means that all communication must be in a serial fashion. The actuator software is responsible for formatting the serial data bits into a meaningful message.

(1) Telemetry format - Both input and output use the same message size and bit rate. A typical message is shown in Figure 3.1. A message is continuously sent and received at the bit rate shown (7 milliseconds) such that a new message occurs every 84 milliseconds. The timing is handled by the actuator software. The various messages that can be sent are shown in Appendix D. Since the telemetry input/output is done in a serial fashion under software control, independently of telemetry hardware, it is possible to alter the message size, format and rate simply by changing the program. Currently the telemetry messages consist of 3 synchronizing bits and 9 data bits. Nine



Page 28

BIT TIME \approx 7 MILLISECONDS \approx 143 BITS/SECOND \approx 12 FRAMES/SECOND

FIGURE 3.1 TELEMETRY DATA FORMAT

bits were needed in order to send all of the inter-section status information upstream to the network computers. If in the future it is decided that less status information will suffice, it will be possible to revert to the more standard 8-data bit format by altering the program.

(2) Telemetry input - This section is entered upon each real-time clock interrupt (1 millisecond). The processing proceeds as follows:

- (a) When a start bit (space) is received (i.e., a 1 to 0 transition in the bit stream) check the next five millisecond samples. If any 1 bits occur, set an error flag.
- (b) Sample the third millisecond of each data bit. Then test the 4th and 5th millisecond samples. If either is different from the 3rd, set error flag.
- (c) For each data bit that is "1", complement the parity cell. After the last data bit check the parity cell. If it is not "1". set the parity error bit in telemetry output and set error flag.
- (d) Check the 2 through 12 millisecond samples of the stop bits. If any discrepancy occurs (i.e., "0" bits), set a resync. flag and set the command reject bit in telemetry output.

- (e) After the last stop bit sample, check the error flag. If set, set the command reject bit in telemetry output. If not set, compare the current input word to the previously received input word. If they are equal, store the full telemetry message in the telemetry input buffer and proceed to (f). If they are unequal, skip steps (f) and (g).
 - (f) If the input word requests an abort/stall, call subroutine ABORT.
 - (g) If the input word requests a telemetry resync., set the resync. flag.
- (3) Telemetry output - This section is entered every 7th time (i.e., 7 milliseconds) through the real-time clock interrupt routine. The processing proceeds as follows:
- (a) If the resync. flag is set, output a continuous "1".
 - (b) If any bit is set in the priority output word (Output Format 15), send the word and clear the bit.
 - (c) If output buffer (TOBUF) is not empty, remove and send first word from it. Otherwise send Output Formats 0, 1 and 2 sequentially.
- (4) Telemetry Input/Output Buffering - The events that the actuator program is responsible for communicating

to the network control computers will occur in a random, asynchronous fashion. Many such events will occur simultaneously or in a very short time span. An example would be that several vehicles might activate various detectors simultaneously. Since a new telemetry word can be sent only every 84 milliseconds, some means must be provided for queueing or stacking messages to be sent. The same argument also applies to telemetry input, since messages might be received faster than they can be processed. For these reasons, a technique referred to as "circular buffering" was used for queueing telemetry messages. A circular buffer can be diagrammed as shown in Figure 3.2. Two subroutines are provided in segment 1 for adding or removing words from the buffer.

(Note: (A) = contents of A).

(a) Subroutine PUT - Puts words into buffer as follows:

(a-1) Set $w = (IN) + 1$. If $w = FIRST$,
 $w = (FIRST)$.

(a-2) If $w = (OUT)$, buffer is full.

(a-3) Store word into (IN).

(a-4) Set $IN = w$.

(b) Subroutine GET - Fetches word from the buffer as follows:

(a-1) If $(IN) = (OUT)$, buffer is empty.

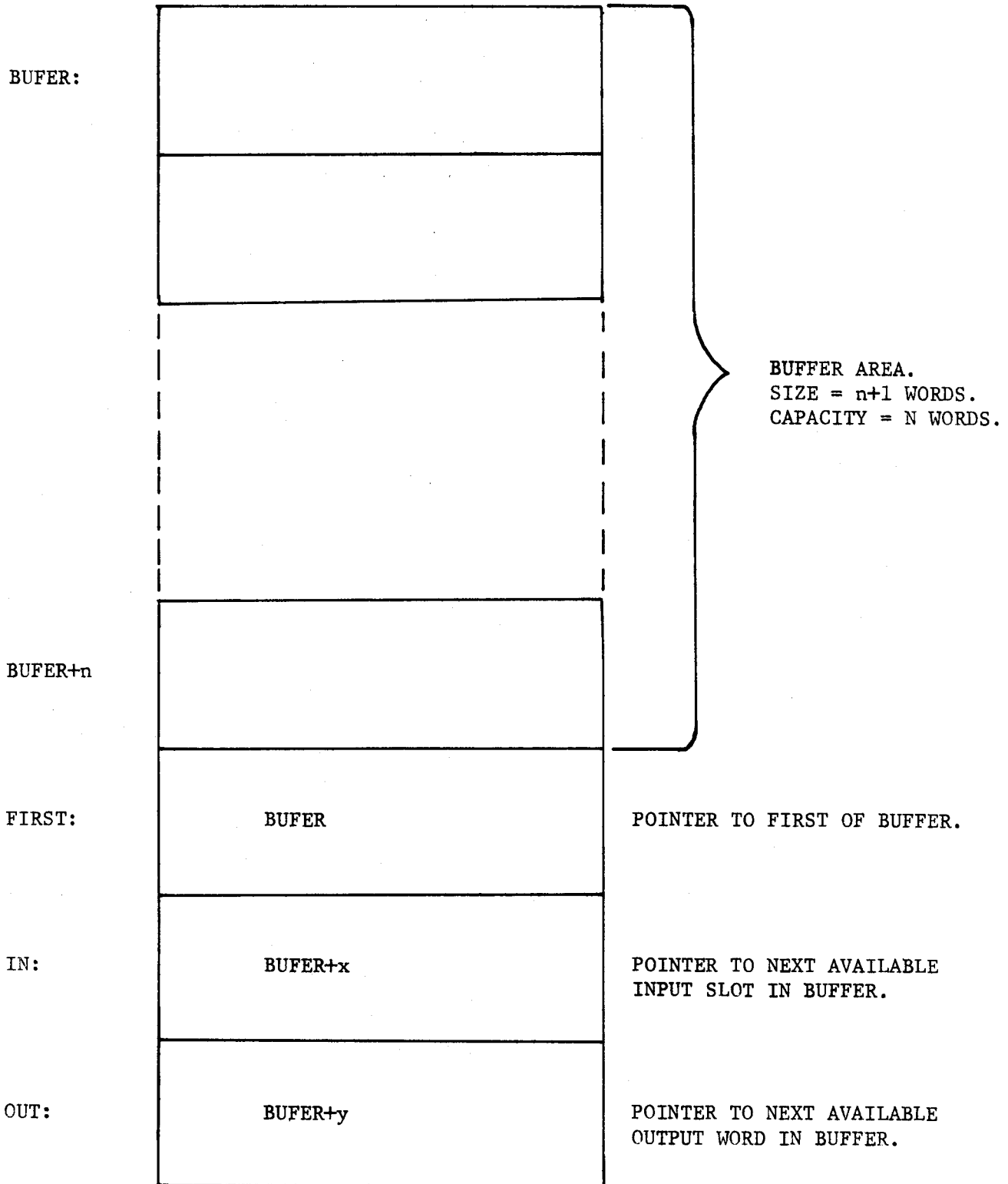


FIGURE 3.2 CIRCULAR BUFFER FORMAT

(a-2) Load word from (OUT).

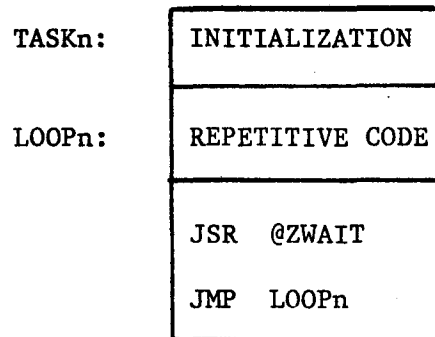
(a-2) Set $OUT = (OUT) + 1$. If $(OUT) = FIRST$,
 $OUT = (FIRST)$.

This circular buffering technique allows each task that places words into a buffer and the task that removes words from the buffer to act independently of each other, since each task manipulates its own pointer (IN or OUT). The subroutines PUT and GET are totally reentrant which means that they can be called from multiple tasks independently. Very little overhead is involved in buffering since PUT requires only 12 words and GET requires 11 words.

b. Task Control Processing - There are five separate tasks in the actuator program, TASK0, TASK1, TASK2, TASK3 and IDLE. TASK0 has the highest priority, IDLE the lowest. All of the tasks except IDLE are executed on a timed basis. For example, TASK0 is executed every 10 milliseconds. IDLE is actually a "background job" that uses any excess CPU time that the other tasks do not need. It was used only for checkout purposes (see 3.10) and is not a part of the operational program.

(1) Task description - A task is a section of code that has an entry point, or starting address, performs a repetitive function every time frame, and termi-

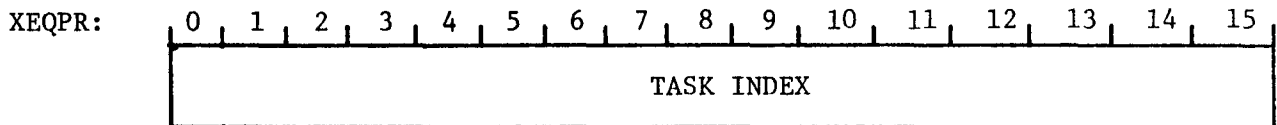
nates with a call to a task scheduler. A task might be diagrammed as follows:



Initially, the task will be entered at TASKn. This portion of code can be used to initialize task variables. The code beginning at LOOPn will be executed every time the task is activated (i.e., at the task repetition rate). The instruction JSR @ZWAIT indicates to the task scheduler that the task is complete for the current frame. The JMP LOOPn will be the first instruction executed at the beginning of the next time frame, and simply defines the task loop that is to be repeated. An example of a task is the section of code that processes the vehicle magnetometer detectors (TASK0). It is entered every 10 milliseconds to scan these detectors and to compare their current status to their status during the previous scan, and to take appropriate action when any change occurs. Then the task is terminated until the next 10 millisecond frame.

(2) Task control parameters - There is a group of variables in page 0 used for task control storage. These variables are described generally in Appendix B.

(a) Executing task index or priority (XEQPR) - This is a one word variable that contains the index of the currently executing task. During task control processing it is loaded into AC2 for the purpose of indexing the task control arrays described below.

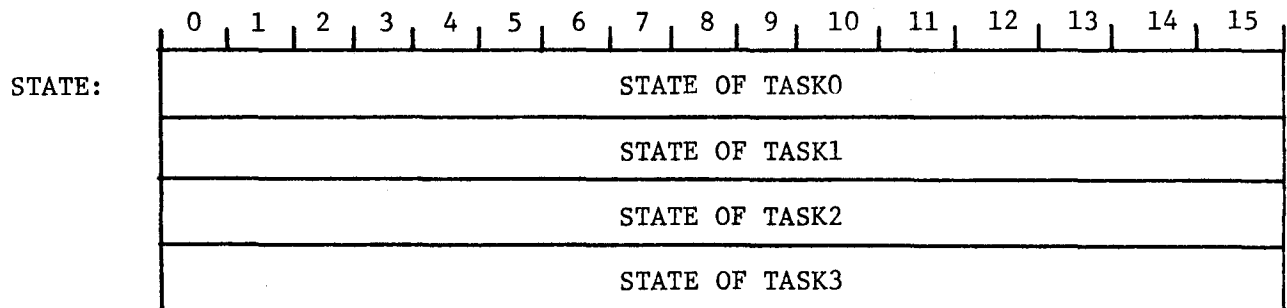


The values are:

TASK0 = -4, TASK1 = -3, TASK2 = -2

TASK3 = -1, IDLE = 0 (checkout only).

(b) Task state (STATE) - This is a four element array that gives the current status of each task.



The values are:

= 0: Task is dormant. Used for checkout only.

= 1: Task is waiting for timer overflow.

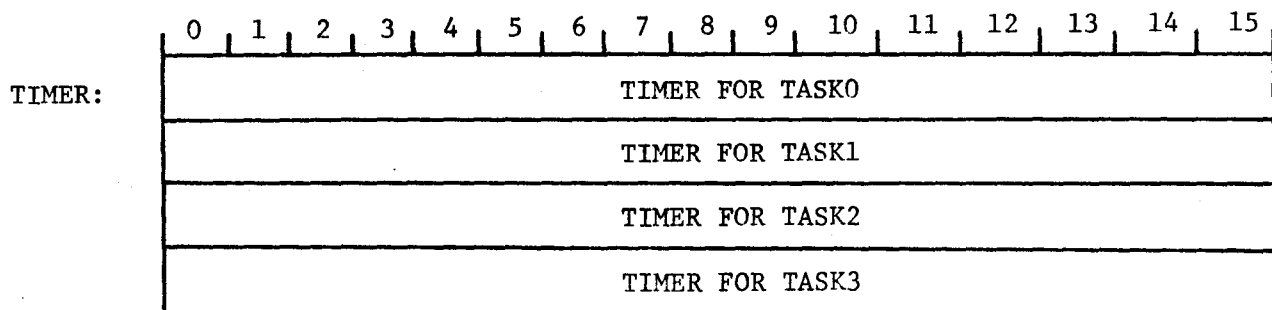
= 2: Task timer has expired. Task is either pending or in execution.

= n (greater than 2): Timer for the task overflowed $n - 2$ times without the task calling the scheduler (JSR @ZWAIT).

That is, $n - 2$ interrupts were missed.

NOTE: IDLE is always assumed to be in STATE = 2 and therefore has no STATE entry.

- (c) Task timer - A four element array that contains a software timer for each task. Each timer is decremented every millisecond. When a timer reaches 0, the task STATE is incremented and the timer is recycled to the repetition time (see CYCLE) for the task.



The values represent the number of milliseconds remaining in the current frame for each task.

NOTE: IDLE does not require a task timer.

- (d) Task repetition time (CYCLE) - A four element array that contains the number of milliseconds in a time frame for each task. This value is loaded into TIMER whenever the TIMER value reaches 0.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CYCLE:	REPETITION TIME FOR TASK0															
	REPETITION TIME FOR TASK1															
	REPETITION TIME FOR TASK2															
	REPETITION TIME FOR TASK3															

Values can range from 65,535 down to 1 millisecond.

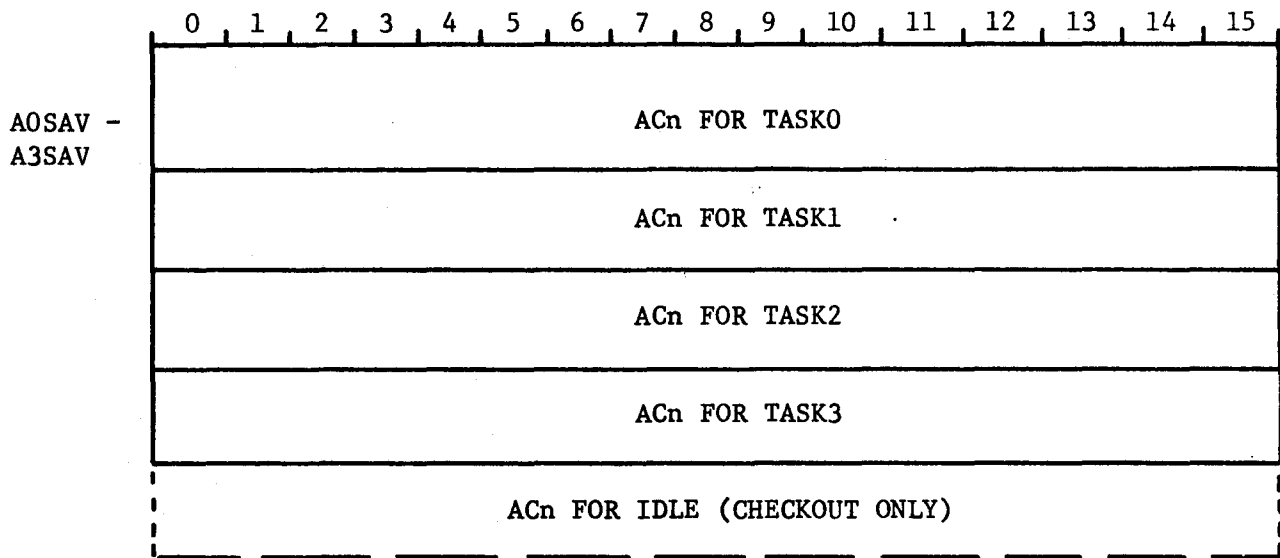
NOTE: No CYCLE value is required for IDLE.

- (e) Task execution address and carry state (PCSAV) - A five element array that contains the current program counter value as well as the state of the carry bit for each task.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
PCSAV:	PC FOR TASK0															C_0
	PC FOR TASK1															C_1
	PC FOR TASK2															C_2
	PC FOR TASK3															C_3
	PC FOR IDLE (CHECKOUT ONLY)															C_{IDLE}

C_n = carry bit for TASKn.

(f) Accumulator save area (AOSAV - A3SAV) - A group of four five element arrays that contain the accumulator register values for each task.



n = 0 → 3.

(3) Task interrupt processing - When the real-time clock interrupt occurs (every millisecond), a scheduler flag, SCEDF, is first tested. If it is non-zero the task scheduler was interrupted, so no registers are saved since the scheduler will be reentered at the end of interrupt processing anyway. However, if SCEDF = 0, all registers are saved in the task control arrays PCSAV → A3SAV indexed by (XEQPR), which always contains the index of the currently executing task. After telemetry processing, all entries in the array

TIMER are decremented. If any TIMER entry is 0, it is recycled to the frame time and the corresponding STATE entry is incremented to indicate that the task is awaiting execution.

- (4) Task scheduling and execution - The scheduler flag, SCEDF, is then set and interrupt is enabled. The task scheduler is entered. It simply searches the STATE array from top to bottom until an entry >1 is found, indicating a task awaiting execution. The index of that task is placed in XEQPR, the registers are loaded from that task's entries in the AOSAV - A3SAV arrays, carry is loaded from PCSAV bit 15, and the execution is begun at the address in PCSAV bits 0-14 after SCEDF has been cleared. If no STATE entry >1 is found, one of two alternatives is selected. In checkout mode, XEQPR is set to 0 and the above logic is followed, thus causing IDLE to be executed wherever it left off. In operational mode, a single instruction JMP . is executed, which simply waits for the next 1 millisecond interrupt. The scheduler flag (SCEDF) is left set so that registers will not have to be saved when the interrupt occurs.
- (5) Task completion - After a task completes processing for each time frame, it calls subroutine WAIT (JSR @ZWAIT). This subroutine disables

interrupt and stores the address of the call +1 (which is in AC3) and the carry bit into the PCSAV array entry for the calling task. No other registers are saved at this point. Therefore, when the TIMER for the task expires, indicating the start of a new time frame, execution of the task will begin at the instruction following the JSR @ZWAIT. In the current actuator tasks, this instruction is always a branch back to the repetitive portion of the task, although it does not have to be. For example, a task could be set up as follows:

TASKn:	INITIALIZATION
ODD:	PROCESSING FOR ODD FRAMES
	JSR @ZWAIT
	PROCESSING FOR EVEN FRAMES
	JSR @ZWAIT
	JMP ODD

It should be noted that the accumulators ACO - AC3 cannot be expected to have the same values after the call to WAIT that they had before the call, since WAIT does not save these registers.

After saving PC, subroutine WAIT decrements the task's STATE entry, which indicates that the task has completed the current time frame's processing and is awaiting the next frame. The task scheduler is then entered as described in (4) above. This allows the next highest priority task that is awaiting execution to be serviced.

3.6 Program Segment 3, Magnetometer Processing (TASK0)

This task is executed every 10 milliseconds. It is responsible for reading the current status of input Z and comparing it bit by bit to the value read in the previous 10 millisecond period. Two conditions are tested for each bit:

1. A 1 to 0 transition - This indicates that a vehicle just left the detector's area of influence. A bit is set in the common word "VOLUM" to indicate the passage of a vehicle. This word will be processed later in TASK1 (see 3.7). Also, the vehicle occupancy time is moved from the array SPCEL to the corresponding entry in the array SPTAB. The SPTAB array will also be processed in TASK1. The entry in SPCEL is zeroed in preparation for the next vehicle.
2. A continuing 1 bit - This indicates that a vehicle is currently over the detector. The SPCEL entry for this detector is incremented by 1, thus accumulating occupancy in 10 millisecond units.

The question might arise as to why some of the magnetometer processing is deferred to TASK1. The reasons are twofold:

1. The telemetry output detector volume bits are more easily assembled and buffered from one task. Since TASK1 has to process $Z + 1$ and $Z + 2$ volumes, it is the logical place to handle the Z volumes.
2. The average speed calculations require a multiplication for each detector occupancy. The actuator does not have a hardware multiply instruction, so a subroutine is used for this purpose. Several magnetometer actuations in the same time frame might cause such calculations to extend beyond the end of the TASK0 frame, especially if the time frame were lowered from its present value. Therefore, the average speed calculations are handled in TASK1 which operates less frequently and at a lower priority.

3.7 Program Segment 4, Detector Processing (TASK1)

This task is executed every 50 milliseconds. Its functions are:

1. To set telemetry output volume bits in array OVOL when a 0 to 1 transition occurs on any detector bit for input $Z + 1$.
2. To increment an occupancy counter for each continuing 1 bit for the loop detectors in input $Z + 1$.
3. To set telemetry output occupancy bits in word OOCUP when any of the occupancy counters for input $Z + 1$ overflow, and to reset these counters to their initial values.

4. To set telemetry output volume bits in array OVOL when a 0 to 1 transition occurs on any pedestrian pushbutton bit for input Z + 2.
5. To set telemetry output volume bits in array OVOL for all bits that are set in the common word "VOLUM." These bits were set by TASKO for 1 to 0 transitions over the input Z detectors.
6. To calculate average speeds for the four approaches to the diamond intersection. The process is as follows:
 - a. A bit set in VOLUM indicates that a vehicle just left the speed detector. Therefore, TASKO has also placed the occupancy time of that vehicle in the proper entry in the array SPTAB. The occupancy, or trap time, represents the time in .01 second units that the vehicle occupied the detector. If the vehicle length were known, an exact speed could be computed, but for our purposes an average vehicle length of 18 feet has been assumed.
 - b. Let t = occupancy time from SPTAB. A test is made to see if $20 \leq t \leq 1800$, which represent a range of 1 to 90 feet/second. If t is outside this range, it is ignored.
 - c. Let v = current average trap time (occupancy time) for the approach direction of this detector. If $(v + 30) < t$, set $t = (v + 15)$. This produces a dampening affect that prevents one slow vehicle from drastically altering the average speed if it is slower by more than .3 seconds over the average.

- d. If $t > 128$, set $\lambda = 128$. Otherwise set $\lambda = t$.
- e. Calculate the new average trap time $v = v + \frac{\lambda (t - v)}{128}$.

The multiplication is done by a subroutine and the division by shifting.

- f. This method of computing average trap times was developed for use in the Dallas North Central Expressway Ramp Control Program. It is described in detail in Reference 5.
- 7. After all average trap times have been computed, a table look up is done using the table SPCOD to translate each of the four average trap times into 4-bit speed codes that can be sent via telemetry. However, a new speed code is sent only if it differs from the previously sent value for the approach, as given in the array TOUTW.
 - 8. The array OVOL and the word OOCUP are then placed in the telemetry output buffer in four bit groups. The format of these words is shown in Figures 3.3 - 3.5. After each 4-bit group is placed in the buffer, the four bits are zeroed in preparation for the next frame. Should the telemetry output buffer be full, these bits will not be zeroed and thus they will remain set until a subsequent frame occurs when the buffer is not full. See the listings for subroutine PTMDT.

3.8 Program Segment 5, Signal Light Processing (TASK2)

This task is executed every 100 milliseconds. Its main purpose is to act as a software interface between other parts of the actuator program and the traffic signal lights. The specific functions accomplished are:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
INPUT WORD LOCATION	Z BIT 0	Z BIT 2	Z BIT 3	Z BIT 5	Z BIT 6	Z BIT 8	Z BIT 9	Z BIT 11	Z BIT 1	Z BIT 4	Z BIT 7	Z BIT 10	Z+1 BIT 0	Z+1 BIT 3	Z+1 BIT 6	Z+1 BIT 9
										HAS-KELL Z BIT 12	WAL-NUT Z+1 BIT 5	* **	FOR-EST LANE Z+1 BIT 8			
TELEMETRY OUTPUT LABEL	VOL 1	VOL 2	VOL 3	VOL 4	VOL 5	VOL 6	VOL 7	VOL 8	VOL 9	VOL 10	VOL 11	VOL 12	VOL 13	VOL 14	VOL 15	VOL 16
TELEMETRY OUTPUT FORMAT #	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9

* = HASKELL, Z, BIT 13
 ** = MOCKINGBIRD AND FOREST LANES, Z+1, BIT 5

FIGURE 3.3 OVOL BIT ASSIGNMENTS

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
INPUT WORD LOCATION	Z+1 BIT 11		Z+2 BIT 9	Z+2 BIT 7	Z+2 BIT 10	Z+2 BIT 3	Z+2 BIT 1	Z+2 BIT 4								
TELEMETRY OUTPUT LABEL	VOL 17		PED CKT #1	PED CKT #2	PED CKT #3	PED CKT #4	PED CKT #5	PED CKT #6								
TELEMETRY OUTPUT FORMAT #	13		13	13	14	14	14	14								

FIGURE 3.4 OVOL+1 BIT ASSIGNMENTS

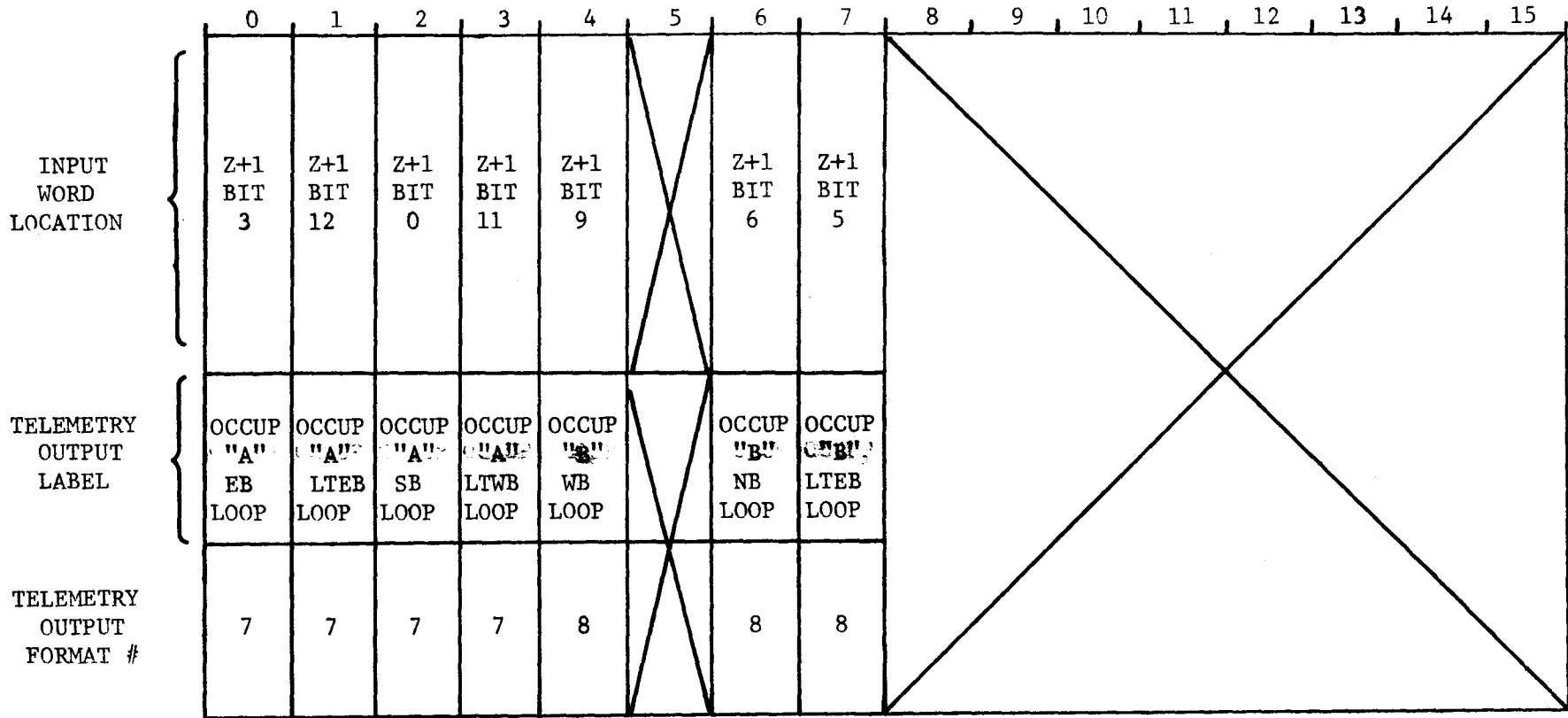


FIGURE 3.5 OCCUP BIT ASSIGNMENTS

1. To form the logical "OR" of the current loop detector input word (Z + 1) with the common word LOOP. The word LOOP will be processed by TASK3.
2. To read the current status of the AC sensors for all signal lights. These bits are then compared to the last output word bits for each of the three outputs (red, amber, green). Errors are processed as described in 2.5.3. A "serious" error, such as an illegal green light on, causes the intersection to be placed on flash for the minimum time if the error persists for two consecutive samples. Any type of signal confirm error causes the flag CNFER to be set. Also, if either of the police-panel switches "signal shutdown" or "manual flash" is set, the subroutine STFLS is called to set up the signal output words to their flash configuration.
3. To decrement a timer (STMR) for each green light or walk light that is on. This is an "absolute minimum" timer that is set whenever the green is first turned on. The green is not allowed to be turned off until its timer decrements to 0. Whenever the timer for a walk light decrements to 0, the walk light bit is immediately turned off and the "don't walk" light turned on.
4. To decrement a timer (STMR) for each amber light that is on. This amber time is set whenever a green light is first turned off and the corresponding amber is turned on. When the amber timer reaches 0, the amber light is turned off and the corresponding red light is turned on.

5. To decrement the telemetry input error timer (ERTIM). If this timer ever reaches 0, it is reset to 1 and a telemetry input error flag (TMERF) is set to 1. As long as new valid telemetry input commands are being received at least every 60 seconds, this error flag will not be set.
6. To encode the status of the current green AC sensors into the codes shown in Appendix D for future telemetry output.
7. To remove a word from the telemetry input buffer and process it. The processing proceeds as follows:
 - a. If the input word is input format 0 (i.e., input bit 3 = 0), the "on-line request" bit is first tested. If it is set and the intersection is currently in flash mode, a command reject flag (CMRJF) is set and the entire command is ignored. If the input "actuator flash" bit is set, then subroutine STFLS is called to place the intersection on flash.
 - b. If the input word is input format 1 (i.e., input bit 3 = 1), the mode is tested. If the intersection is on flash or under emergency control, the command reject flag (CMRJF) is set. The codes for the "A" and "B" intersections are then decoded into a green movement command, as shown in Appendix D. An illegal code results in CMRJF being set. The movement that each bit of this decoded command controls (as well as the bits of the signal output words) is shown in Figure 3.6. The decoded signal command is stored into COMCD.
8. If the intersection flash bit is set, skip to #13 below. If the emergency control mode is set, the emergency command in ECOMC is






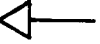


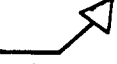


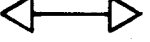

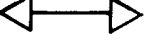
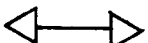
<u>BIT NO.</u>		<u>MOVEMENT</u>
0	"A"	
1	"A"	
2	"A"	
3	"A"	
4		
5		
6		
7		
8	"A"	 (FOREST LANE ONLY)
9	"A"	 (PED WEST SIDE)
10	"A"	 (PED SOUTH SIDE)
11	"A"	 (PED NORTH SIDE)
12		UNUSED
13	"B"	 (PED EAST SIDE)
14	"B"	 (PED NORTH SIDE)
15	"B"	 (PED SOUTH SIDE)

FIGURE 3.6 SIGNAL OUTPUT BIT ASSIGNMENTS

placed into COMCD, thus replacing any telemetry input command that might have been received.

9. If any bit set in COMCD (which contains the desired green movement bits) corresponds to a bit set in the current amber light status word (ASTA), the entire COMCD is ignored and a command reject flag (CMRJF) is set.
10. The desired green bits in COMCD are compared to the current green bits in GSTA. Any 1 to 0 transitions between GSTA and COMCD are stored into NAMS, indicating new amber bits. For each bit in NAMS, the corresponding green timer STMR is tested to see if the green has been on for the required minimum time. If any bit in NAMS fails to pass this test, the command reject flag is set and the entire COMCD is ignored.
11. For each bit in NAMS, the amber time for that movement (AMTM) is stored into the corresponding signal timer (STMR). Then NAMS is OR'ed with ASTA and the result stored in ASTA. The complement of NAMS is AND'ed with GSTA and the result stored into GSTA. Thus, the terminating greens are removed from GSTA and the new ambers are included in ASTA.
12. Any 0 to 1 transitions between GSTA and COMCD are stored into NGRN, indicating new green bits. Subroutine TVALD is then called twice, once to process the green bits for the "A" intersection and once for the "B" intersection. This subroutine picks off each bit of NGRN and tests to see if it could be turned on in GSTA without causing any conflicts

with the current green and amber lights at the intersection. This is accomplished by searching a table of valid greens for each side of the intersection. Any invalid bit is ignored, but this does not invalidate the entire command. Instead, each bit in NGRN is tested separately and whenever a valid one is found, it is set in GSTA and reset in RSTA (red light status). Also, whenever a new green bit is set in GSTA, the corresponding "walk" light bit is also set in GSTA. The "don't walk" bit in RSTA is cleared if the corresponding walk bit is set. The pedestrian minimum time (PDMNT) for the new green movement is stored into the proper signal timer (STMR) and pedestrian timer (PTMR).

13. The new signal status words, RSTA, ASTA and GSTA are then moved to the signal output words ROUDP, AOUDP and GOUDP. The exception is when the signal shutdown switch on the police-panel is set in which case AOUDP is zeroed. RSTA and GSTA will have previously been zeroed by subroutine STFLS in this case.
14. If any telemetry input was received during this time frame, the command reject flag (CMRJF) is tested. If it is set, the current input word is compared to the last input word of the same format type that was rejected. If they are the same, no action is taken. Otherwise, the command reject bit is set for telemetry output. The same procedure is followed for a valid command that was received, i.e., a command acknowledge bit will be sent via telemetry output if the command is not the same as the previously acknowledged one of the same format type.

15. If more words are in the telemetry input buffer, go back to #7 above. Otherwise, the signal output words AOUDP, GOUDP and ROUDP are checked for validity as follows:
 - a. GOUDP is compared to a table of valid green bits (VALGN). There must be at least one entry in VALGN that has a 1-bit wherever GOUDP has a 1-bit.
 - b. Unless flash mode is set, each bit position of AOUDP, GOUDP and ROUDP must contain exactly one bit that is set. This insures that one and only one light will be on per signal head.
 - c. Any failure of tests a or b will cause an internal program abort, which will cause the signals to flash.
 - d. If tests a and b are successful, the words ROUDP, AOUDP and GOUDP will be output to the signal light digital output registers V, V + 1 and V +2 respectively.
16. The following telemetry data is then assembled into the proper output format and stored into the output buffer if a change has occurred:
 - a. The current signal light status codes as shown in Appendix D.
 - b. Signal confirm error bit.
 - c. Cabinet door ajar switch.
 - d. Emergency control mode status.
 - e. Intersection flash status.
 - f. Status of the temperature, signal shutdown, manual flash, and manual cabinet switches.

- g. The four interconnect function switches for Haskell and Fitzhugh.
- h. The status of the telemetry output buffer overflow flag.
- i. The status of the abort/stall input bit.

3.9 Program Segment 6, Emergency Control (TASK3)

This task is executed every .5 second. Its main purpose is to take over control of the signal lights under abnormal circumstances. The specific functions are:

1. Read the four pin matrix words on the actuator front panel. If bit 15 of the first word (input Z + 8) is 0, skip to #4 below.
2. Unpack each pin matrix word by calling subroutine UNPKM. This subroutine shifts out each phase time, adds an appropriate base value to it and stores the result in the proper timing array for maximum (MAXT), minimum (MINT), or extension (EXTT) times in .5 second units.
3. The amber times for movement numbers 0-8 in .5 second units are read from the console data switches. Each amber time is computed by shifting out two bits, adding a base value, multiplying by 5 to obtain .1 second units, and storing the result in the amber time array (AMTM).
4. Determine if more than the permissible number of serious signal confirm errors have occurred by comparing CNFLS to MAXCF (see Appendix G). If so, a "permanent flash flag" (PERFL) is set.
5. A confirm error timer (CNERT) is decremented. If it is 0, the number of serious confirm errors in CNFLS is cleared and CNERT is recycled to its initial value (see CNFTM in Appendix G).

6. If the permanent flash flag (PERFL) is set, subroutine STFLS is called to enter or retain flash mode and TASK3 is exited.
7. The flash timer FLTIM is decremented. If it is non-zero, subroutine STFLS is called and TASK3 is exited.
8. The flash status bit (bit 15 of common word ASTA) is tested. If it is 0, skip to #9 below. If it is 1, indicating flash is in progress, a determination must be made whether or not to terminate flash. The logic used is:
 - a. If the last input format 0 telemetry command had the actuator flash request bit set, leave the flash bit alone and exit TASK3.
 - b. Otherwise, clear the flash bit in ASTA. Set all red bits in RSTA, set the emergency command ECOMC to 0, indicating all red, and set the emergency mode flag EMMOD. Set the emergency phase number EPHSN to 1 and begin emergency control (#15 below). Note that if the flash mode was caused by the police-panel flash switch, the above action will be negated by TASK2 when it reads the status of the switches.
9. If the actuator is not in flash mode, the telemetry input error flag (TMERF) and the emergency control request bit in the latest telemetry input format 0 are tested. If neither is set, skip to #10 below. Otherwise, the emergency mode flag (EMMOD) is tested. If it indicates that emergency control is already in progress, skip to #11 below. If emergency mode is not in effect, the emergency command (ECOMC) is set to the current status of the signal lights (GSTA). A search is then made of a phase table

(PHAST) to determine if any emergency phase has all of the green bits set that are currently on. If so, the phase number for that phase is stored into EPHSN and emergency control is begun (see #15 below). If not, flash mode is entered by calling subroutine STFLS and TASK3 is exited.

10. If there is no request present to begin emergency control, the emergency control flag (EMMOD) is tested to determine if emergency mode is currently in progress. If not, TASK3 is exited.
11. If emergency mode is in progress, the current red light status in RSTA is compared to the complement of the desired green bits in ECOMC. If they are not the same, TASK3 is exited.
12. Whenever the green lights are in the commanded emergency control state, a phase timer PTIM is incremented. If PTIM is not greater than the minimum time for the current phase (EPHSN), TASK3 is exited. Otherwise, PTIM is compared to the maximum phase time. If the maximum time has been reached, skip to #14 below.
13. If the phase timer PTIM is somewhere between the minimum and maximum phase times, the decision whether or not to extend the phase is made. This is done by testing the status of the appropriate loop detector bit for this phase in the common word LOOPS. (As stated in the TASK2 description, the status of the loop detectors is "OR'ed" with LOOPS every 100 milliseconds). If the bit in LOOPS is set, it is cleared and the phase is extended by skipping to #16 below. Otherwise, a green extension timer GRTO is decremented. If the timer has not reached 0, TASK3 is exited.

14. If the phase is due for termination because the maximum time has expired or because an extension interval has passed with no loop detector actuations, a test is made to determine if emergency control should continue. If either the telemetry error flag or the telemetry input emergency control request bit indicates that emergency control is no longer necessary, the emergency mode flag (EMMOD) is cleared and TASK3 is exited. Otherwise, the phase number in EPHSN is incremented to the next phase, or recycled to phase 1 if the last phase is being terminated.
15. When a new emergency phase is initiated, the phase timer PTIM is cleared and the green bits for the new phase are stored into the emergency command word ECOMC.
16. When a phase extension is desired or a new phase is begun, the extension interval time is stored into the green extension timer GRTO. TASK3 is then exited.

3.10 Program Segment 7, IDLE TASK

This task is used only during program checkout. It can operate in either of two modes:

1. As a background task that uses any available time while the other actuator tasks are active. This mode is in effect whenever the actuator program is started at entry point INITR.
2. As a stand-alone job that uses all of the CPU time. This mode is initiated by starting the CPU at entry point IDLE. It is generally used in this fashion to make program patches and

initialize checkout variables before starting the actuator program.

IDLE is basically an interactive teletype program that requires 200_8 words.

The program operation is as follows:

1. When the program is initially entered an "*" is output to the teletype printer. This is a cue to the user that the program is waiting for teletype input.
2. Each character from the keyboard is checked as it is typed. The expected input format is:
 - a. The first five characters represent an octal memory address. A non-octal character in any of these positions is illegal. Exception: A "." as the first character is interpreted as "the last address + 1."
 - b. If the next character after the address field is a carriage return (CR), the contents of the octal memory address will be monitored for change. Each change will cause the contents to be printed as six octal digits followed by a (CR), line feed (LF). This monitor sequence can be terminated by initiating another keyboard input sequence.
 - c. If the character following the address field is a space (blank), then the next six characters are expected to be an octal value that will replace the contents of the octal memory address. The entire input must terminate with a (CR). If the input is valid, the new value will be loaded into the memory address and the contents of the memory

address will be monitored for change as described in b above.

- d. Any illegal character that occurs during the input will cause the sequence (CR), (LF), "*" to be output and a new input sequence to be initiated.

3. Examples:

- a. 01234(CR) - Monitor the contents of location 01234_g.
- b. 00002b177775(CR) - Load -3 into location 2.
- c. 00600b000000(CR) - Load 0 into location 600_g.
 - .b000001(CR) - Load 1 into location 601_g.
 - .b177777(CR) - Load - 1 into location 602_g.
 - .(CR) - Monitor location 603_g.
- d. 008 - Input will be terminated here since an illegal character was read.
- e. 01234b00000(CR) - Input will be ignored since only five characters of the new value were provided.

4.0 PROGRAM TESTING

4.1 Program Configuration

Since the programming of the actuators was done before any actuators were installed at the intersections, it was necessary to perform the checkout on a leased Nova 1200 computer with 8K of memory and an ASR 33 teletype. The checkout version of the program was identical to the operational version, except that it was assembled with the system configuration parameter "DEBUG" equal to 1. This causes the program to behave differently in the following respects:

1. All instructions to read digital input registers into accumulators were replaced with instructions to load the accumulator with the contents of memory locations designated as "debug input registers." For instance, instead of using a DIA instruction to read input Z, there is a LDA instruction from location "ZODEBG" in TASK0.
2. The task IDLE was run as a background job to allow for teletype interaction during checkout.
3. The clock frequency was changed to 10 Hz. This allowed events to occur slowly enough that they could be observed by monitoring the proper words on the teletype.
4. The instructions to write digital output registers were not executed.
5. A simulated telemetry input word (STMIN) was provided in memory that could be changed using the teletype. Every telemetry frame time, this word was loaded into a simulated input shift register (TMINW). Every bit time, a bit was shifted from TMINW to simulate the serial input bit stream.

6. A simulated telemetry output word (STMOT) was also provided that could be monitored on the teletype. It was updated every telemetry frame time with the current telemetry output word.
7. Code was assembled in the task control logic for timing the various tasks. Error checking was incorporated to "halt" if any task ever exceeded its allotted time frame.

4.2 Program Checkout Variables

The variables that were monitored or altered during checkout were:

1. ZODBG - Simulated input register Z (magnetometers). Program segment 3.
2. Z1DBG - Simulated input register Z + 1 (loops). Program segment 4.
3. Z2DBG - Simulated input register Z + 2 (pedestrian buttons). Program segment 4.
4. Z3DBG - Simulated input register Z + 3 (cabinet switches). Program segment 5.
5. DBGAN - "Debug" green AND word. This word is "AND'ed" with the previous green light output word to produce a simulated AC sensor input V + 2. A "0" in any bit position will simulate a failure for a green light to turn on.
6. DBGOR - "Debug" green OR word. This word is "OR'ed" with the previous green light output word to produce a simulated AC sensor input Z + 6. A "1" in any bit position will simulate a failure for a green light to go off.

7. DBRAN - "Debug" red AND word.
8. DBROR - "Debug" red OR word.
9. DBAAN - "Debug" amber AND word.
10. DBAOR - "Debug" amber OR word. All of the above are in program segment 5.
11. Z8DBG - Simulated input register Z + 8 (pin matrix word 1). Program segment 6.
12. Z9DBG - Simulated input register Z + 9 (pin matrix word 2). Program segment 6.
13. Z10DB - Simulated input register Z + 10 (pin matrix word 3). Program segment 6.
14. Z11DB - Simulated input register Z + 11 (pin matrix word 4). Program segment 6.
15. STMIN - Simulated telemetry input word. Program segment 2.
16. STMOT - Simulated telemetry output word. Program segment 2.
17. TETIM - A four element array that contains the task completion time for each task. Subroutine WAIT stores the value of timer + n into TETIM + n when TASKn completes each frame.
18. CLOKF - The real-time clock frequency code. 0 = 16 Hz, 1 = 10 Hz, 2 = 100 Hz, 3 = 1000 Hz.

4.3 Checkout Procedure

The checkout was carried out in three distinct phases:

1. Phase 1 - This phase was basically a test of the interrupt and task control logic, although it was also necessary for the tasks themselves to operate for timing purposes. The test procedure was:

- a. Set CLOKF for 1000 Hz.
- b. Set all simulated input variables to the values that require the most time consuming processing. For instance, the speed detector bits in ZODBG were all set to ones, as were the loop detector bits in Z1DBG. This caused the occupancy counter logic to be entered every cycle.
- c. Set STMIN to an input format 1 command. This caused TASK2 to have to read, decode, and process the command every cycle.
- d. The program was started and the task completion times in TETIM were monitored. Any excessive times in TETIM would have indicated a task was taking too long to complete. Some errors were uncovered and corrected during this test. The test was successfully completed with the result that the lowest priority task (TASK3) was always completed in less than 10 milliseconds of its allotted 500 millisecond time frame. The telemetry output word STMOT was also monitored for the proper configuration.
- e. The IDLE job was then replaced with a loop that incremented a double word counter. The actuator program was started as before and allowed to run for 15 minutes. The double word counter was then used along with the execution time of the loop to compute the CPU time available during the actuator program execution. The result was that 10 minutes, or two-thirds of the total time, was available.

2. Phase 2 - This was a test of the performance of the actuator when no telemetry input was being received. The test procedure was:
 - a. Set CLOKF to 100 Hz.
 - b. Set STMIN to all ones, thus simulating a dead telemetry input line.
 - c. Monitor the telemetry output word STMOT. The result should be:
 - (1) Two frames of all ones (resync.).
 - (2) Output format 0 with the restart bit set.
 - (3) Output formats 1 and 2 with signal codes = 0.
 - (4) Output format 0 with the actuator flash bit set.
 - (5) After relative 15 seconds of flash, output format 0 with emergency control bit set.
 - (6) Output formats 0 and 1 should be sent with the signal codes indicating that the emergency phases 1 through 6 are cycling. Each phase should last the minimum time with 4 second ambers. Also, monitor the output words ROUDP, AOUDP, GOUDP to insure that the proper signal light bits are being output.
 - d. Set Z1DBG = all ones. All phases except 2 and 5 should increase to the maximum time. Occupancy bits should be output every 50 seconds.
 - e. Set Z2DBG = all ones. Proper walk lights should come on at beginning of phases and last for 5 seconds. Check ROUDP, AOUDP, and GOUDP for proper output bits.

- f. Reset Z1DBG and Z2DBG to zero. Phases should return to minimum times and walk lights should stay off.
 - g. Set the manual flash bit in Z3DBG. Check proper TM output and proper signal outputs.
 - h. Set CLOKF = 10 Hz. Set Z0DBG = all ones, then all zeros. Check telemetry output. Should get speed and volume bits for all magnetometer detectors.
 - i. Repeat above for Z1DBG and Z2DBG. Check telemetry volume bit outputs.
 - j. Set individual bits in Z3DBG and check proper output telemetry bits.
 - k. Simulate signal confirm errors by setting various bits in DBGAN, DBGOR, etc. Check for proper telemetry and signal outputs.
 - l. Set each detector input bit individually in Z0DBG, Z1DBG and Z2DBG and test for proper telemetry volume, speed, occupancy, or pedestrian outputs.
 - m. Set the occupancy counters to overflow in one count, set all loop detector bits in Z1DBG, and check for the buffer overflow bit in telemetry output.
3. Phase 3 - This phase was a test of the actuator program during communication with the network computers. The procedure was:
- a. Set CLOKF to 10 Hz. Set STMIN to all ones to simulate a resync. condition.
 - b. Set STMIN with stop bits = 0 and test for telemetry output resync.

- c. Set STMIN with incorrect parity and test for proper telemetry output.
- d. Send abort/stall bit in STMIN. CPU should halt. Reset ASTLF.
- e. Set CLOKF = 100 Hz and restart.
- f. Set perform resync. bit in STMIN. Test telemetry output for resync. mode.
- g. Set actuator flash bit in STMIN. Test for actuator going to flash mode.
- h. Send all red command in STMIN. Should get command reject in STMOT.
- i. Send "no-flash" command in STMIN. Should go into emergency control mode.
- j. Send "on-line" bit in STMIN, followed by all red command. Should get command acknowledge.
- k. Send all combinations of signal codes in STMIN. Check reactions of signals and telemetry output.
- l. Set CLOKF = 10 Hz. Restart.
- m. Bring up various combinations of greens by sending commands in STMIN. Attempt to end some greens too soon. Should get command reject.
- n. Attempt to turn on some greens while corresponding amber is on. Should get command reject.
- o. Attempt to bring up some greens that would conflict with current greens. Command should be ignored until current greens have been on for their minimum time.

- p. Attempt to send illegal command codes in STMIN. Should get command reject.
- q. Repeat k - p with all pedestrian pushbutton bits set in Z2DBG.
- r. Change the pin matrix settings in words Z8DBG, Z9DBG, Z10DB, Z11DB, and the console data switches. Check for proper signal timings.
- s. Change CLOKF = 1000 Hz. Check for proper signal operation during both emergency and on-line modes.
- t. Alternate between on-line and emergency modes by the various means available (i.e., dead telemetry, on-line request bit) and check for smooth transitions.
- u. Unplug the CPU with the console switch in the LOCK position. Plug it back in and check for proper restart of the actuator program.

Of course, many other less formal tests were performed during the debug phases of actuator programming. However, it was felt that the tests described here showed that all required functions of the actuator were operational.

5.0 PROGRAM LOADING AND OPERATION

5.1 Program Preparation

This procedure is accomplished on any off-line Nova. Each of the program segments 0 through 6 is assembled using the extended assembler to produce relocatable object tapes. These tapes are then loaded and linked using the Relocatable Loader (see Reference 7) as follows:

1. Respond with carriage return to SAFE =.
2. Set the address of the word RRFLG + 1 (see listing of program segment 0) into the console data switches. Select loader mode 3.
3. Successively load each object tape in the paper tape reader and select loader mode 1 (teletype reader) or 2 (high speed reader).
4. After the last tape has been loaded, select loader mode 6 to produce a loader map listing.
5. Select loader mode 8 to terminate the loading process.
6. Load the high core binary format punch program (see Reference 8) using the binary loader, and use it to punch memory from address 0 through NMAX as printed on the loader map. Punch location 3 as the starting address on the tape.
7. Splice a copy of the Nova 800/1200 program load tape on the first of the absolute actuator tape just punched.

5.2 Loading the Actuator

The tape produced in the above steps will be loaded into each actuator.

The procedure is:

1. Move the switch inside the computer cabinet from "NORMAL." to "FLASH." Place the police-panel SIGNAL FLASH and SIGNAL SHUT-DOWN switches down.
2. Move the pin from position F-7 to position B-7 of the pin matrix.
3. Connect the paper tape reader. Turn computer key to "ON" position. Place appropriate intersection tape in reader, move reader switch to "RUN."
4. Set the console data switches to 12_8 (switches 12 & 14). Press "RESET" and "PROGRAM LOAD" switches.
5. Binary loader will be read and computer will halt with 121_8 displayed in the "ADDRESS" lights (Bits 9, 11, 15). Set data switch 0 and press "CONTINUE."
6. After the tape is read, the computer should reset the "STALL ALARM" and turn all display board lights dark. At this time place the amber times in the console data switches as shown in Figure 5.2.
7. Remove the pin from pin matrix position B-7 and place it in F-7. Turn the computer key to "LOCK." Disconnect the paper tape reader.
8. Monitor the display board for proper phasing and timing. If all is well, place the police panel FLASH switch up and the computer cabinet switch to NORMAL. The signals should continue to flash and the display board should go dark.
9. When traffic subsides place the police-panel FLASH switch down. There will be a 1-10 second delay before the flash terminates.

10. Walk around the intersection to check for proper signal operation.
11. Be sure the computer key is in the LOCK position before leaving the intersection.

5.3 Pin Matrix Operation

The grounding pin matrix on the actuator front panel is used in conjunction with the computer console to select various times associated with the signal light operation. A pin inserted in a slot produces a binary "1" input to the actuator, and the absence of a pin in a slot produces a binary "0". Figure 5.1 shows the layout of the pin matrix and the assignment of each group of pin slots. The definitions are:

1. MAX(n) = Maximum phase time in seconds for phase n.
2. MIN(n) = Minimum initial interval in seconds for phase n.
3. EXT(n) = Extension time in seconds for phase n.
4. TIM(n) = Total phase time in seconds for phase n. This is used only for the two overlap phases 2 and 5 which have no extensions.
5. READ MATRIX (Position B-7) = Remove pin from here when changing pin matrix values. Insert pin here to cause actuator to use new values in pin matrix and data switches.

The pin should be left in for about 1 second and then removed.

The amber times for each traffic movement are selected by using the console data switches as shown in Figure 5.2. These times are in .5 second units.

Each of the timing parameters will have a base value added to it as it is read from the pin matrix and data switches. These base values and param-

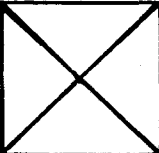
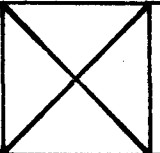
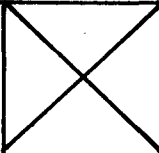
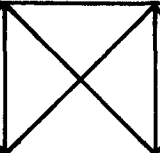
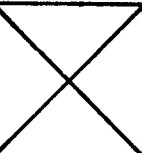
		COLUMN #							
		0	1	2	3	4	5	6	7
ROW #									
A		MAX(1) 2^5	MAX(1) 2^4	MAX(1) 2^3	MAX(1) 2^2	MAX(1) 2^1	MAX(1) 2^0	MIN(1) 2^3	MIN(1) 2^2
B		MIN(1) 2^1	MIN(1) 2^0	EXT(1) 2^2	EXT(1) 2^1	EXT(1) 2^0			READ MATRIX
C		MAX(3) 2^5	MAX(3) 2^4	MAX(3) 2^3	MAX(3) 2^2	MAX(3) 2^1	MAX(3) 2^0	MIN(3) 2^3	MIN(3) 2^2
D		MIN(3) 2^1	MIN(3) 2^0	EXT(3) 2^2	EXT(3) 2^1	EXT(3) 2^0	TIM(2) 2^2	TIM(2) 2^1	TIM(2) 2^0
E		MAX(4) 2^5	MAX(4) 2^4	MAX(4) 2^3	MAX(4) 2^2	MAX(4) 2^1	MAX(4) 2^0	MIN(4) 2^3	MIN(4) 2^2
F		MIN(4) 2^1	MIN(4) 2^0	EXT(4) 2^2	EXT(4) 2^1	EXT(4) 2^0			
G		MAX(6) 2^5	MAX(6) 2^4	MAX(6) 2^3	MAX(6) 2^2	MAX(6) 2^1	MAX(6) 2^0	MIN(6) 2^3	MIN(6) 2^2
H		MIN(6) 2^1	MIN(6) 2^0	EXT(6) 2^2	EXT(6) 2^1	EXT(6) 2^0	TIM(5) 2^2	TIM(5) 2^1	TIM(5) 2^0

FIGURE 5.1 PIN MATRIX LAYOUT






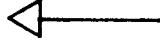


0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
MOVEMENT #0 AMBER TIME		MOVEMENT #1 AMBER TIME		MOVEMENT #2 AMBER TIME		MOVEMENT #3 AMBER TIME		MOVEMENT #4 AMBER TIME		MOVEMENT #5 AMBER TIME		MOVEMENT #6 AMBER TIME		MOVEMENT #7 AMBER TIME	
															
"A"		"A"		"A"		"A"		"B"		"B"		"B"		"B"	

FIGURE 5.2 DATA SWITCH ASSIGNMENTS FOR AMBER TIMES

ometer ranges are shown below:

<u>Parameter</u>	<u>Base Value</u>	<u>Parameter Range</u>
MAX	1 second	1 - 64 seconds
MIN	1 second	1 - 16 seconds
EXT	1 second	1 - 8 seconds
TIM (Phases 2 & 5)	1 second	1 - 8 seconds
AMBER	3.0 seconds	3.0 - 4.5 seconds

Therefore, if all pins for the parameters are removed and all data switches are set to 0, the parameters will assume the base values shown above when a pin is placed in position (B-7).

At Forest Lane, the MAX, MIN and EXT times for phase 7 must be stored or assembled into the MAXT, MINT and EXTT arrays respectively in program segment 6. The amber time for movement 8 must also be stored or assembled into AMTM + 8 in segment 0.

5.4 Clearing a "Stall Alarm" Condition

An actuator hardware problem may cause the stall alarm light in the cabinet to come on. If so, it will be necessary to perform some manual operations before restarting the actuator.

1. Turn the mode key to the ON position.
2. Press the RESET switch.
3. Examine the contents of memory location ASTLF (see the loader map). Record the value.
4. Deposit zeroes in ASTLF.
5. Turn the mode key to the LOCK position.

6. Restart the actuator at location 3.

7. Reset the stall alarm.

The recorded value of ASTLF may make it possible to track down the cause of the failure.

6.0 REFERENCES

1. "How to Use the Nova Computers." Data General Corporation, October, 1972.
2. "Operational Study of Signalized Diamond Interchanges." Charles Pinnell and Donald G. Capelle, Texas Transportation Institute. Reprint from Highway Research Board Bulletin 324 (1962).
3. "Proposal to the State Highway Commission for the Construction of Installation of Frontage Road Traffic Control Equipment in Dallas County, Texas." Texas Highway Department Project #T9001(9), 1972.
4. "Introduction to Programming the Nova Computers." Document #093-000067-00, Data General Corporation, 1972.
5. "Description of Digital Computer Control Program for the North Central Expressway Ramp Control System." Charles W. Blumentritt and William R. McCasland, Texas Transportation Institute Research Report 836-2. July, 1972.
6. "Functional Specifications for a Prototype Actuator Device." Charles W. Blumentritt, Texas Transportation Institute, January, 1970.
7. "Relocatable Loader." Document #093-000039-00. Data General Corporation, 1969.
8. "Binary Format Punch." Document #093-000001-00. Data General Corporation, 1969.

APPENDIX A

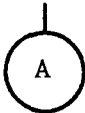
ACTUATOR PROGRAM FLOW CHARTS


FLOW CHART

TABLE OF CONTENTS

	<u>PAGE NO.</u>
1. NOTES ON THE FLOW CHARTS	A-1
2. PROGRAM SEGMENT 1 - SUBROUTINES	A-2
a. STPRI	A-3
b. PUT	A-4
c. GET	A-5
d. STFLS	A-6
e. ABORT	A-7
f. PTSUB	A-8
3. PROGRAM SEGMENT 2 - INTERRUPT	A-9
a. Initialization and Restart	A-10
b. Interrupt Processing	A-11
c. Telemetry Input Processing	A-13
d. Telemetry Output Processing	A-22
e. Task Timer Control	A-26
f. Task Scheduler	A-27
g. Subroutine WAIT	A-28
4. PROGRAM SEGMENT 3 - TASK0 (SPEEDS)	A-29
5. PROGRAM SEGMENT 4 - TASK1 (DETECTORS)	A-31
6. PROGRAM SEGMENT 5 - TASK2 (LIGHTS)	A-39
7. PROGRAM SEGMENT 6 - TASK3 (EMERGENCY)	A-55
8. PROGRAM SEGMENT 7 - IDLE	A-63

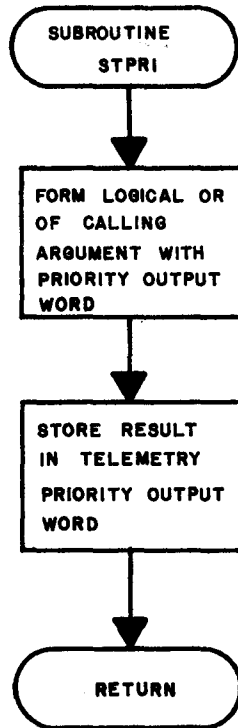
NOTES ON THE FLOW CHARTS

1. The symbol  indicates a connector on the same page with label A.

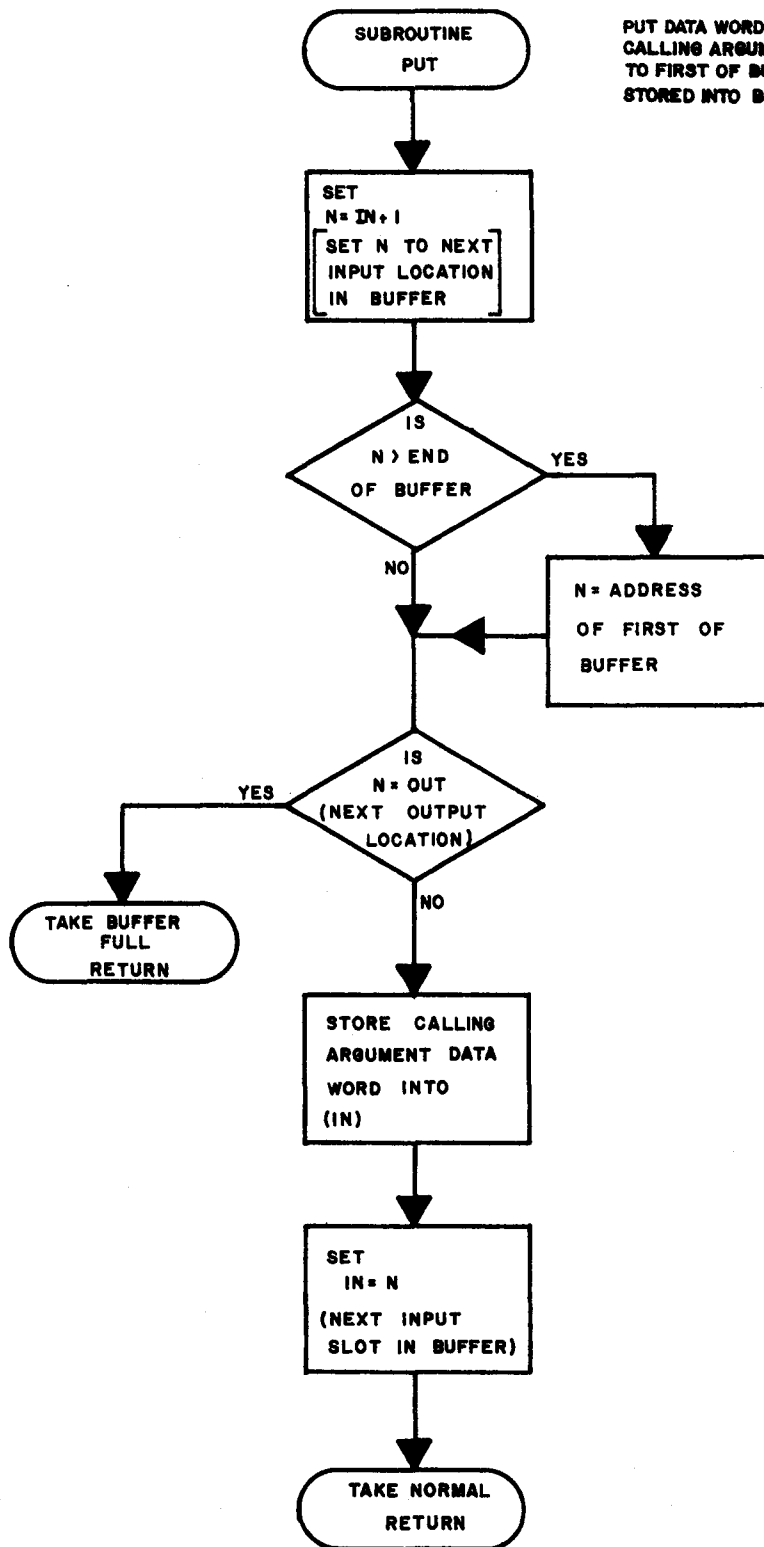
2. The symbol  indicates an off-page connector with label A located on page n.

3. Program loops are shown enclosed within a broken line box.

PROGRAM SEGMENT 1

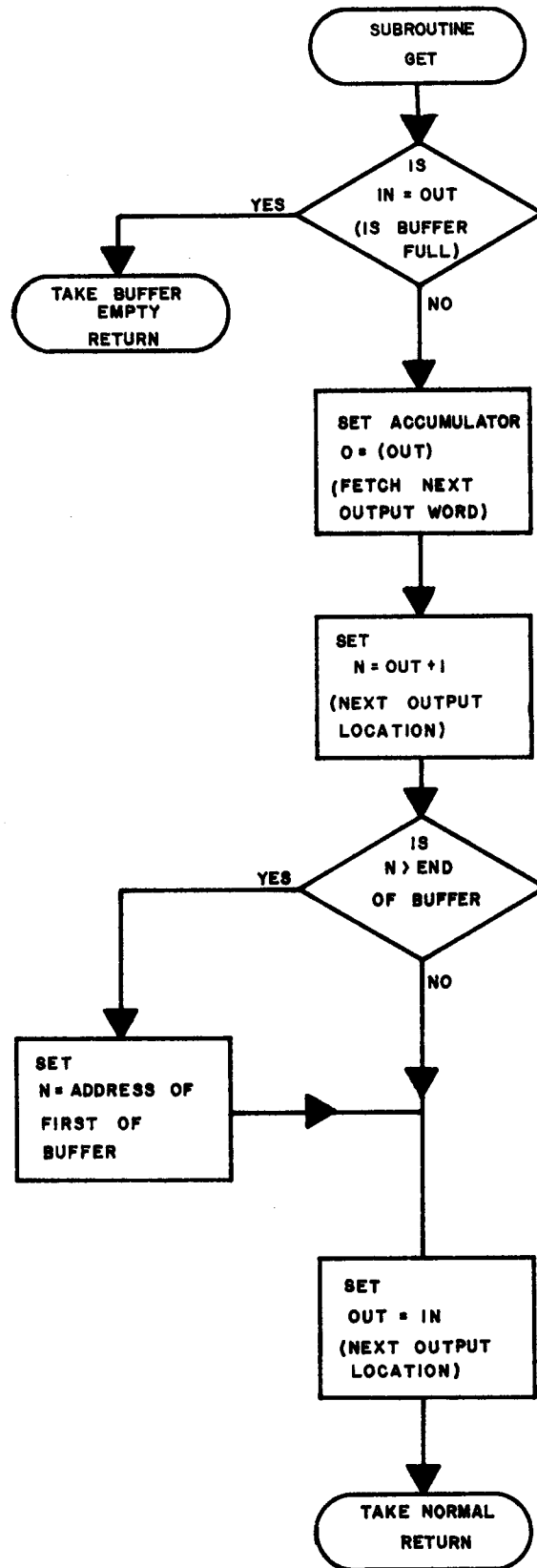


SET BITS IN TELEMETRY
TOP-PRIORITY OUTPUT WORD.
CALL WITH ARGUMENT VALUE
IN ACCUMULATOR O.

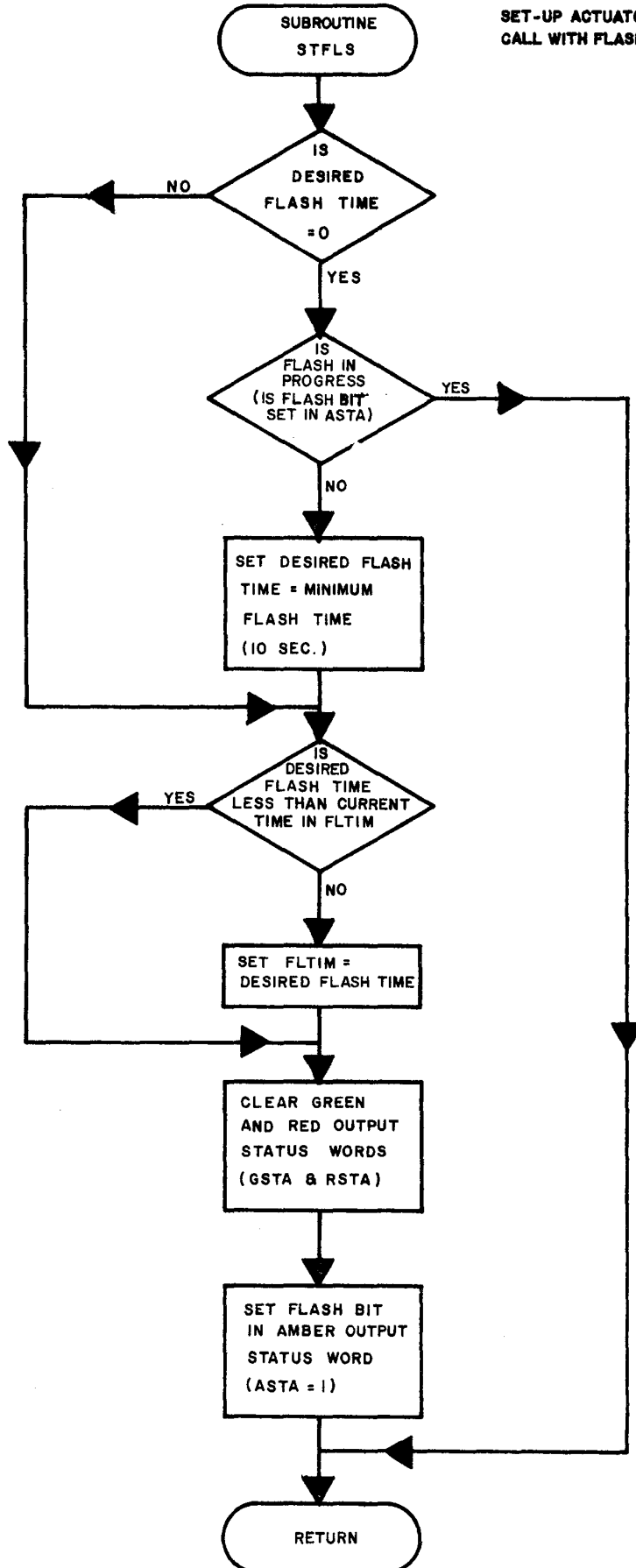


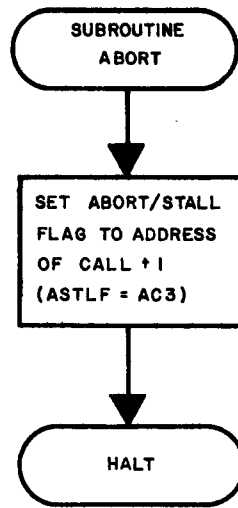
PUT DATA WORD IN CIRCULAR BUFFER. CALLING ARGUMENTS INCLUDE POINTER TO FIRST OF BUFFER, AND WORD TO BE STORED INTO BUFFER.

FETCH DATA WORD FROM CIRCULAR BUFFER. CALLING ARGUMENT IS A POINTER TO FIRST OF BUFFER. RETURNS DATA WORD IN ACCUMULATOR

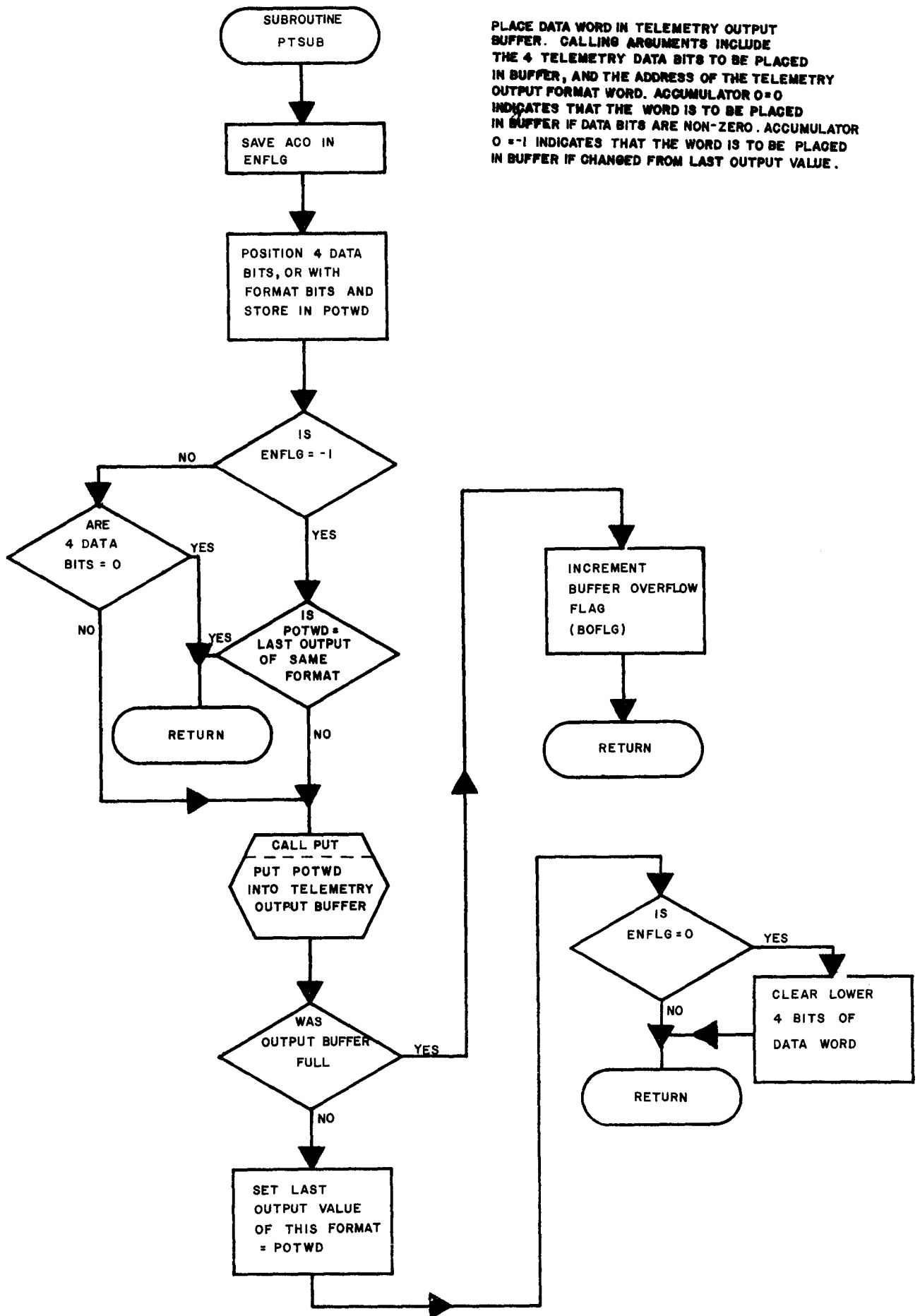


SET-UP ACTUATOR FLASH MODE.
CALL WITH FLASH TIME IN ACCUMULATOR O.



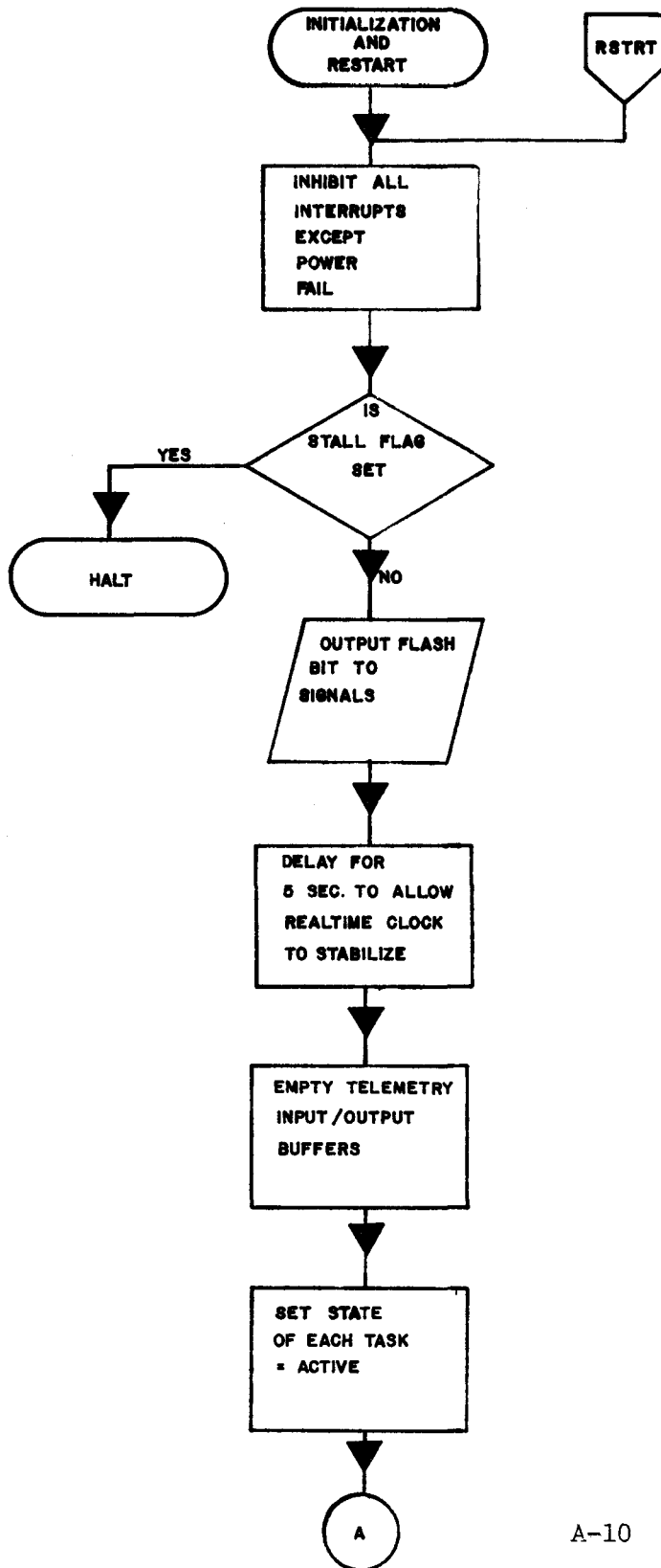


**ACTIVATE ABORT/STALL MODE BY
SETTING ABORT FLAG AND PERFORMING HALT.**

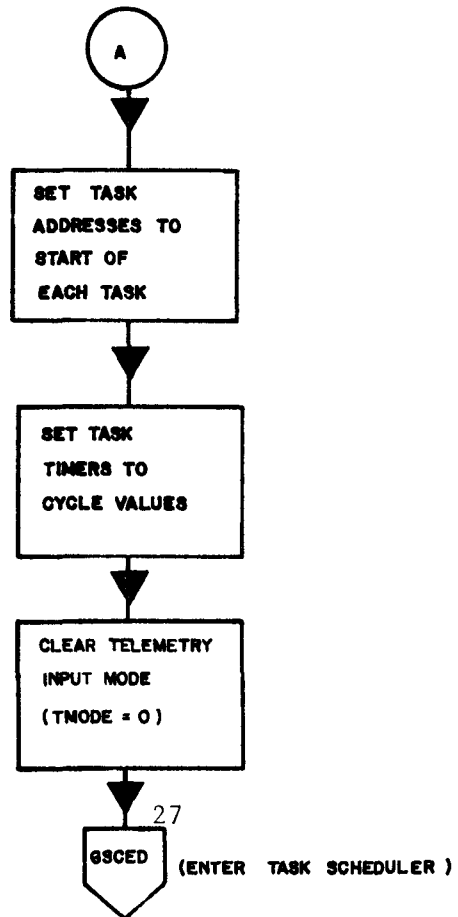


PLACE DATA WORD IN TELEMETRY OUTPUT BUFFER. CALLING ARGUMENTS INCLUDE THE 4 TELEMETRY DATA BITS TO BE PLACED IN BUFFER, AND THE ADDRESS OF THE TELEMETRY OUTPUT FORMAT WORD. ACCUMULATOR 0=0 INDICATES THAT THE WORD IS TO BE PLACED IN BUFFER IF DATA BITS ARE NON-ZERO. ACCUMULATOR 0=-1 INDICATES THAT THE WORD IS TO BE PLACED IN BUFFER IF CHANGED FROM LAST OUTPUT VALUE.

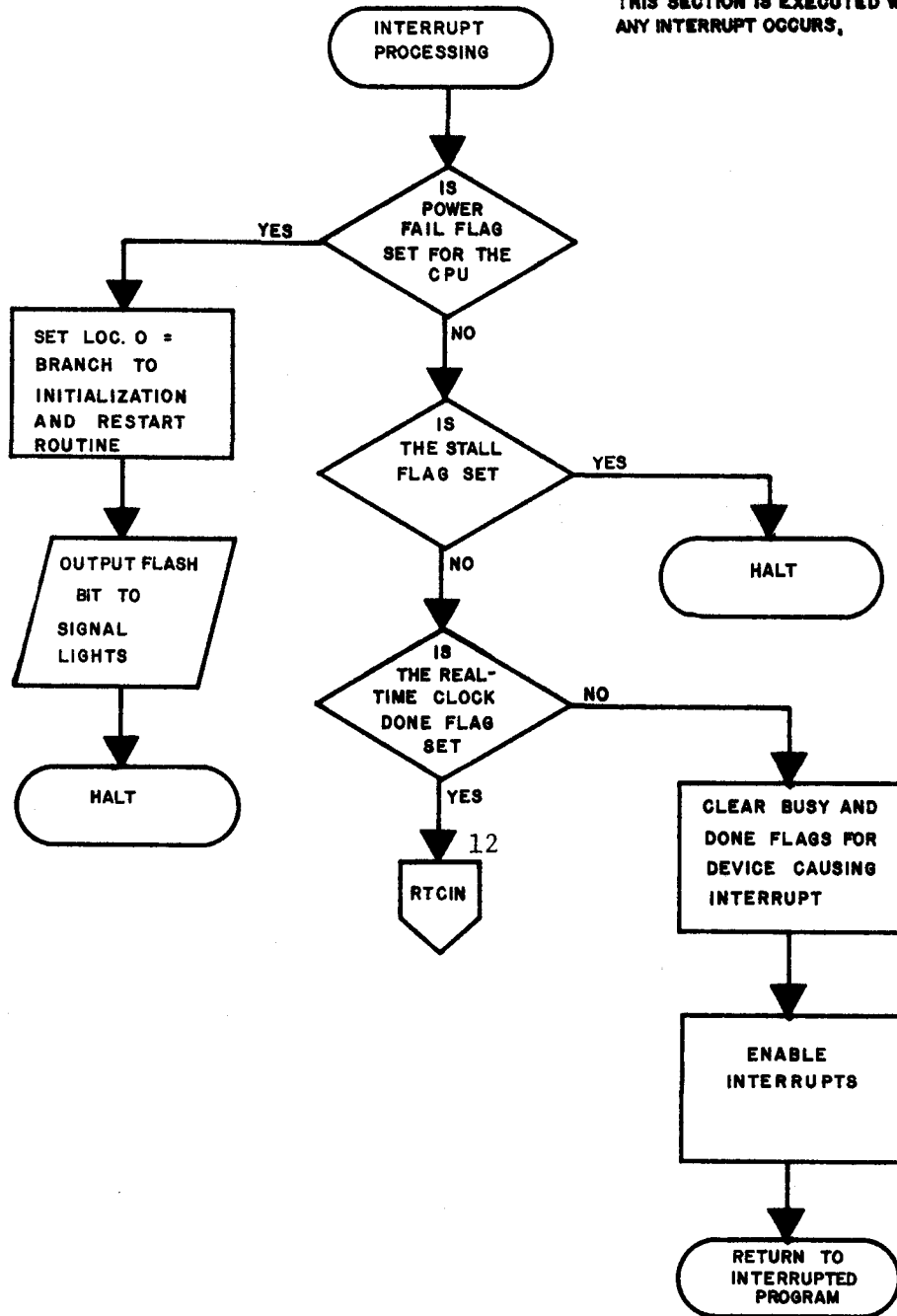
PROGRAM SEGMENT 2



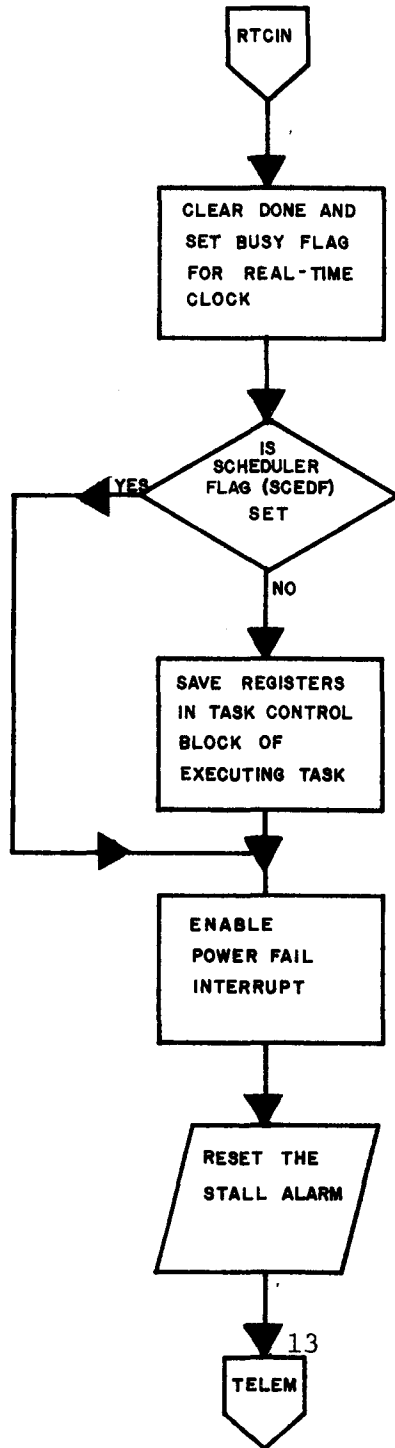
THIS SECTION IS EXECUTED AFTER A RELOAD OR AFTER A POWER FAILURE



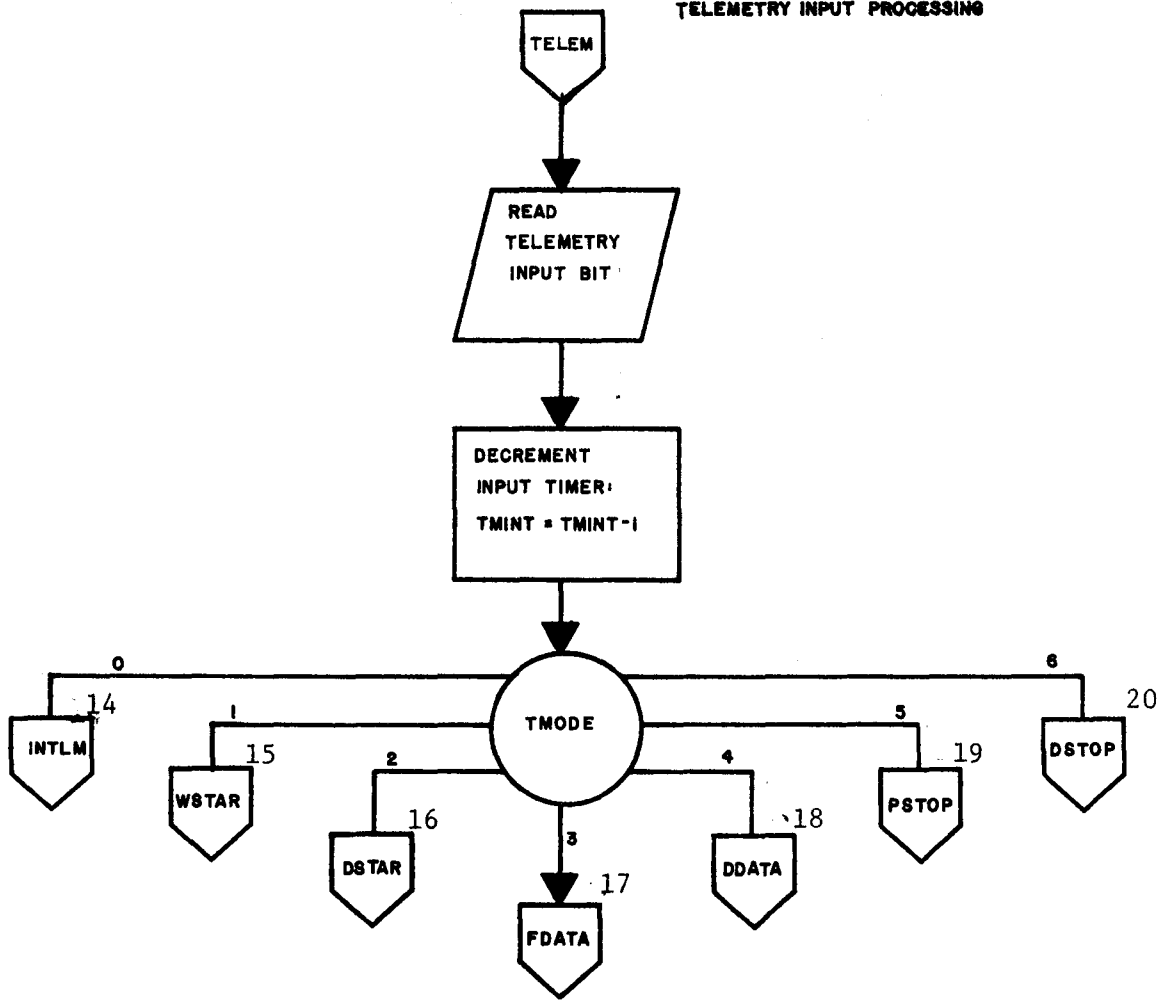
THIS SECTION IS EXECUTED WHEN ANY INTERRUPT OCCURS,

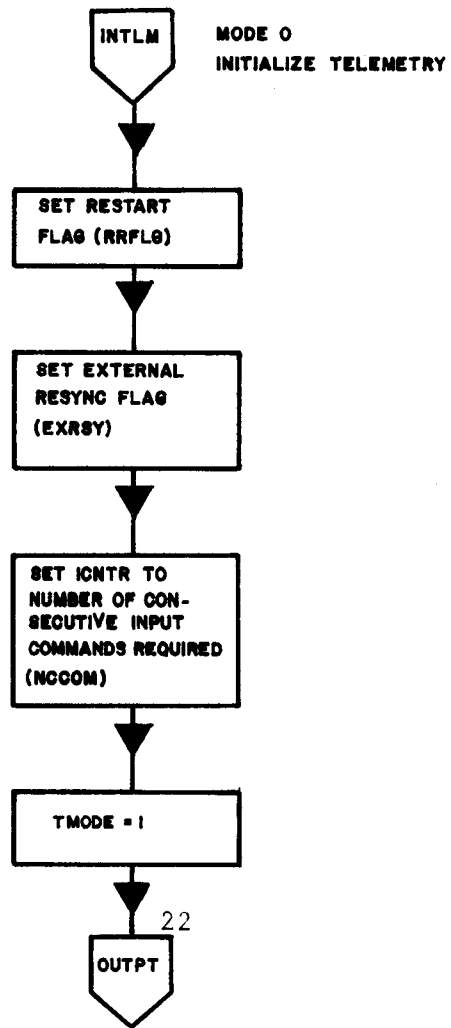


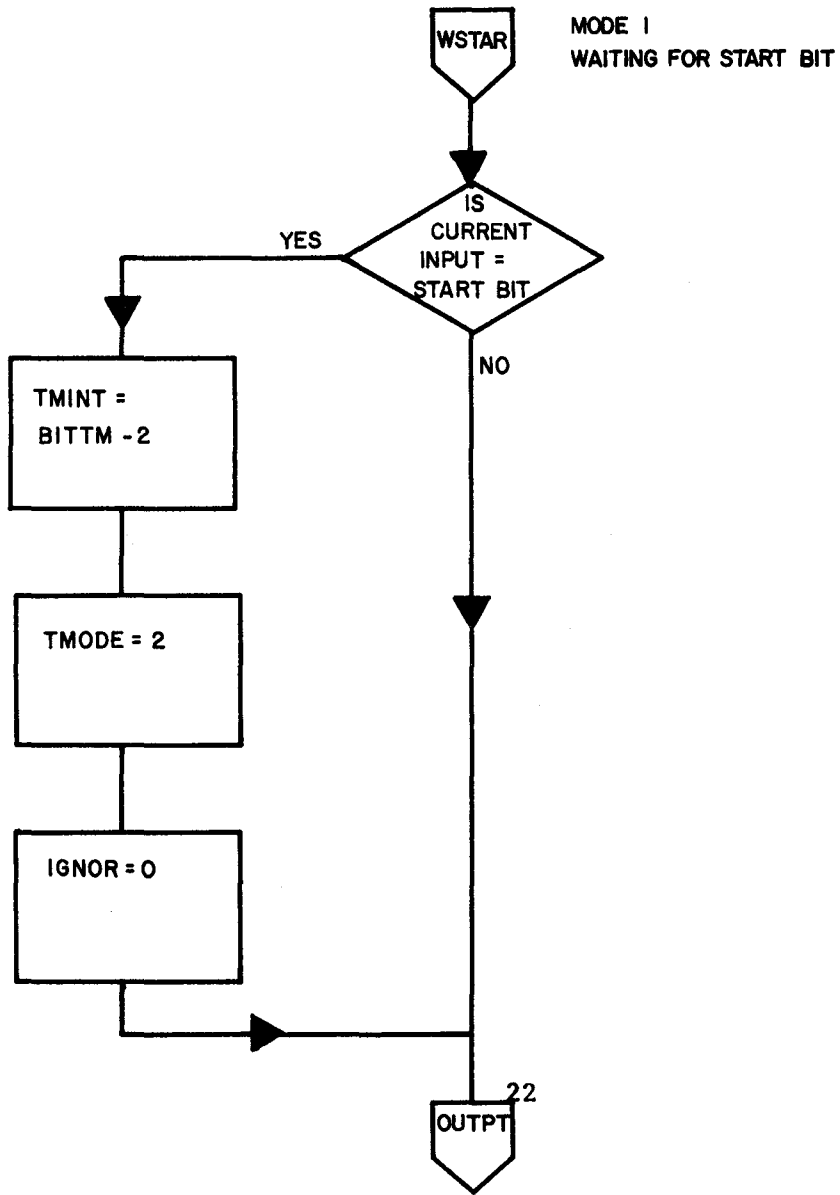
(REAL-TIME CLOCK INTERRUPT)
THIS SECTION EXECUTES EACH
MILLISECOND.

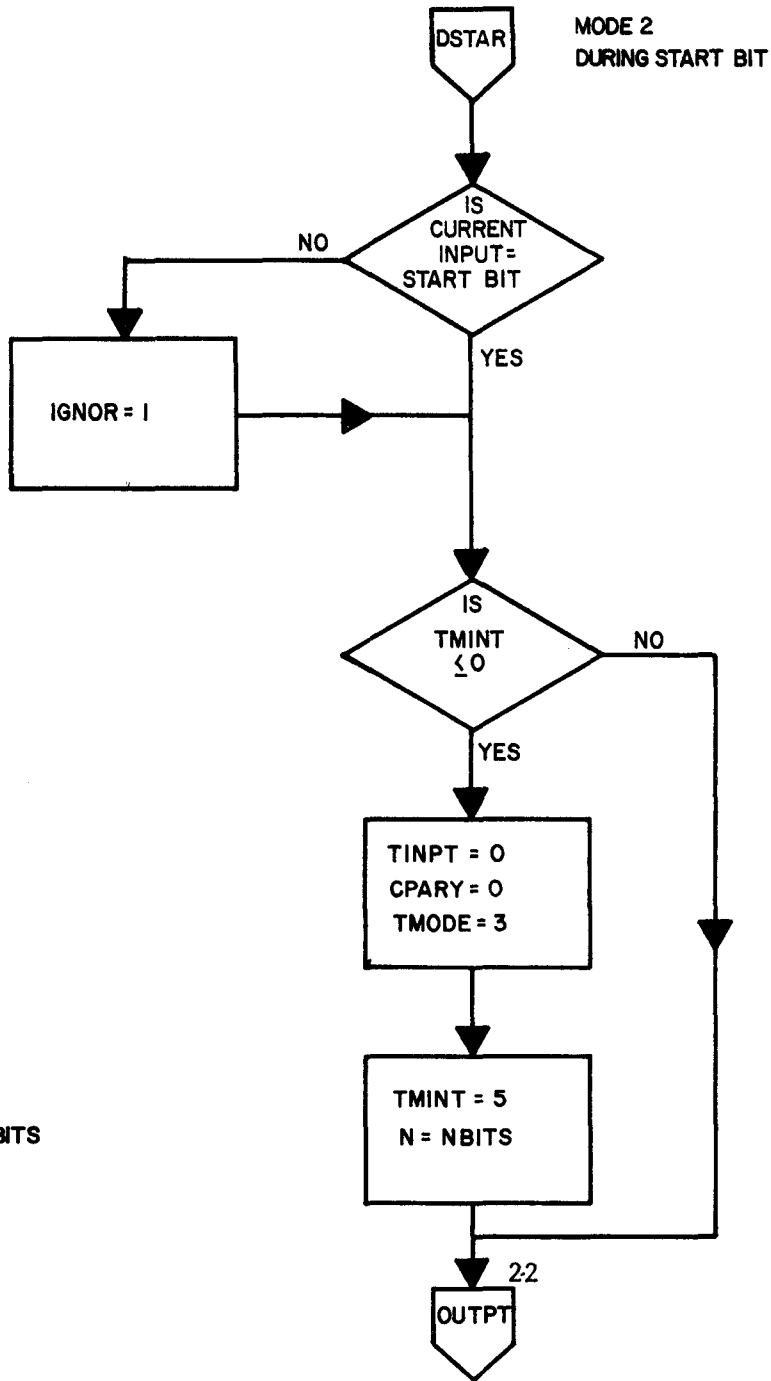


TELEMETRY INPUT PROCESSING

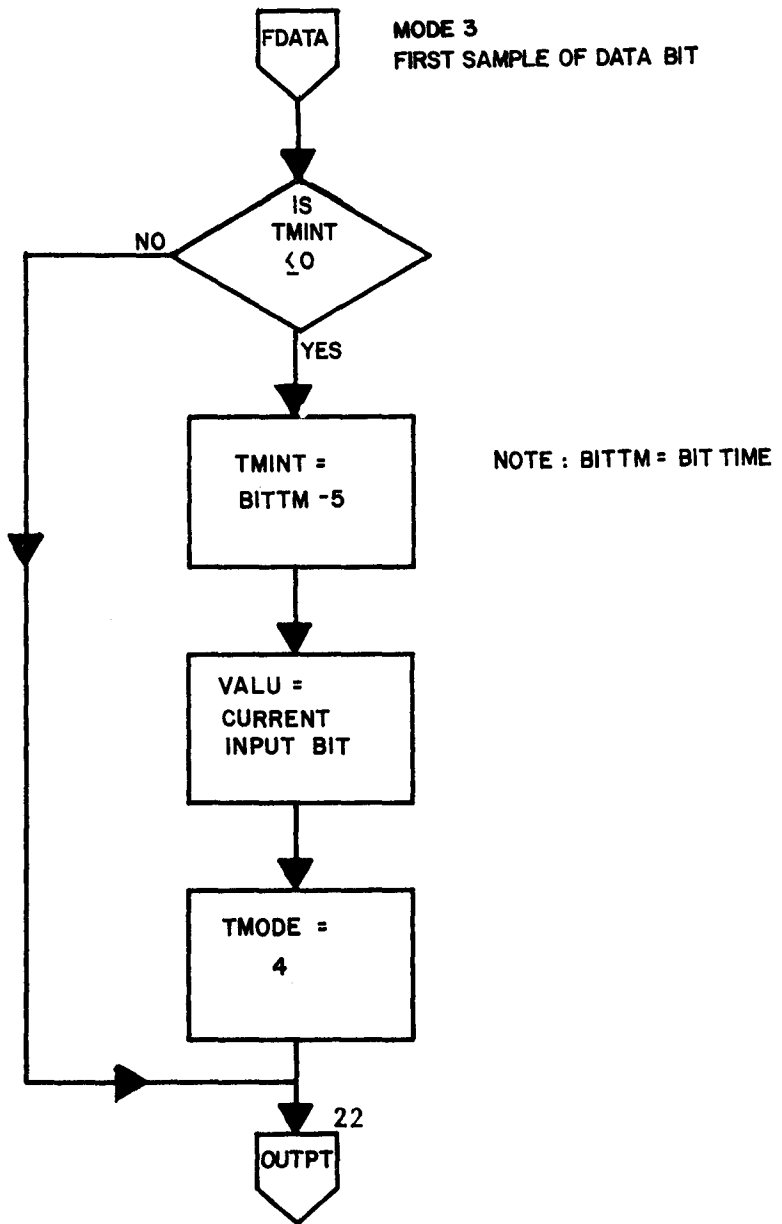


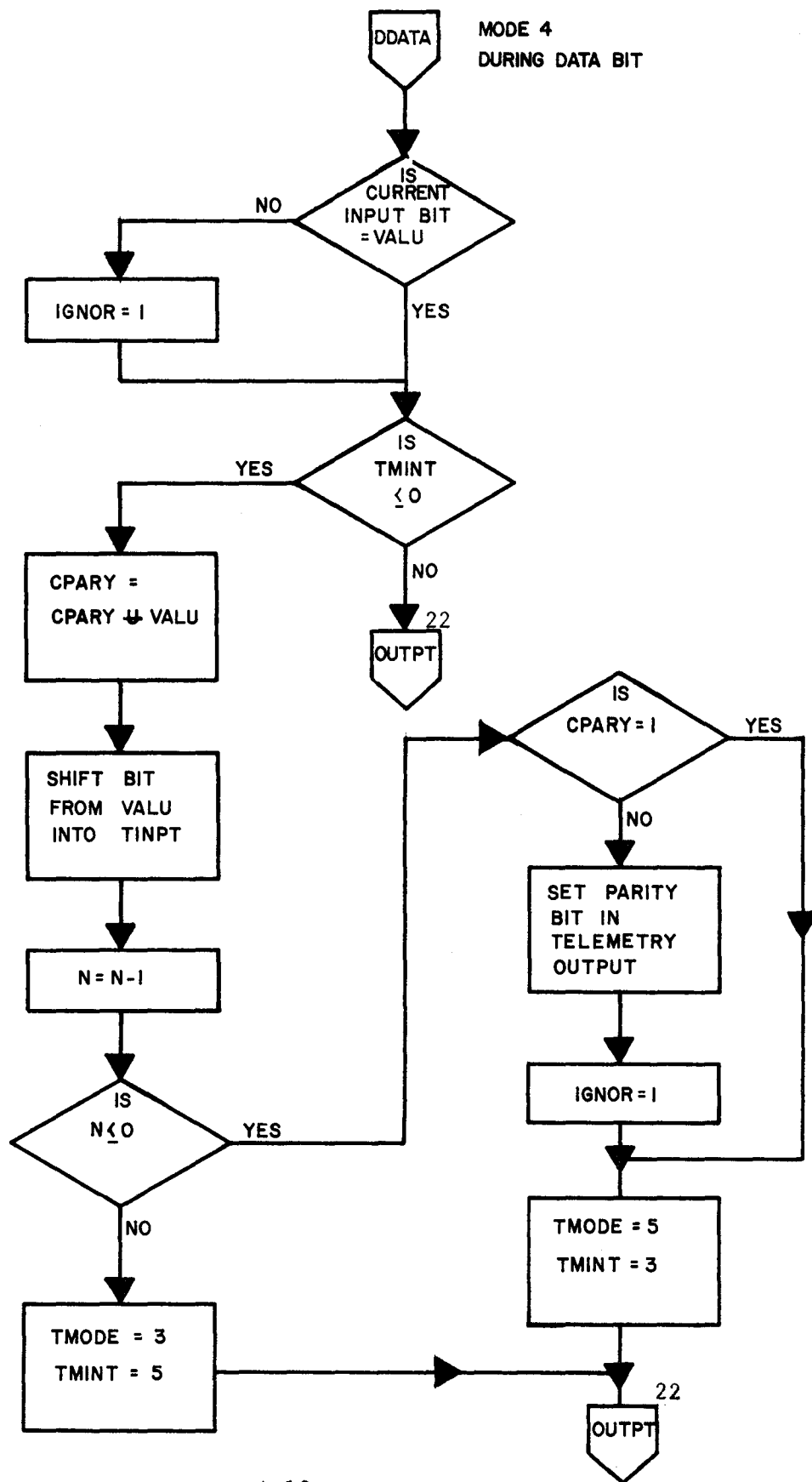


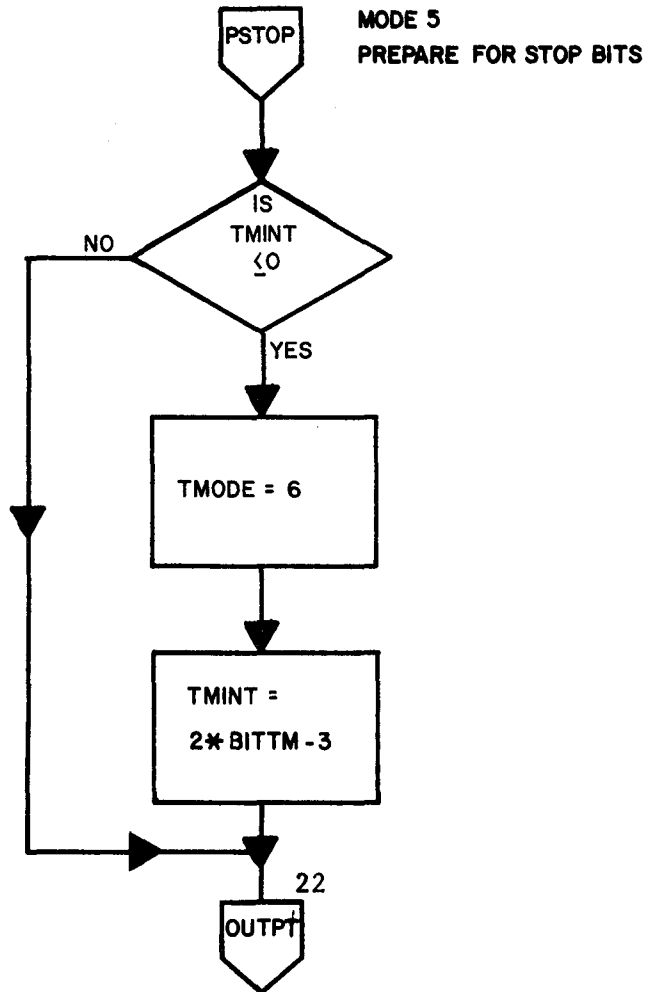


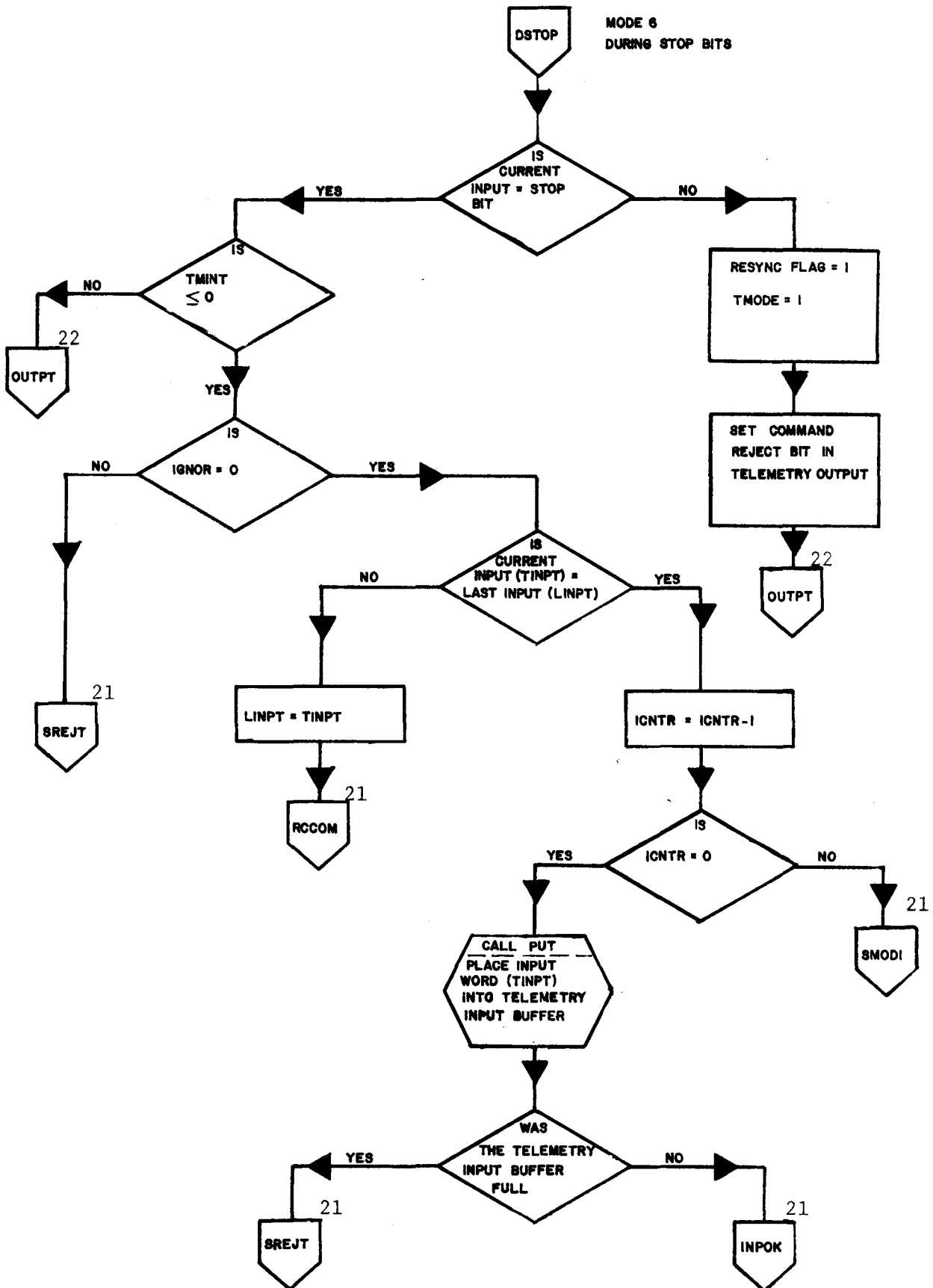


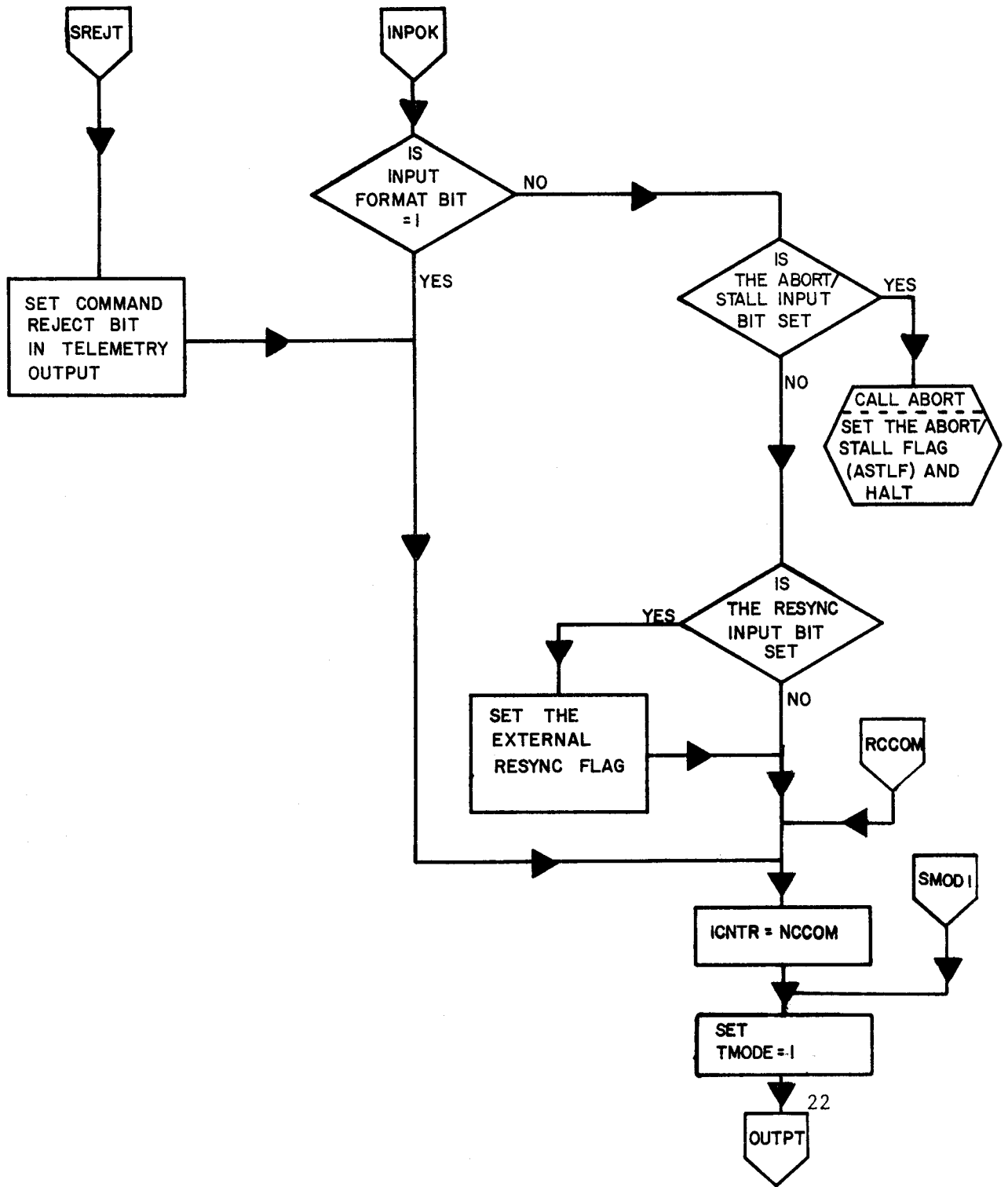
NOTE : N BITS = # OF DATA BITS



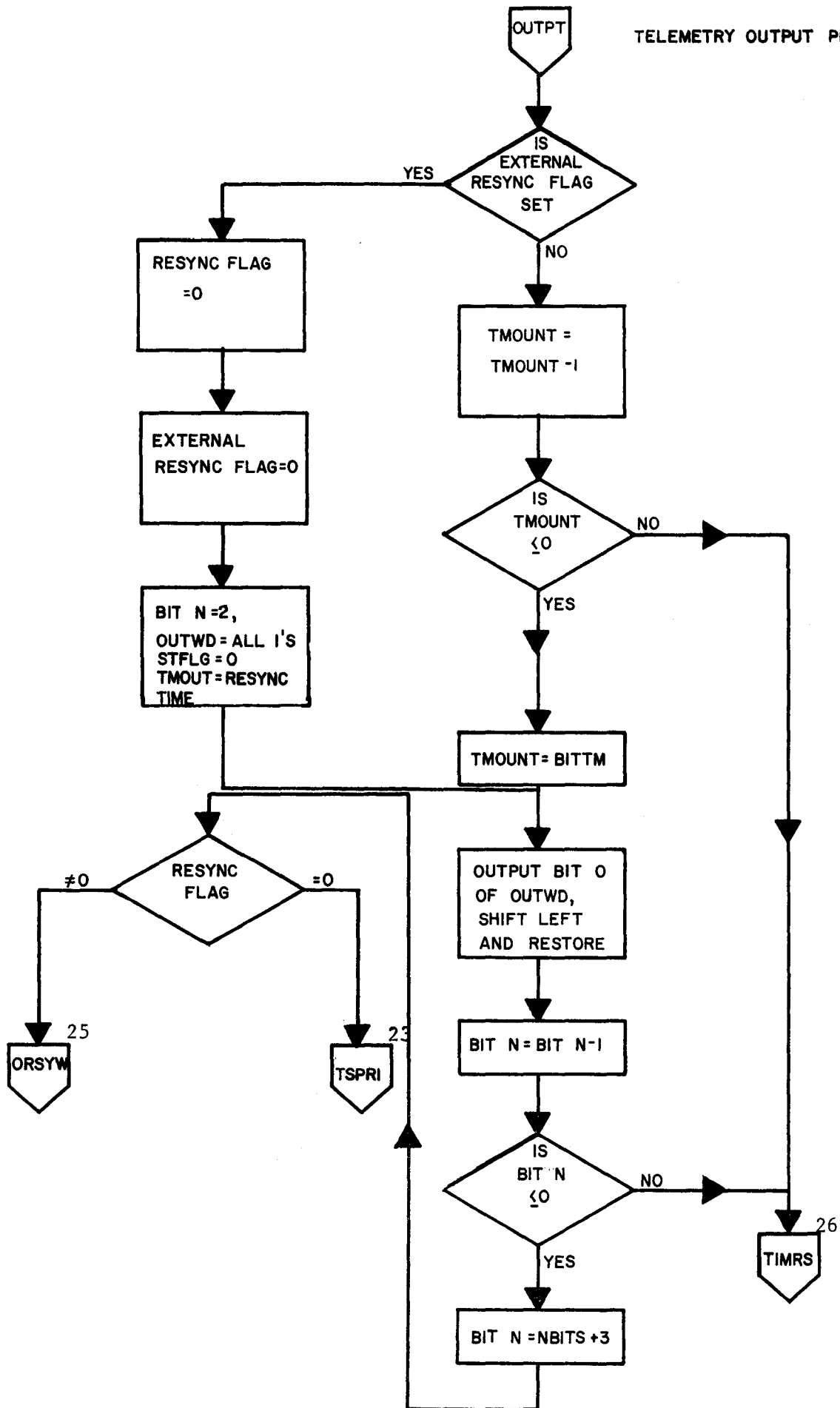


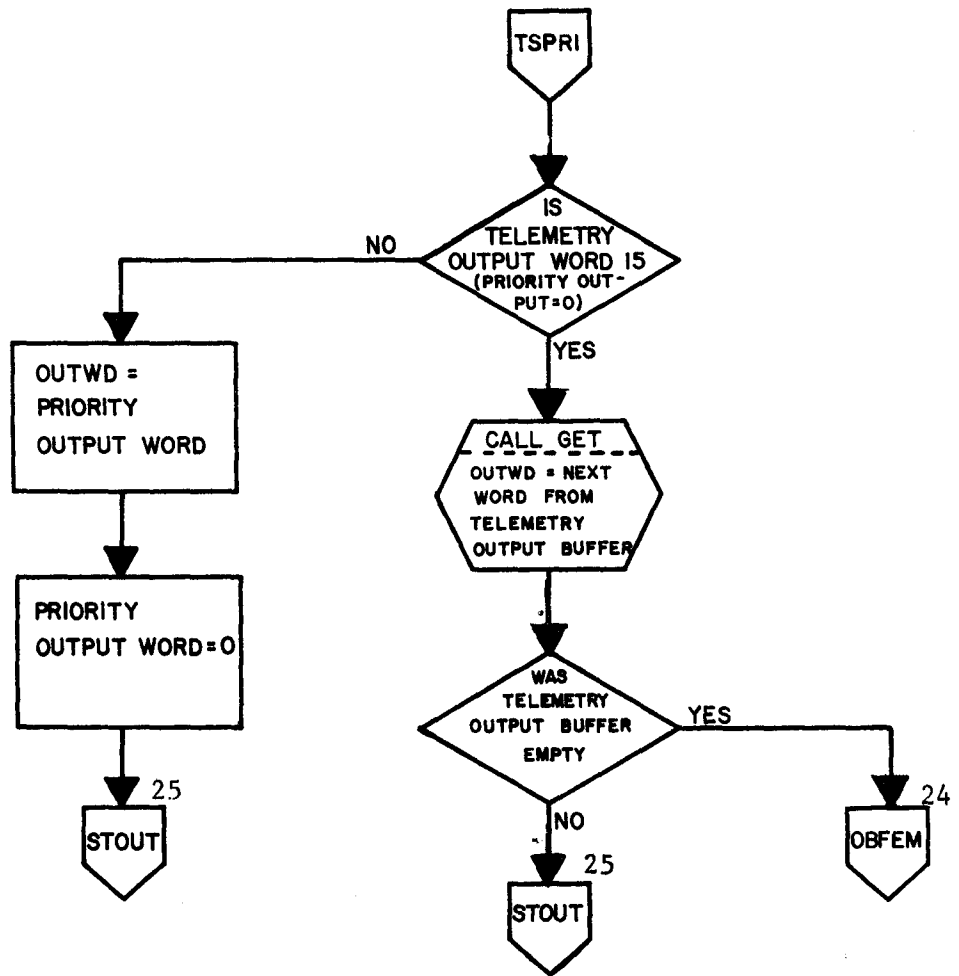


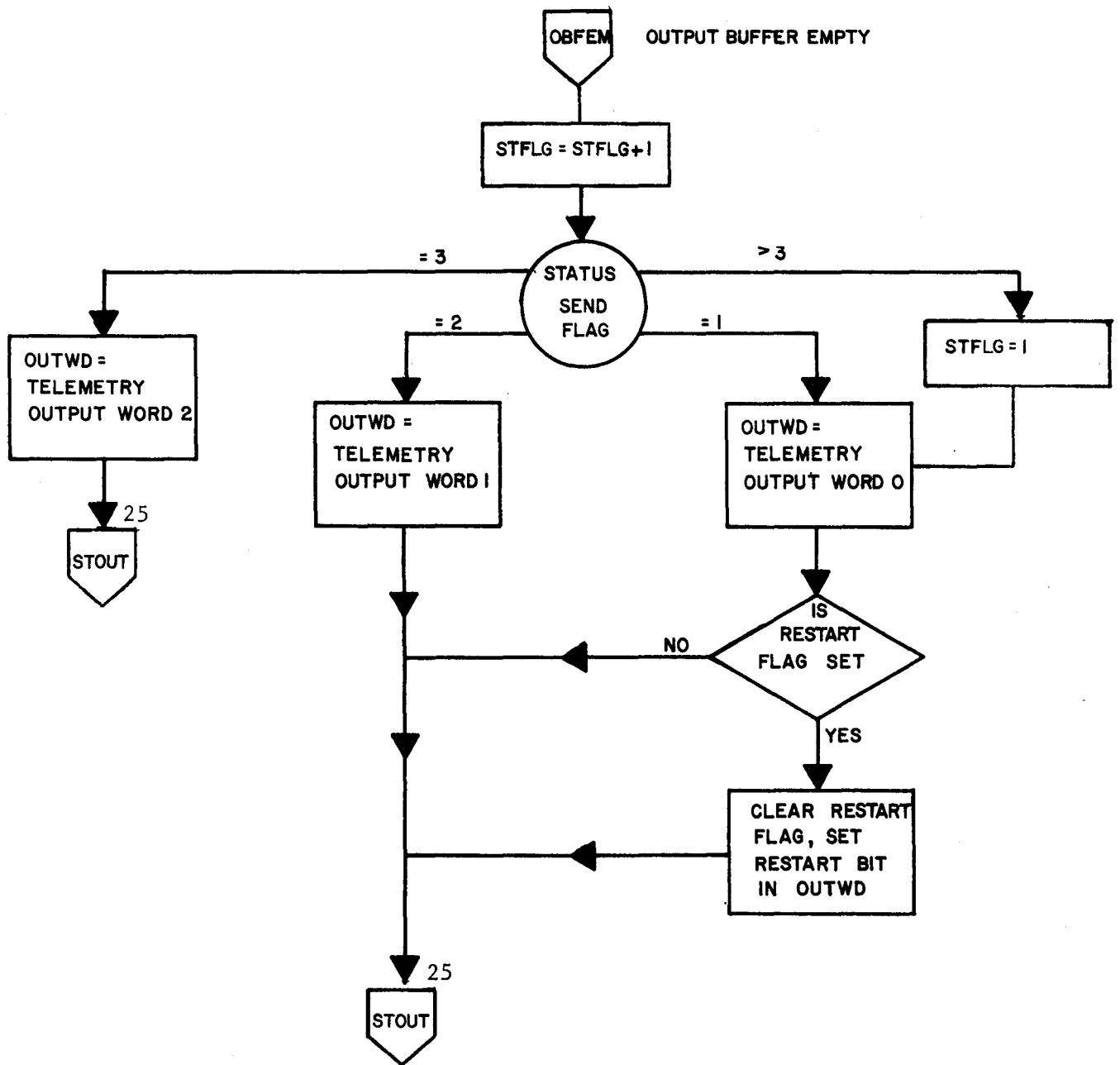


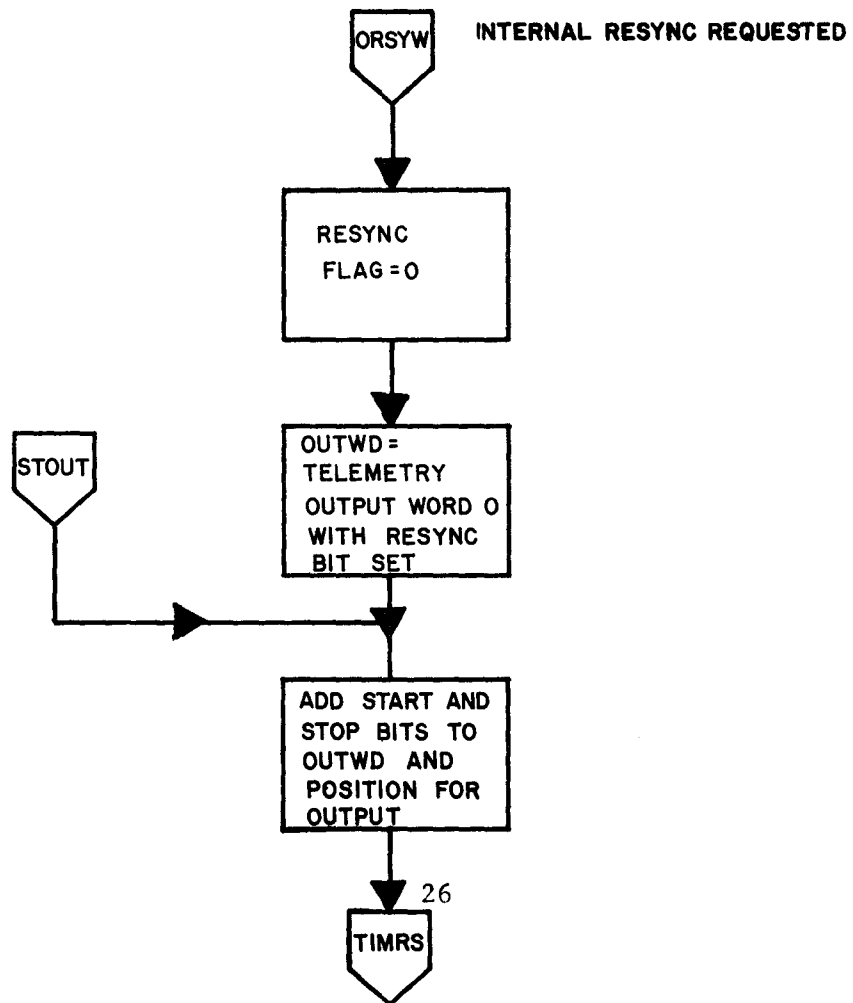


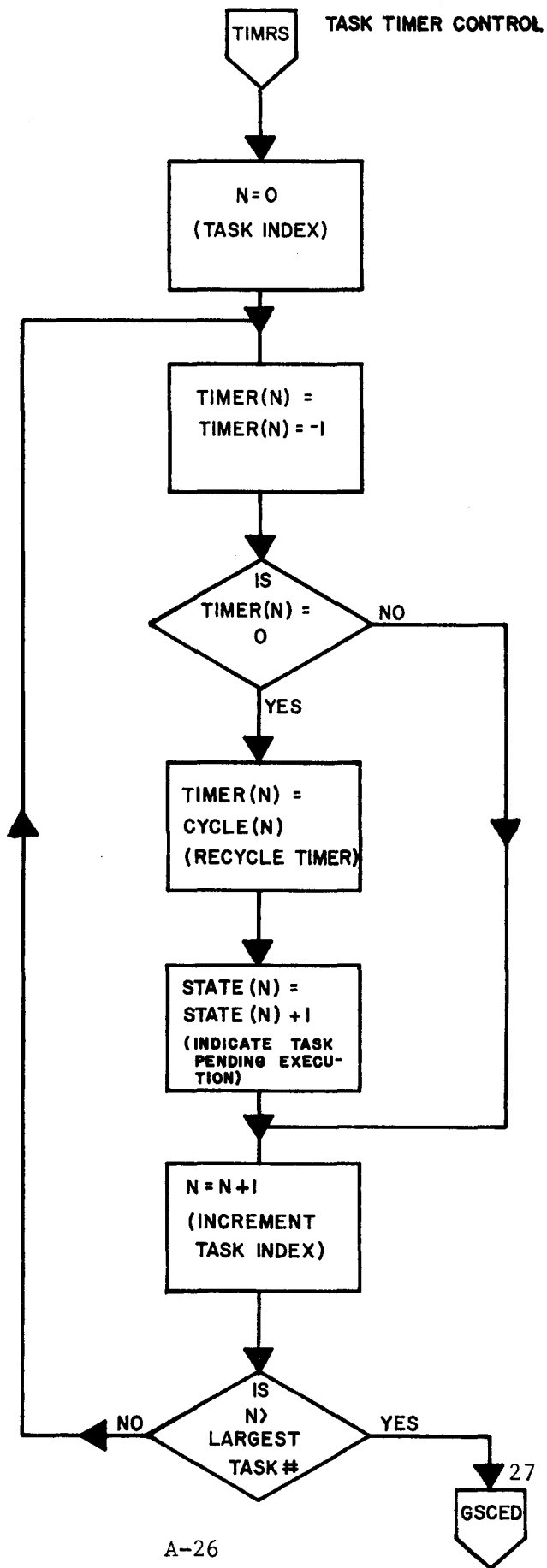
TELEMETRY OUTPUT PROCESSING

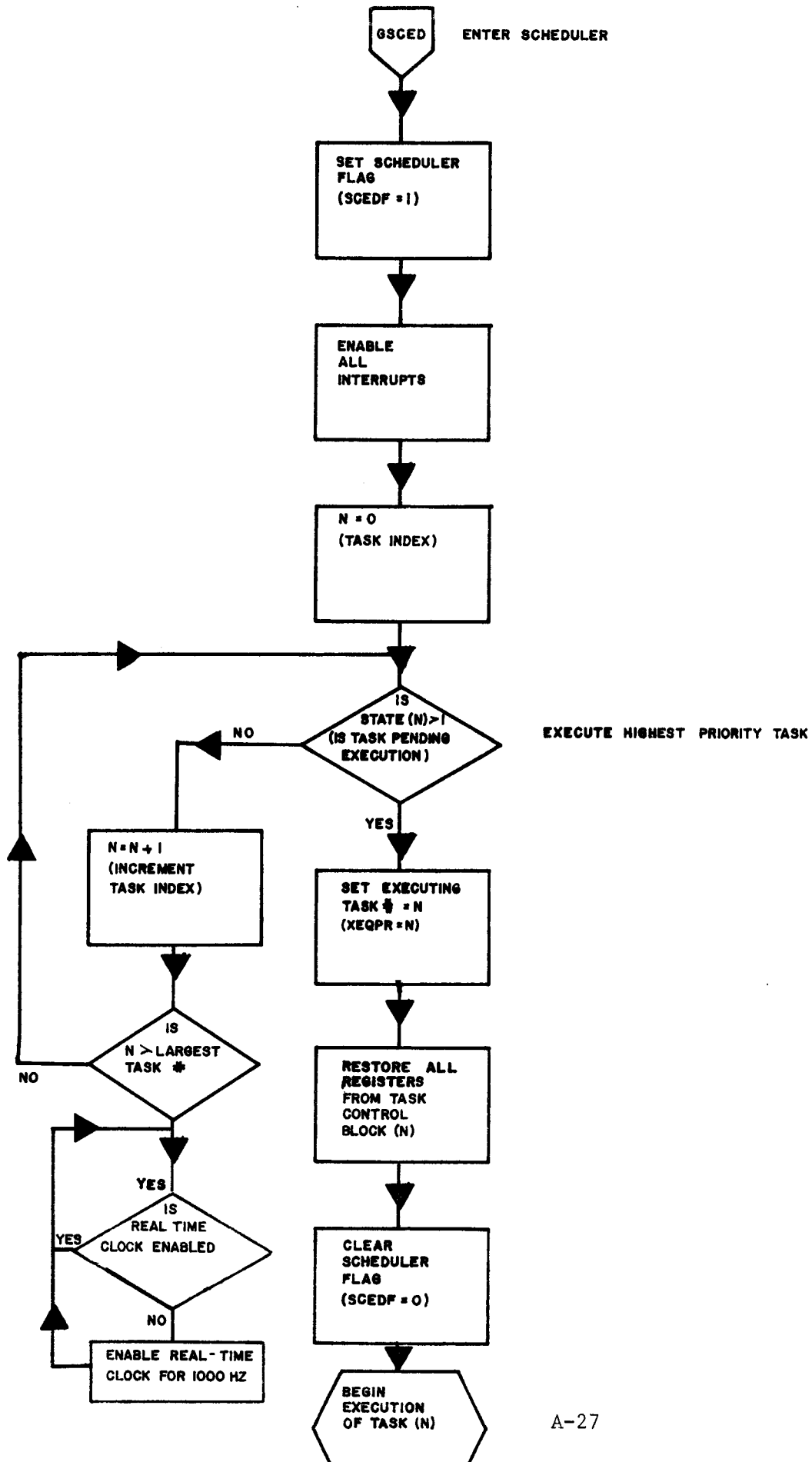










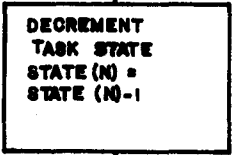
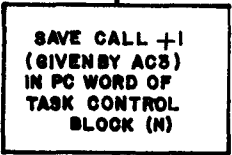
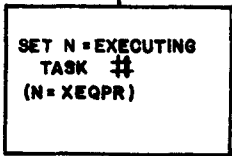


EXECUTE HIGHEST PRIORITY TASK

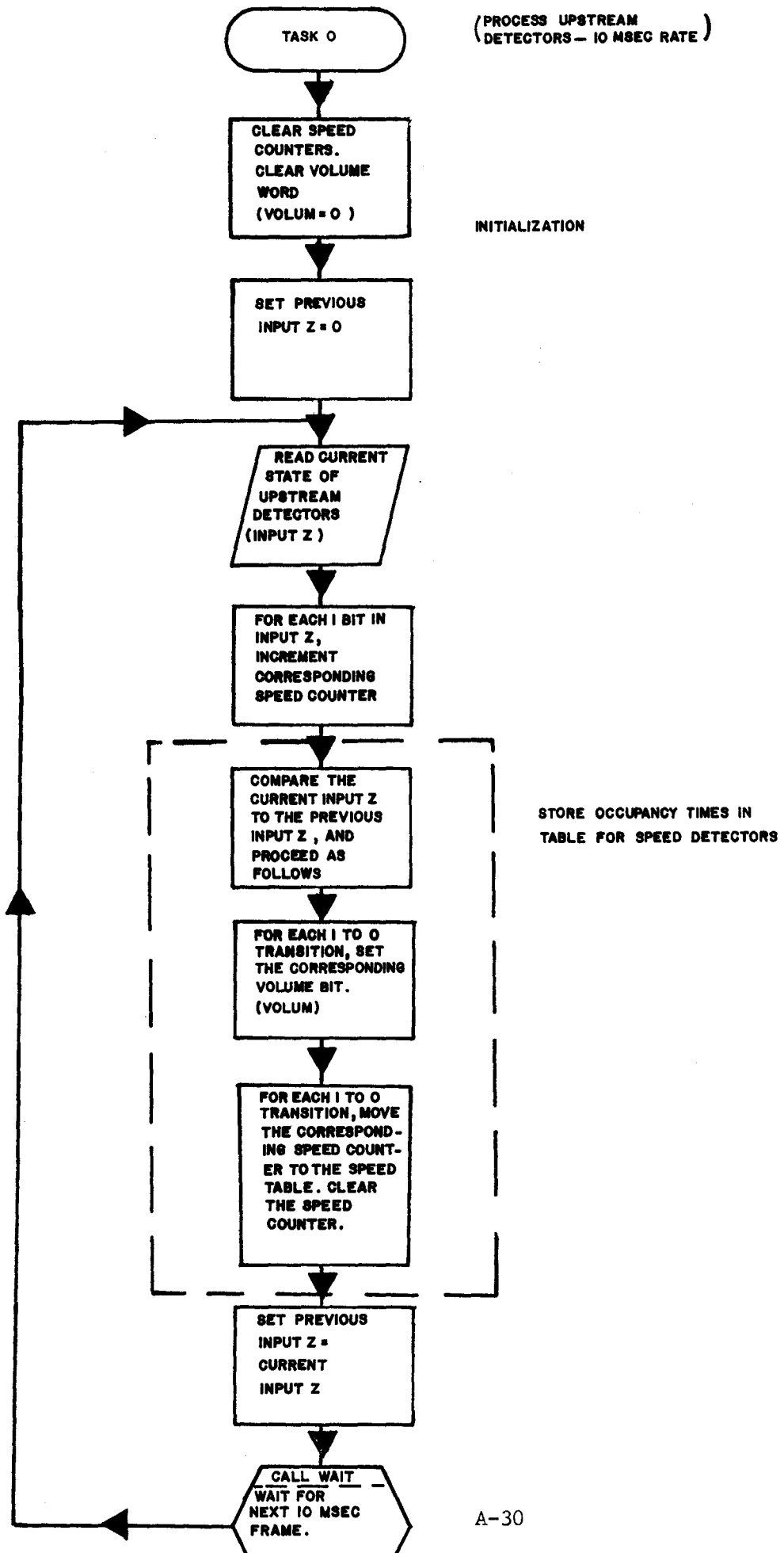


(TASK COMPLETION SUBROUTINE)

CALLED BY EACH TASK
AT COMPLETION OF EACH
TIME FRAME'S PROCESSING



PROGRAM SEGMENT 3



PROGRAM SEGMENT 4

TASK 1

(DETECTOR PROCESSING AND
TELEMETRY OUTPUT BUFFERING -
50 MSEC RATE

CLEAR THE FOLLOW-
ING: PREVIOUS Z+1
INPUT (PRZP1);
OUTPUT OCCUPANCY
WORD (OOCUP);
OUTPUT VOLUME
WORDS (OVOL, OVOL+1);
PREVIOUS Z+2
INPUT (PRZP 2)

INITIALIZATION

SET ALL
OCCUPANCY CELLS
(OCCUP) =
5 SEC. OVERFLOW
VALUE.

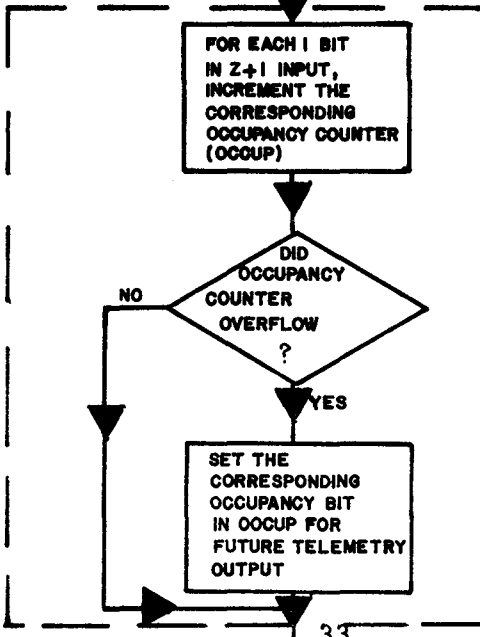
SET ALL
AVERAGE TRAP
TIME (AVTRP)
CELLS @ 30 MPH

DTLOP

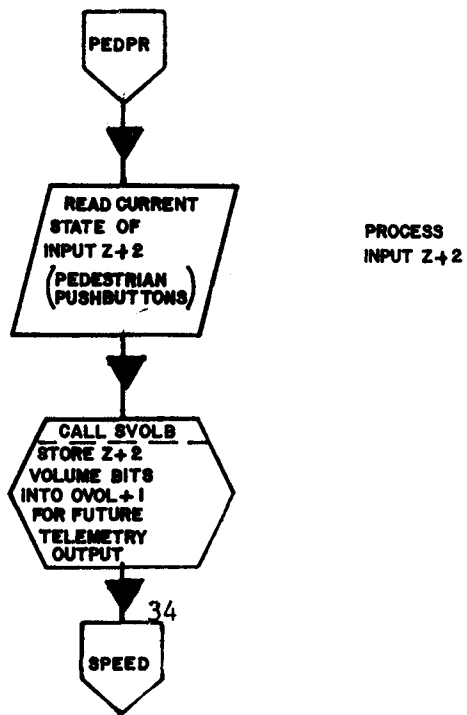
READ CURRENT
STATE OF
INPUT Z+1
(LOOP AND
MISCELLANEOUS
DETECTORS

PROCESS INPUT Z+1

CALL SVOLB
STORE Z+1
VOLUME BITS
INTO OVOL, OVOL+1
FOR FUTURE
TELEMETRY
OUTPUT



33
PEDPR



SPEED

CALCULATE AVERAGE SPEEDS OVER UPSTREAM DETECTORS BY USING DYNAMIC LAMBDA AVERAGING TECHNIQUE ON THE OCCUPANCY TIMES STORED BY TASK 0

SET N = VOLUME BITS FROM TASK 0 (VOLUM)
CLEAR VOLUM

PERFORM THE FOLLOWING FOR EACH BIT SET IN N :

SET THE CORRESPONDING VOLUME BIT IN OVOL FOR FUTURE TELEMETRY OUTPUT

SET TPTIM = CORRESPONDING ENTRY IN SPEED TABLE FOR THIS BIT

DOES TPTIM INDICATE A SPEED BETWEEN .7 AND 60 MPH

IS AVTRP + 30 > TPTIM FOR THIS DETECTOR

SET T = AVTRP + 15

IS T > 128

LAMB = T

YES

SET T = TPTIM

YES

LAMB = 128

$$AVTRP = AVTRP + \frac{(T - AVTRP) \cdot LAMB}{128}$$

NOTE: AVTRP IS THE AVERAGE TRAP TIME FOR 2 OR 3 ADJACENT LANE MAGNETOMETERS

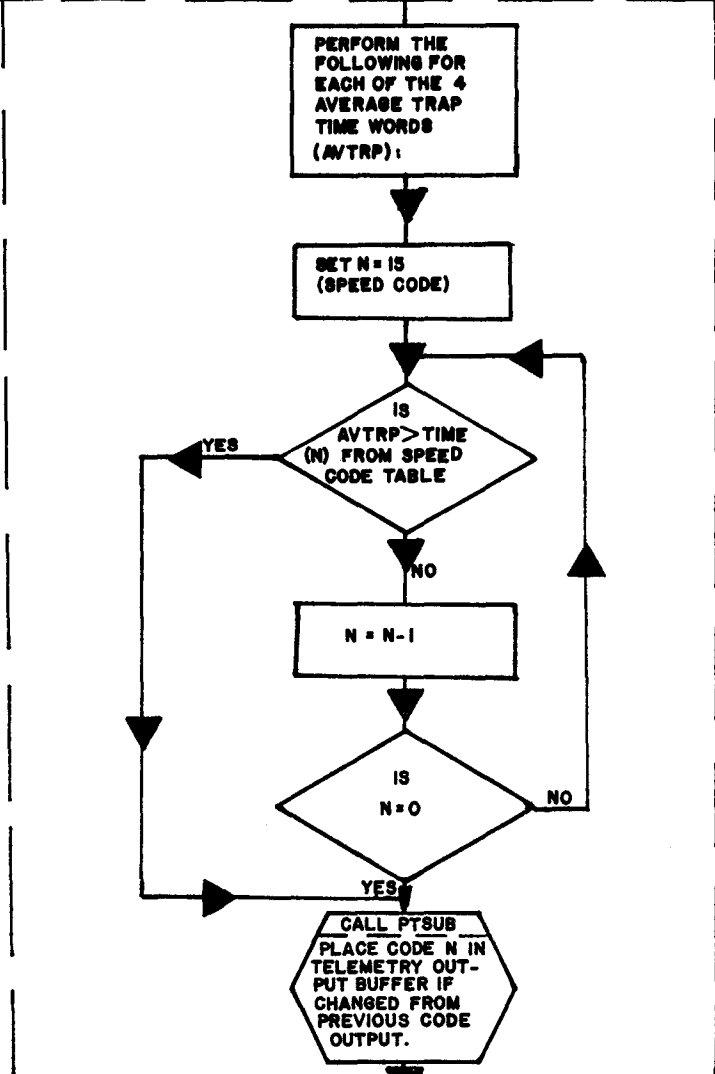
OUTBF

CALL PTMDT
PLACE VOLUME
BITS FROM OVOL
INTO TELEMETRY
OUTPUT BUFFER.
CLEAR OVOL.

PLACE VOLUME AND OCCUPANCY
BITS IN TELEMETRY OUTPUT
BUFFER.

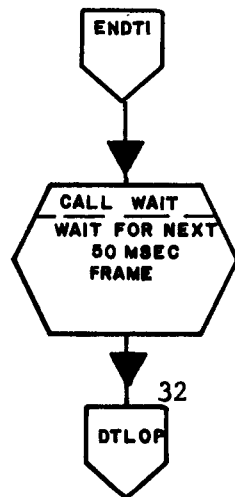
CALL PTMDT
PLACE VOLUME
BITS FROM OVOL-H
INTO TELEMETRY
OUTPUT BUFFER
CLEAR OVOL-H

CALL PTMDT
PLACE OCCUPANCY
BITS FROM OCCUP
INTO TELEMETRY
OUTPUT BUFFER.
CLEAR OCCUP

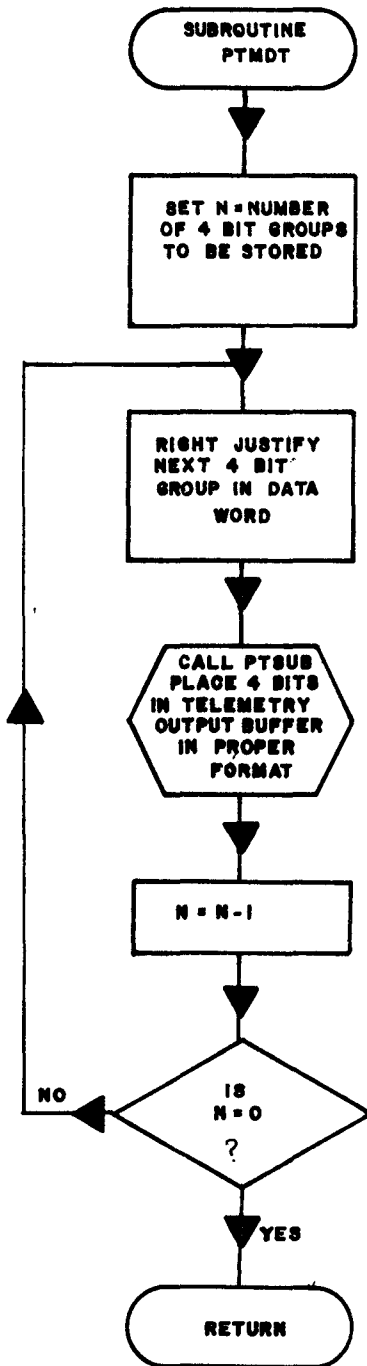


CONVERT AVERAGE TO 4
BIT CODES AND PLACE IN
TELEMETRY OUTPUT BUFFER.

ENDT I



TASK 1 SUBROUTINE TO
PLACE DATA BITS IN
TELEMETRY OUTPUT
BUFFER



SUBROUTINE
SVOLB

TASK 1 SUBROUTINE FOR
STORING VOLUME BITS

COMPARE THE
CURRENT DETECTOR
INPUT TO THE
PREVIOUS INPUT

FOR EACH 0 TO I
TRANSITION BIT,
SET THE CORRESPOND
ING VOLUME BIT IN
OVOL OR OVOL + 1

SET PREVIOUS
INPUT VALUE =
CURRENT INPUT

RETURN

PROGRAM SEGMENT 5

TASK 2

PROCESS TRAFFIC SIGNAL LIGHTS - 100 MSEC RATE



INITIALIZATION

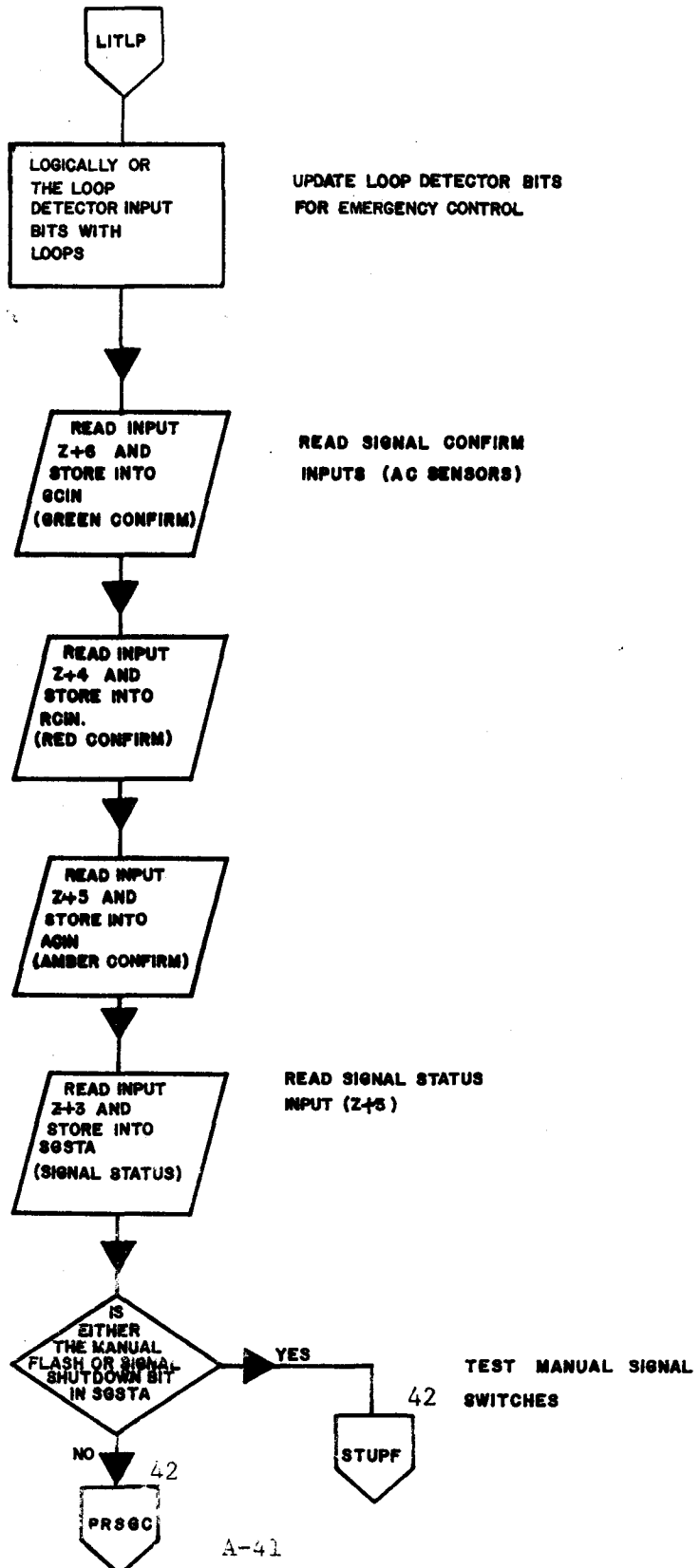
CLEAR THE FOLLOWING VARIABLES:
LOOPS (LOOP DETECTOR CALLS)
GOUTP (LAST GREEN AND)
ROUTP (RED OUTPUT BITS)
LAKGO (LAST ACKNOWLEDGED FORMAT 0 TELEMETRY INPUT)
LAKGI (LAST ACKNOWLEDGED FORMAT 1 TELEMETRY INPUT)
ENMOD (EMERGENCY CONTROL MODE FLAG)
LRJCO (LAST REJECTED FORMAT 0 TELEMETRY INPUT)
LRJCI (LAST REJECTED FORMAT 1 TELEMETRY INPUT)
COMCD (SIGNAL COMMAND WORD)
CNPER (SIGNAL CONFIRM ERROR FLAG)
CNFLS (NUMBER OF SIGNAL CONFIRM SERIOUS ERRORS)
LPRI0 (LAST PRIORITY TELEMETRY OUTPUT VALUE)
PRAER (AMBER ERROR BITS)
PRER (RED ERROR BITS)
PRGR (GREEN ERROR BITS)
LSAER (LAST AMBER ERROR BITS)
LSRER (LAST RED ERROR BITS)
LSGER (LAST GREEN ERROR BITS)

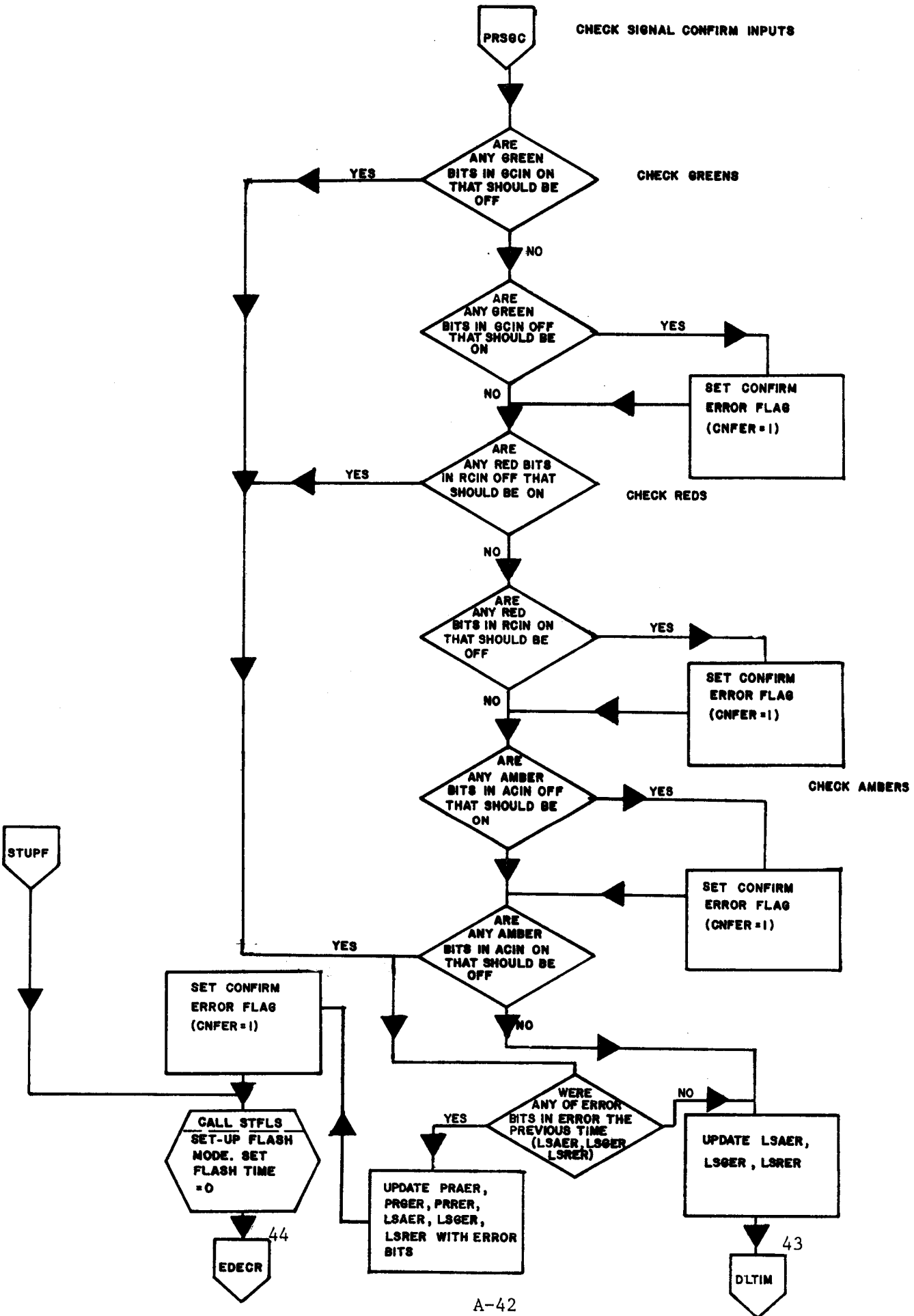
SET LAST AMBER
OUTPUT = FLASH BIT
AOUTP = 1

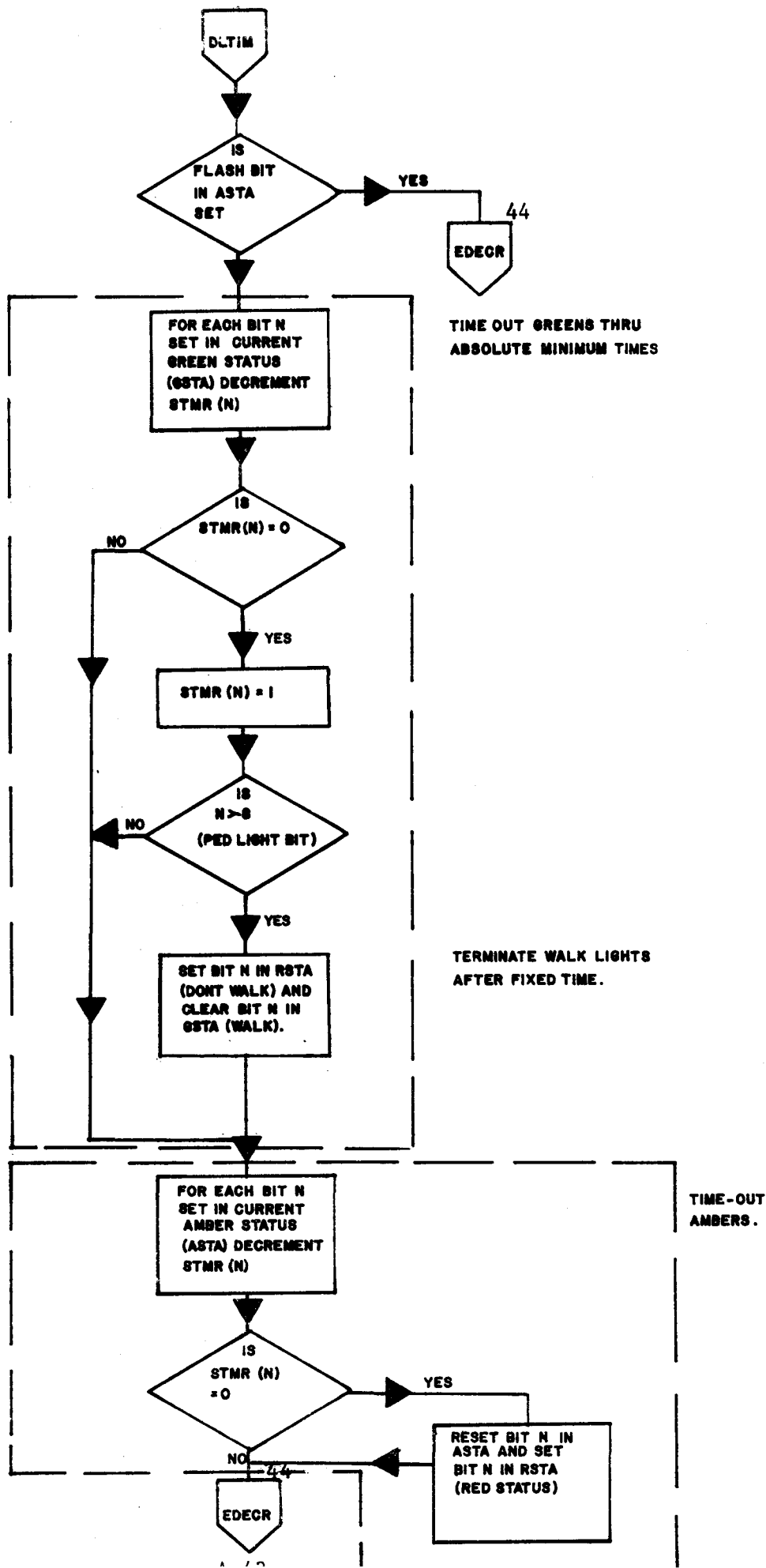
SET TELEMETRY
INPUT ERROR
TIMER = 1
(ERTIN)

41









DLTIM

IS
FLASH BIT
IN ASTA
SET

YES

44

EDECR

FOR EACH BIT N
SET IN CURRENT
GREEN STATUS
(GSTA) DECREMENT
STMR (N)

TIME OUT GREENS THRU
ABSOLUTE MINIMUM TIMES

IS
STMR(N) = 0

NO

YES

STMR (N) = 1

IS
N > 8
(PED LIGHT BIT)

NO

YES

SET BIT N IN RSTA
(DONT WALK) AND
CLEAR BIT N IN
GSTA (WALK).

TERMINATE WALK LIGHTS
AFTER FIXED TIME.

FOR EACH BIT N
SET IN CURRENT
AMBER STATUS
(ASTA) DECREMENT
STMR (N)

TIME-OUT
AMBERS.

IS
STMR (N)
= 0

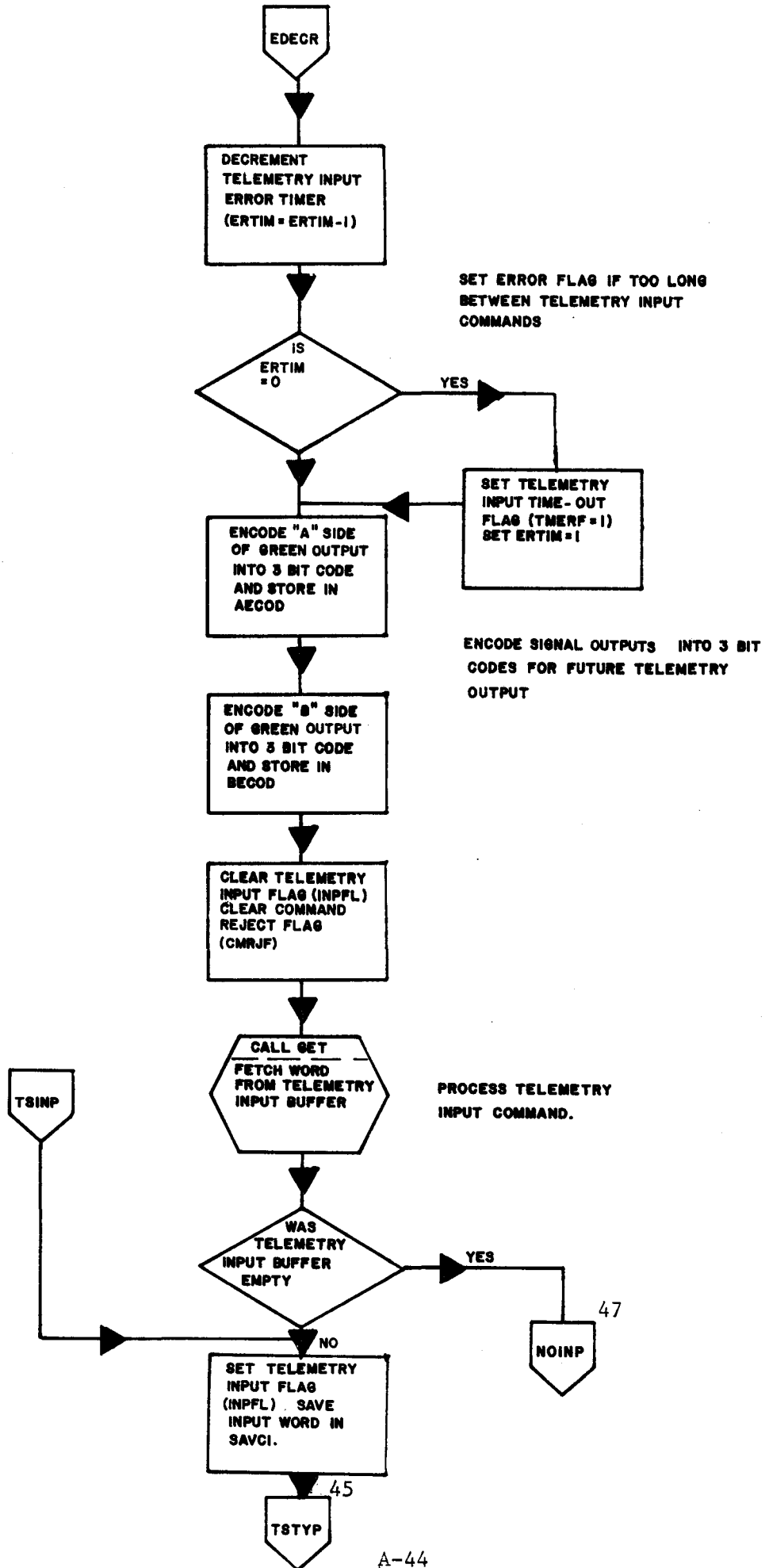
YES

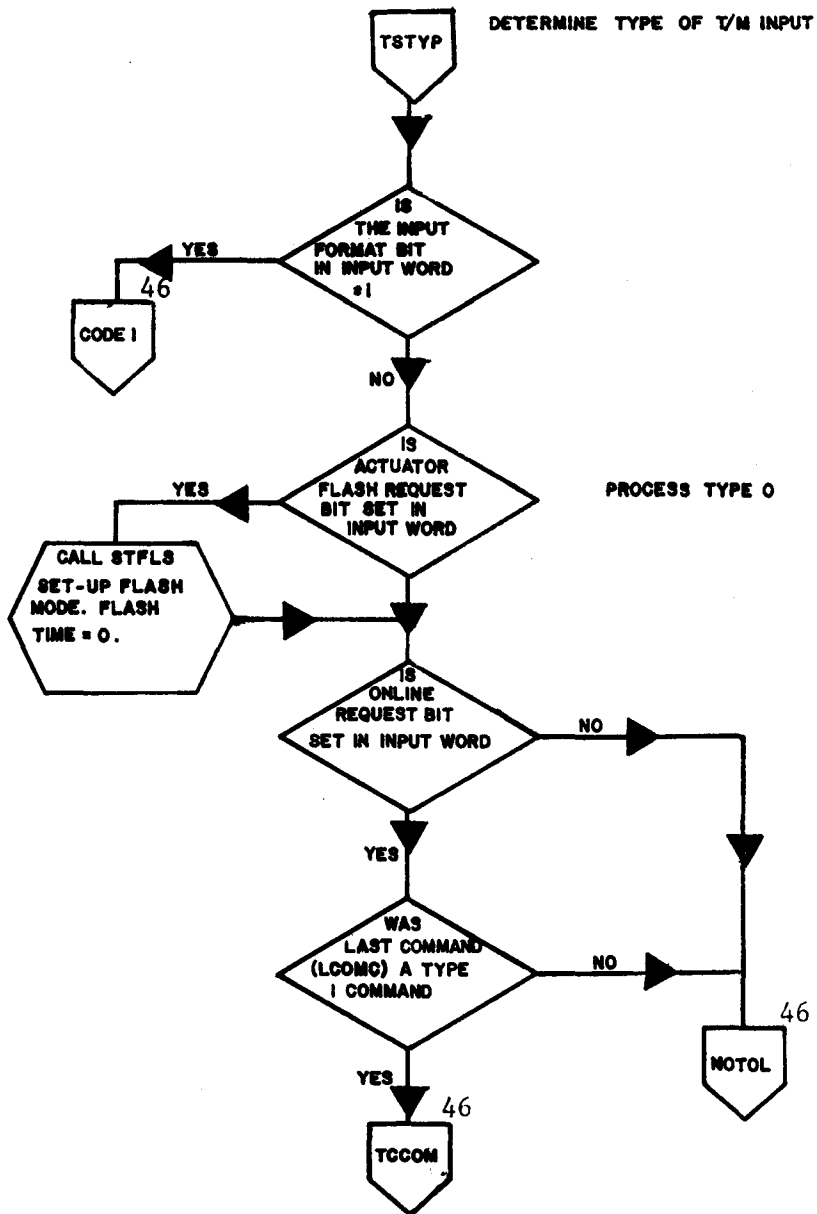
RESET BIT N IN
ASTA AND SET
BIT N IN RSTA
(RED STATUS)

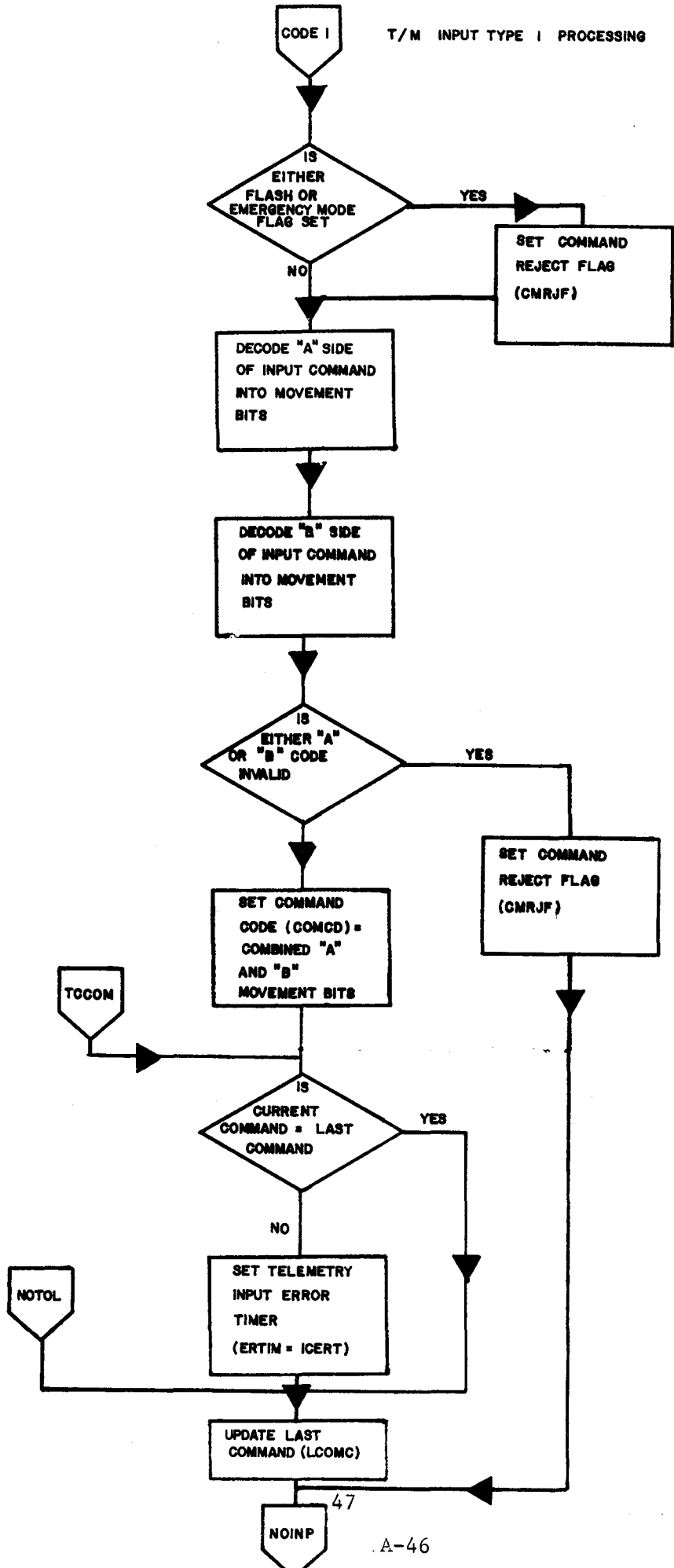
NO

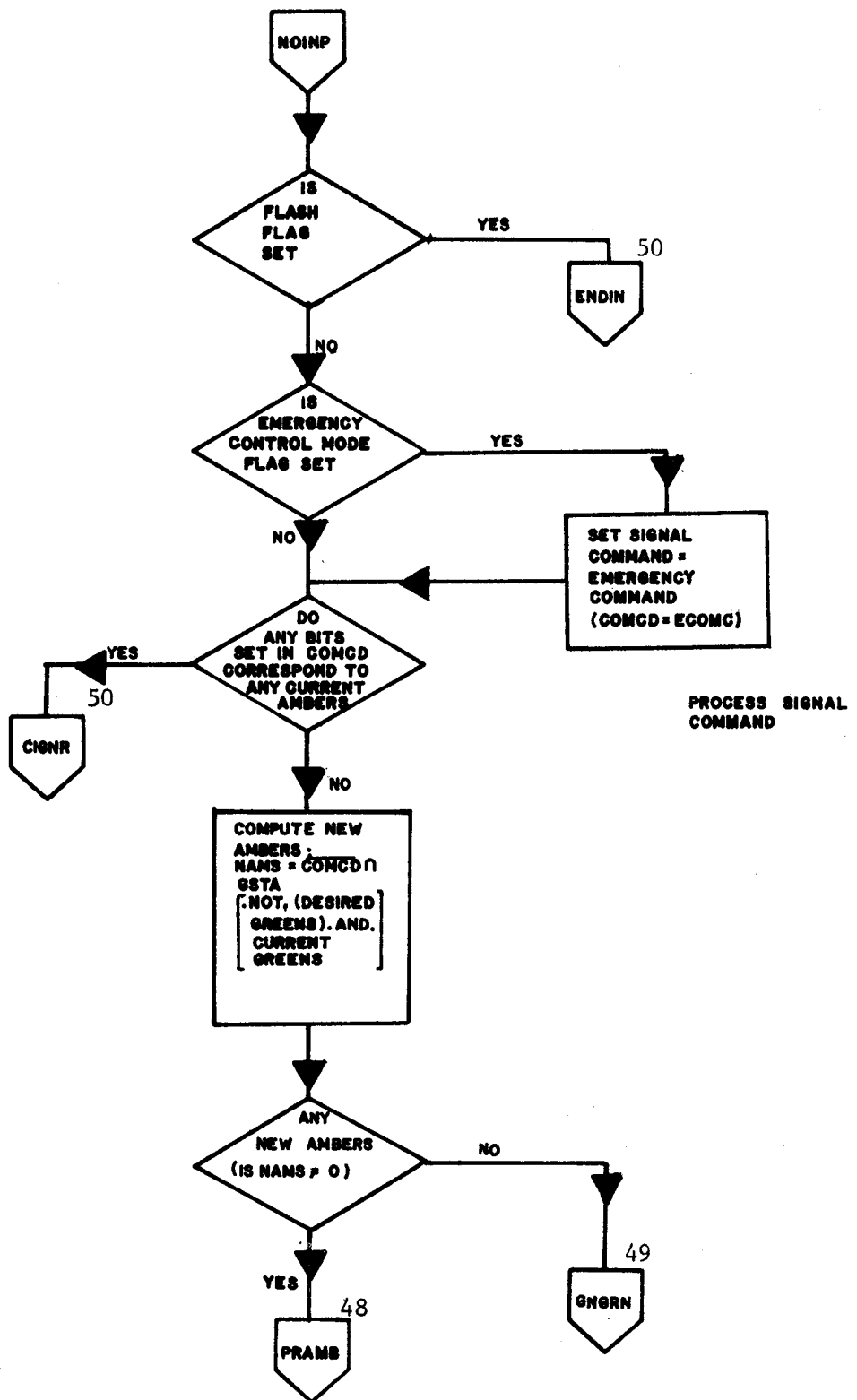
EDECR

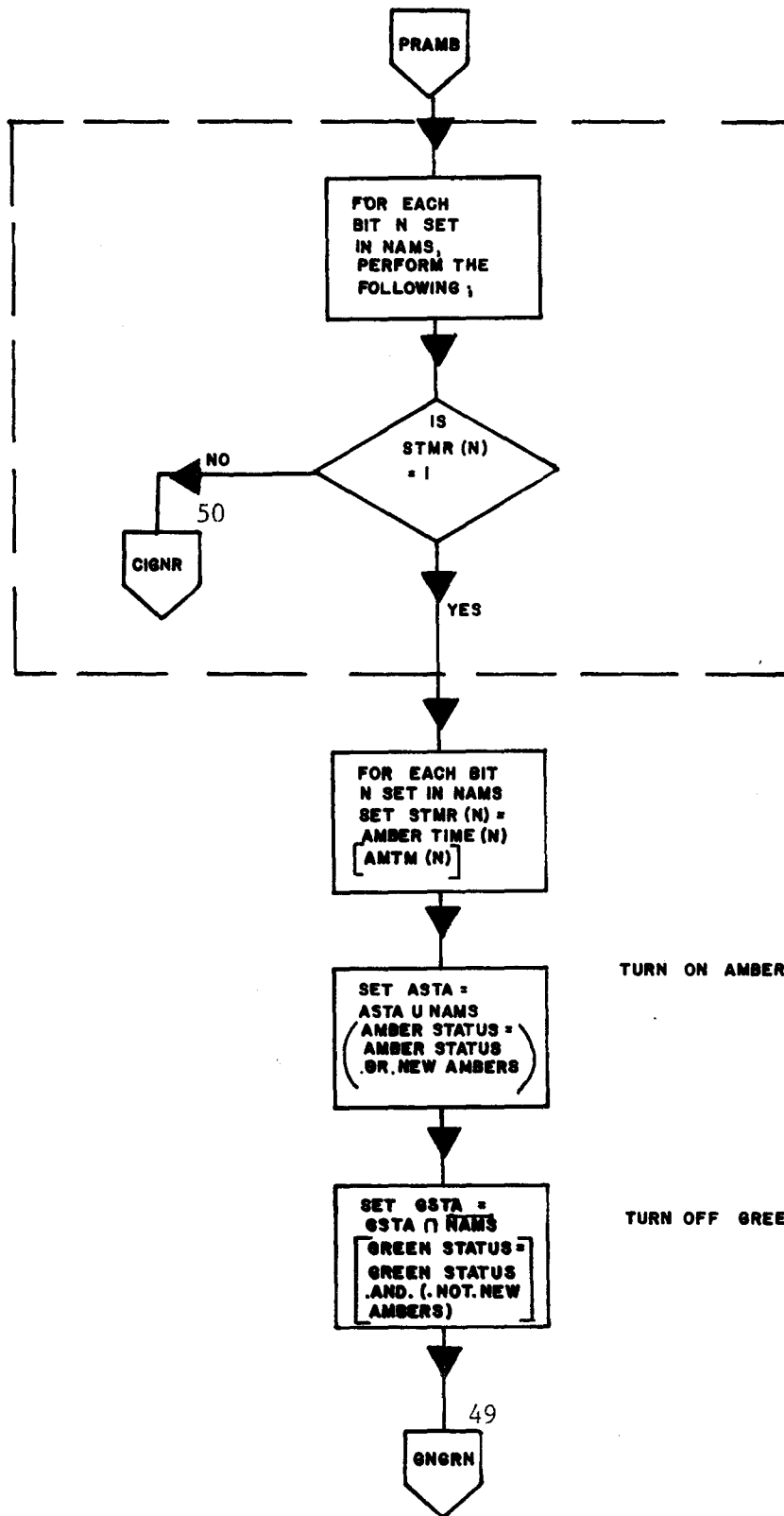
44







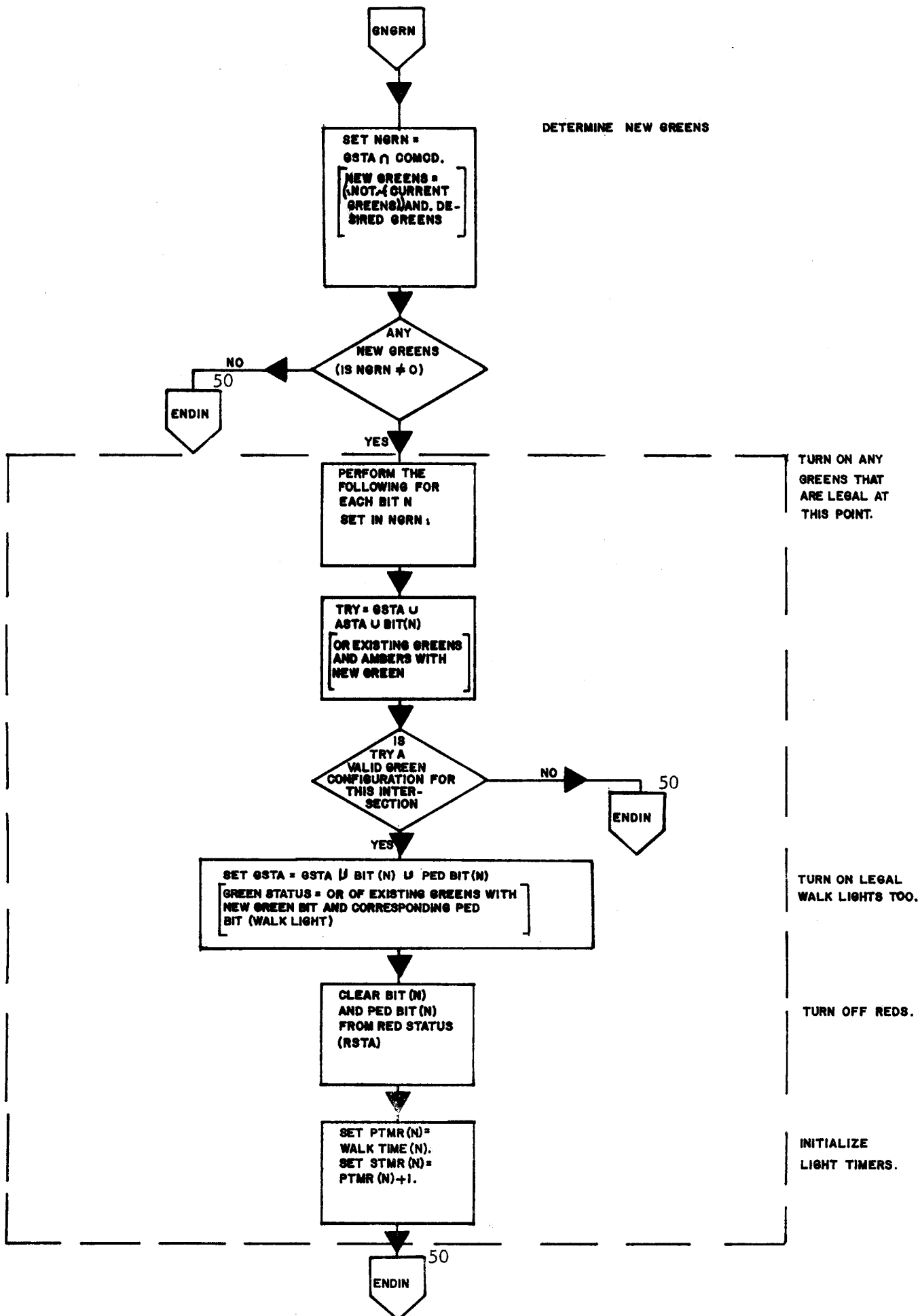




MAKE SURE ENDING GREENS
HAVE BEEN ON FOR THE
REQUIRED ABSOLUTE
MINIMUM TIME.

TURN ON AMBERS

TURN OFF GREENS



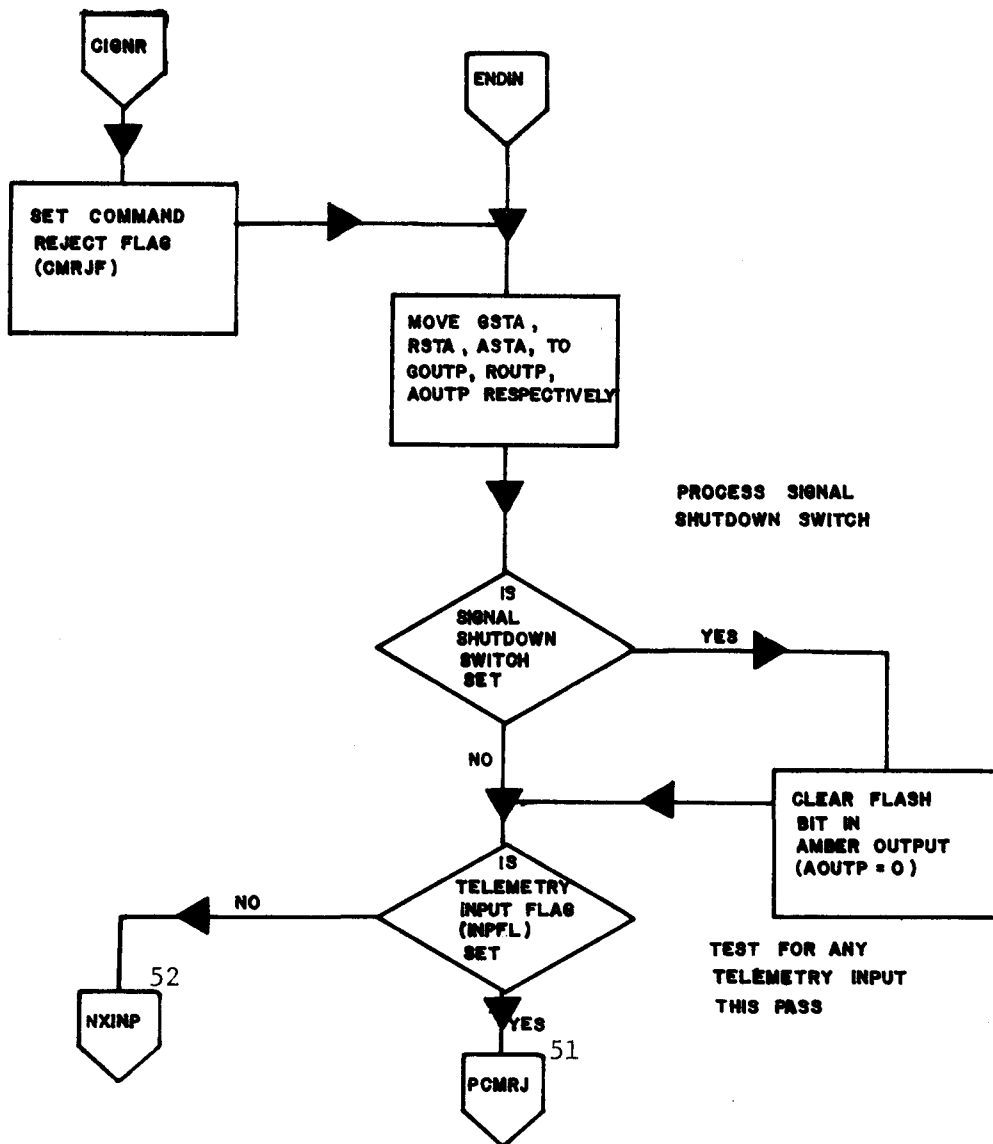
DETERMINE NEW GREENS

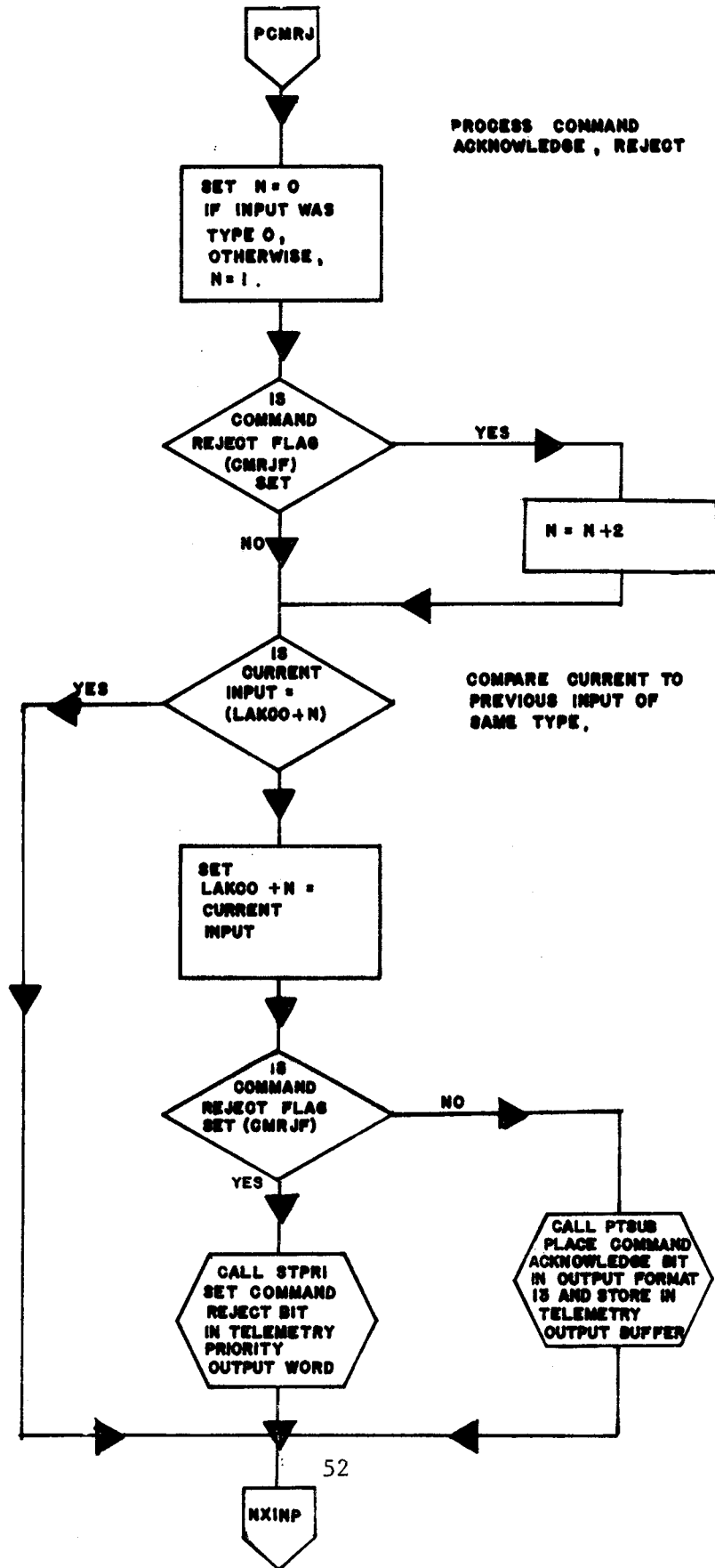
TURN ON ANY GREENS THAT ARE LEGAL AT THIS POINT.

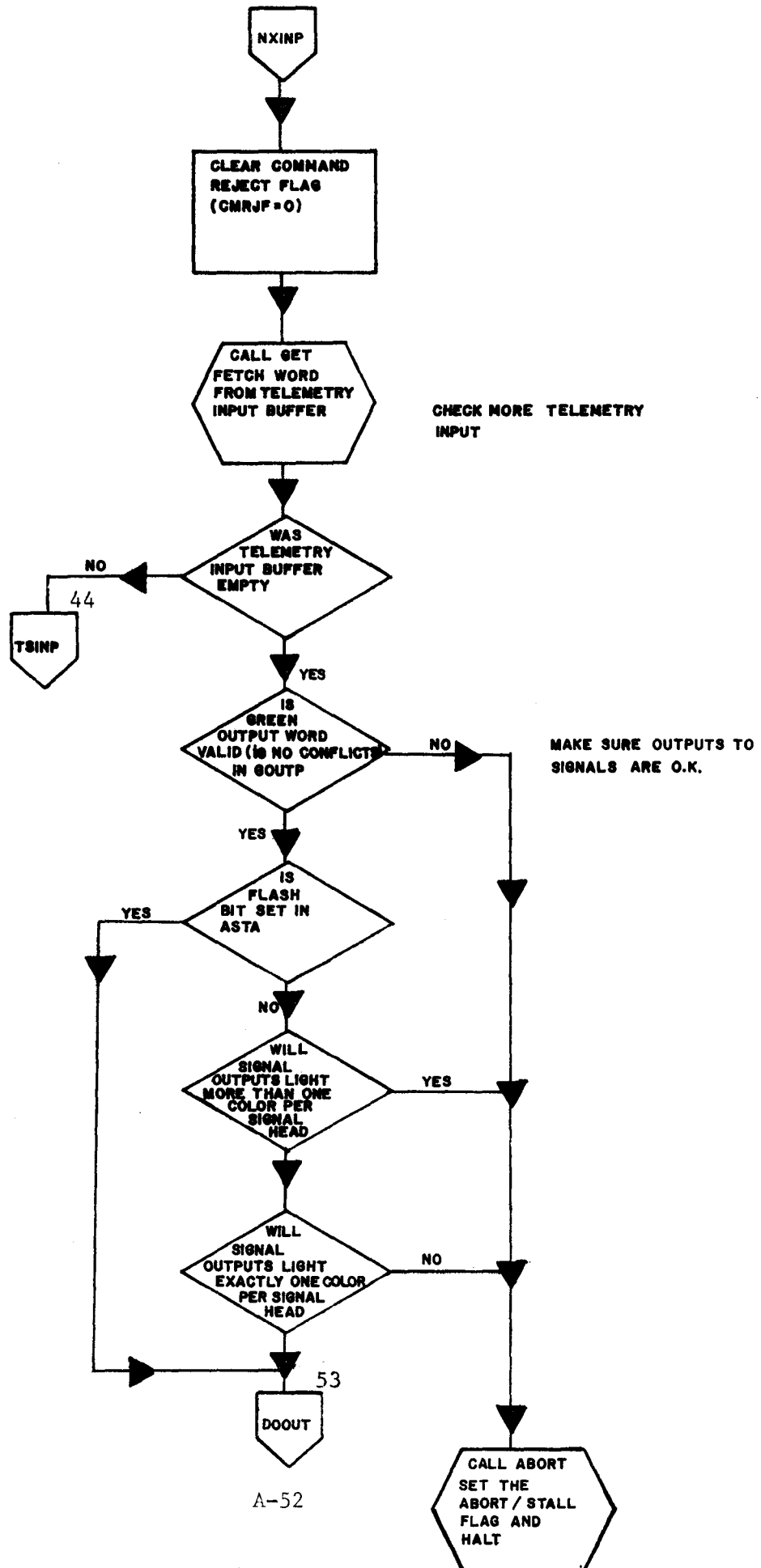
TURN ON LEGAL WALK LIGHTS TOO.

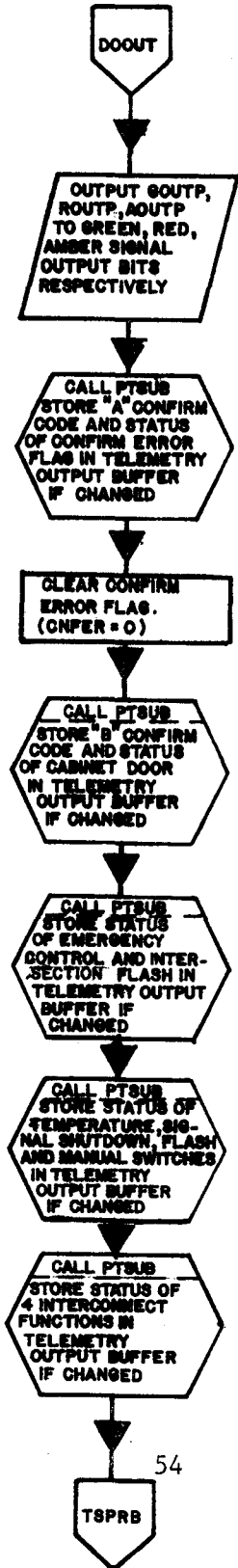
TURN OFF REDS.

INITIALIZE LIGHT TIMERS.



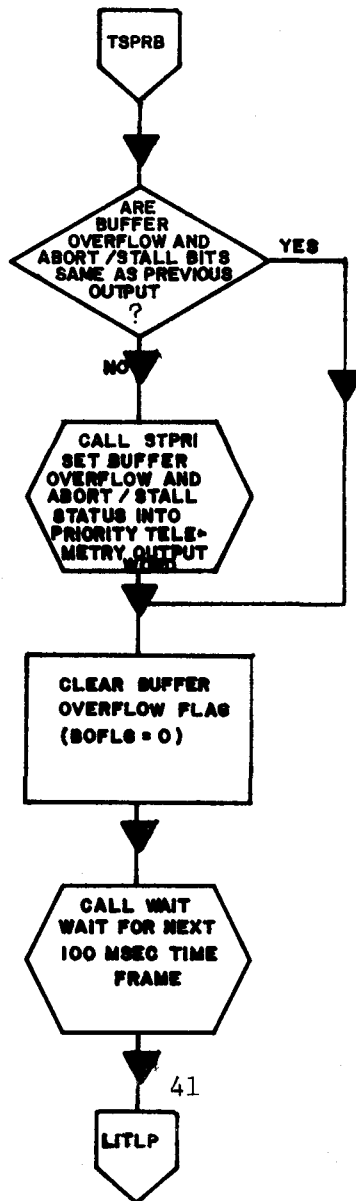




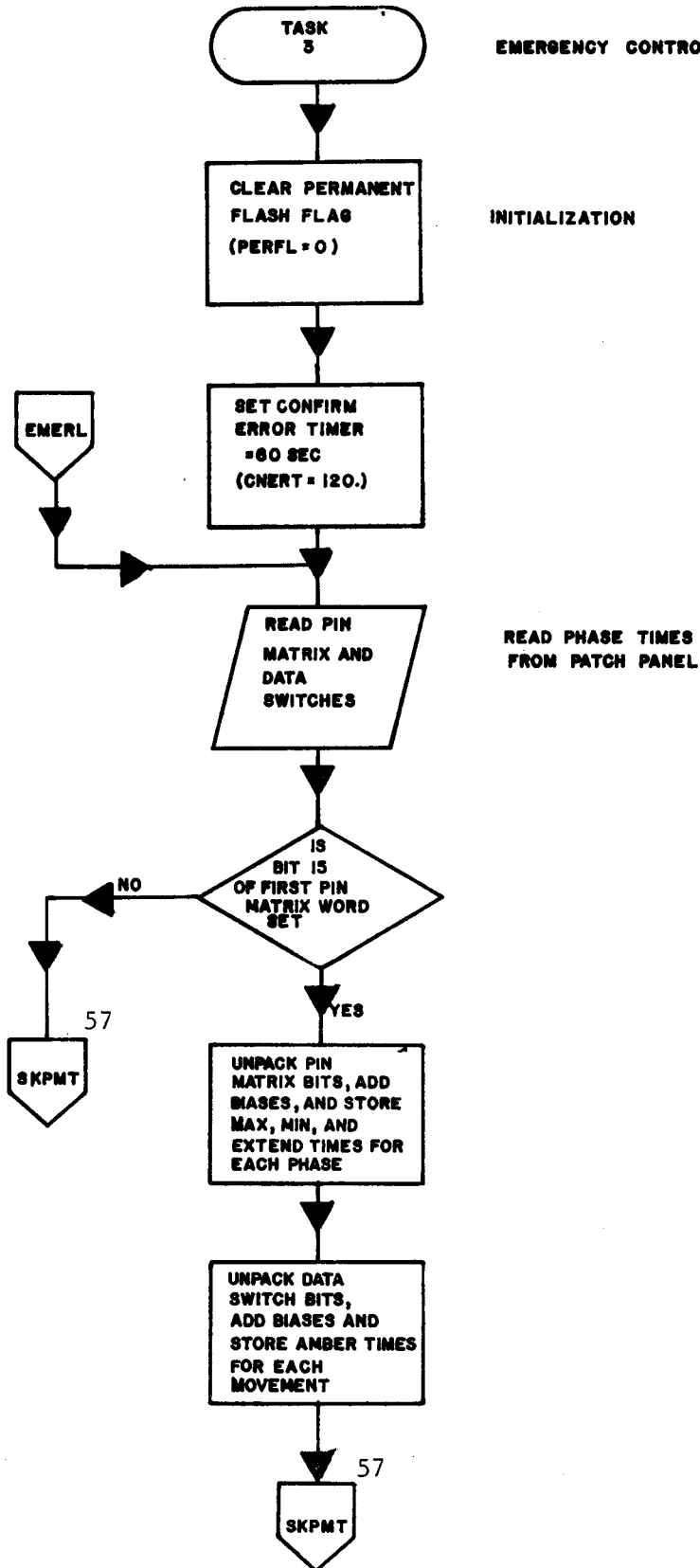


FORMAT AND SEND VARIOUS STATUS BITS VIA TELEMETRY

54



PROGRAM SEGMENT 6



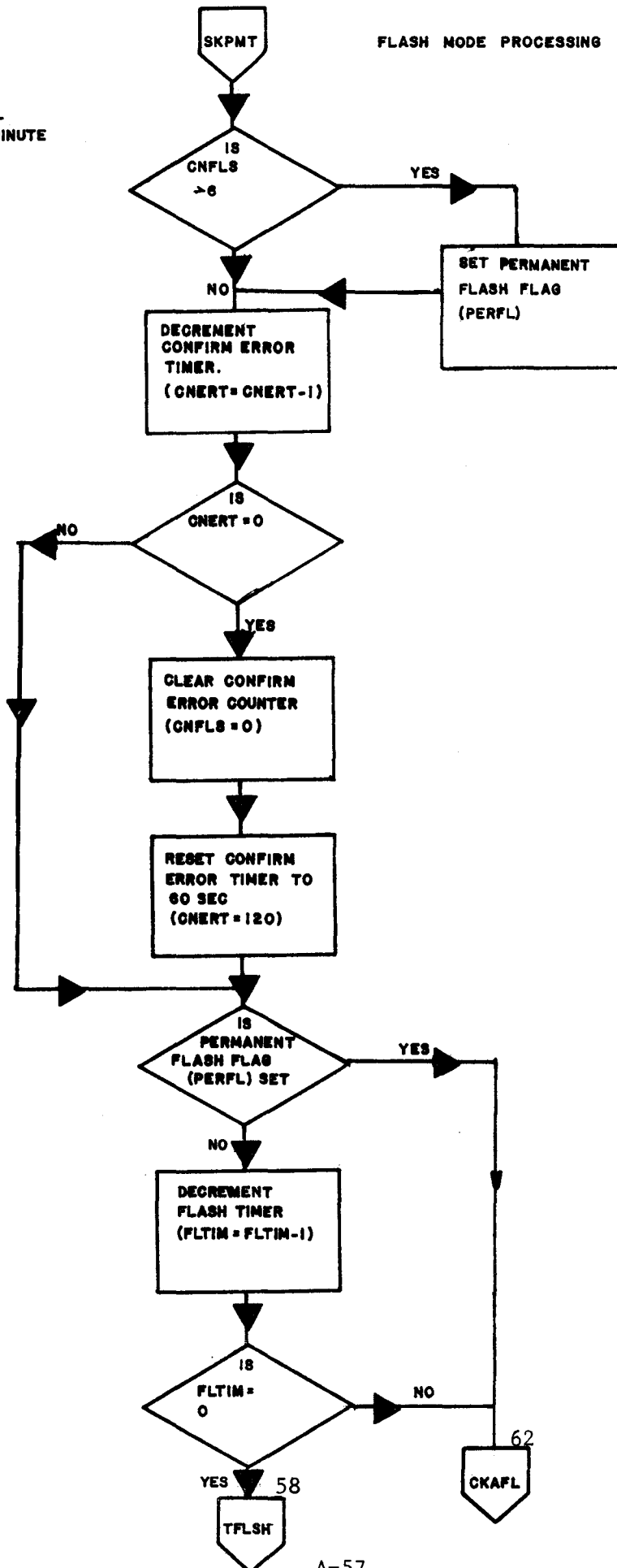
EMERGENCY CONTROL — .5 SEC RATE

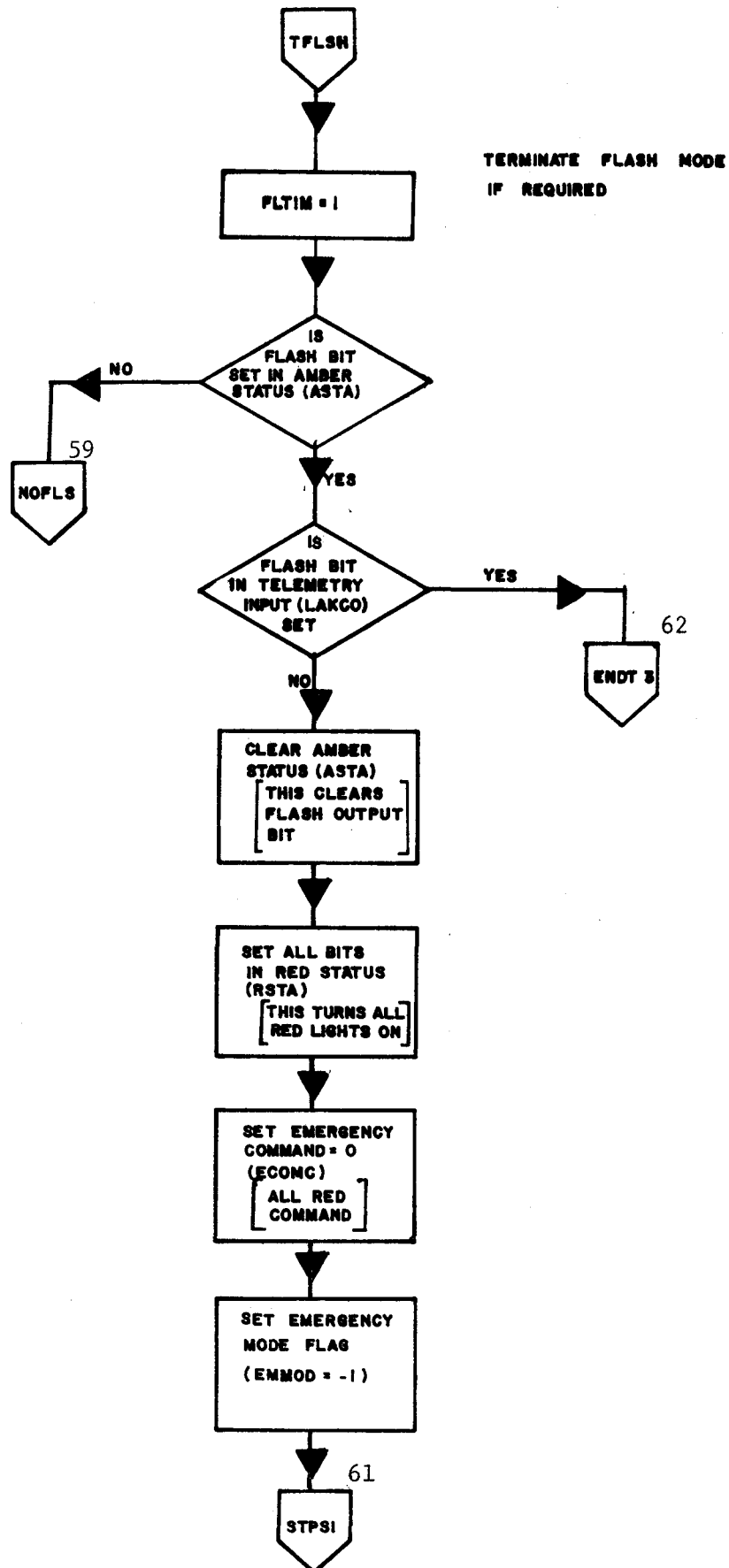
INITIALIZATION

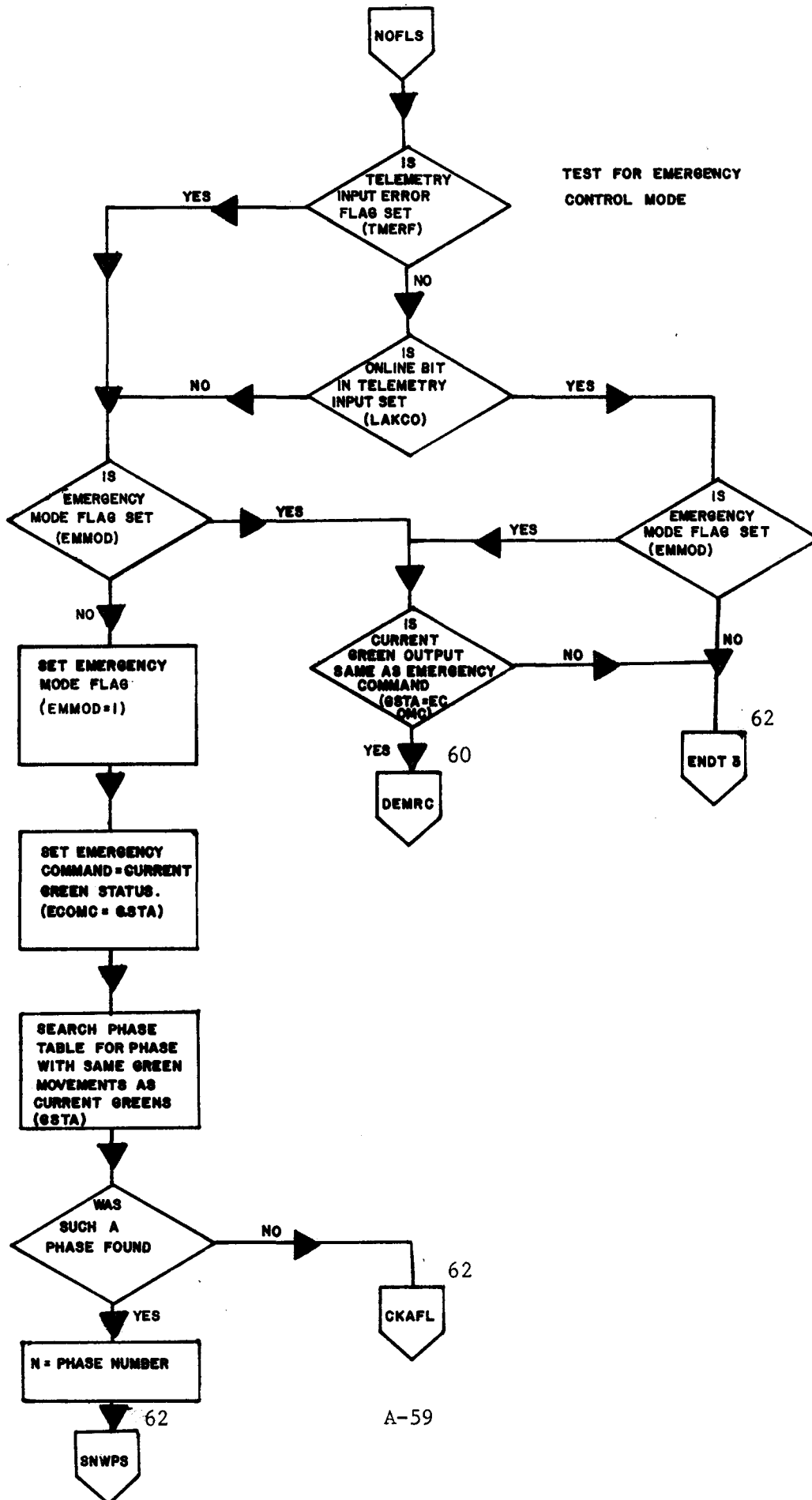
READ PHASE TIMES
FROM PATCH PANEL

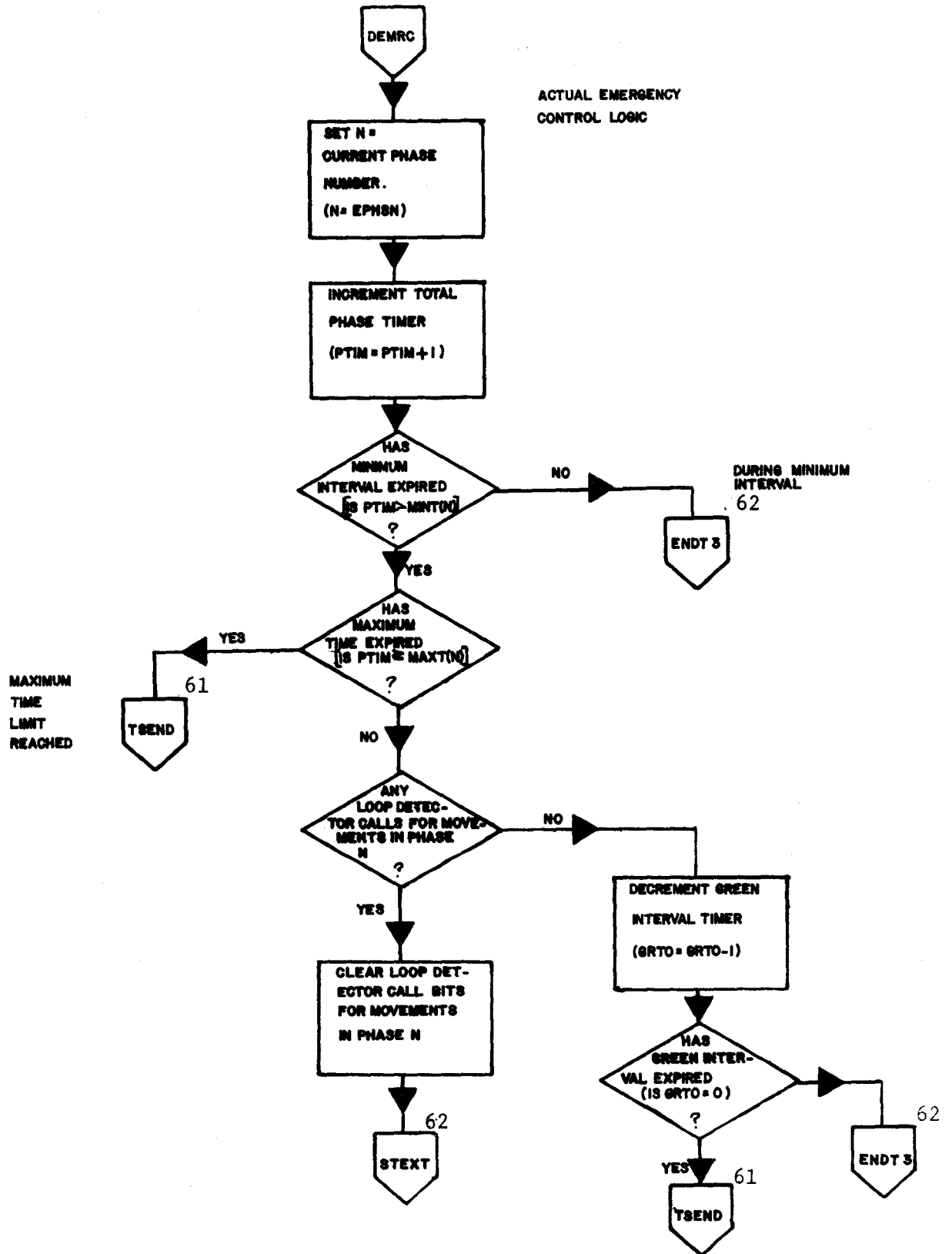
DETERMINE IF # OF SERIOUS SIGNAL
CONFIRM ERRORS EXCEEDS 6 PER MINUTE

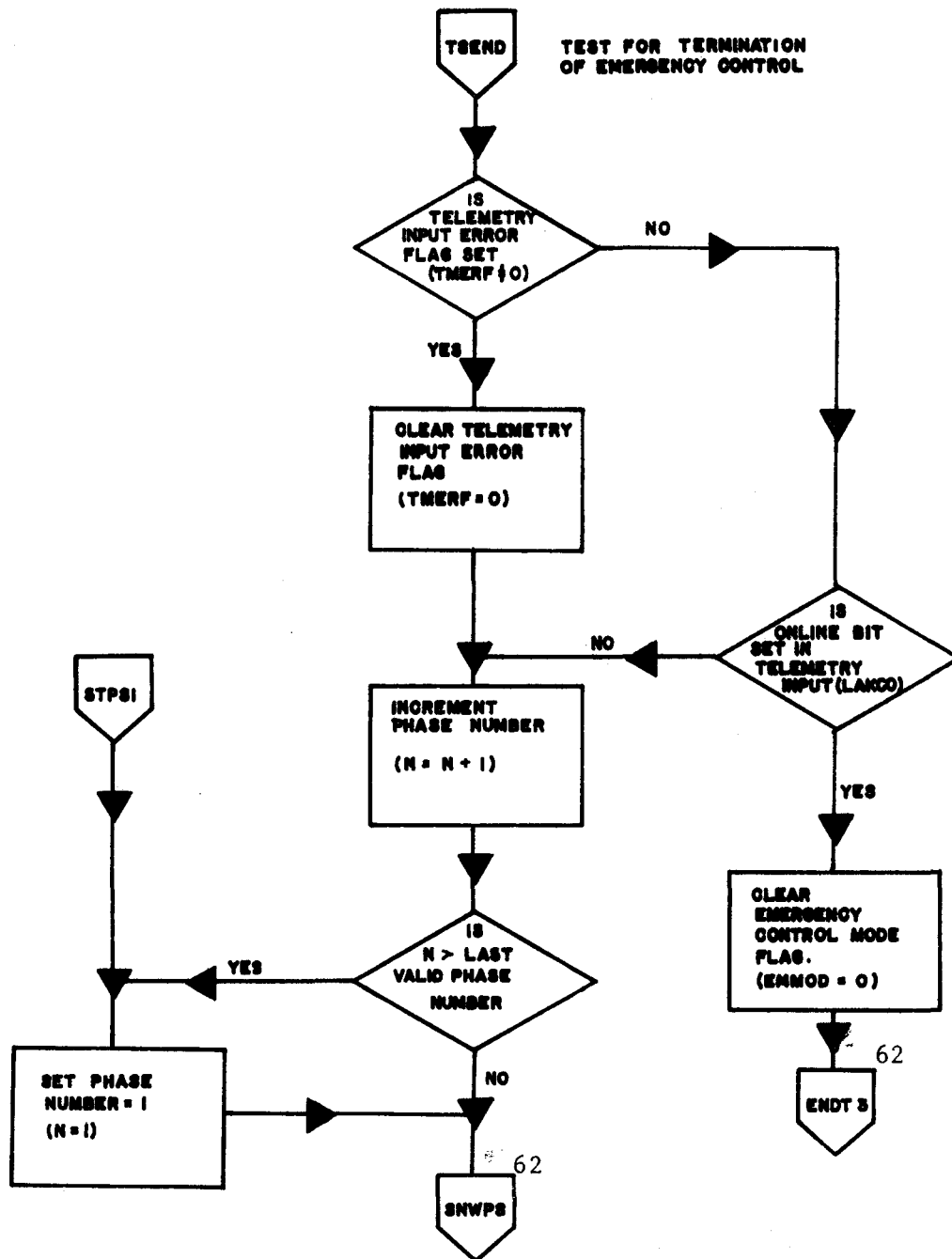
FLASH MODE PROCESSING

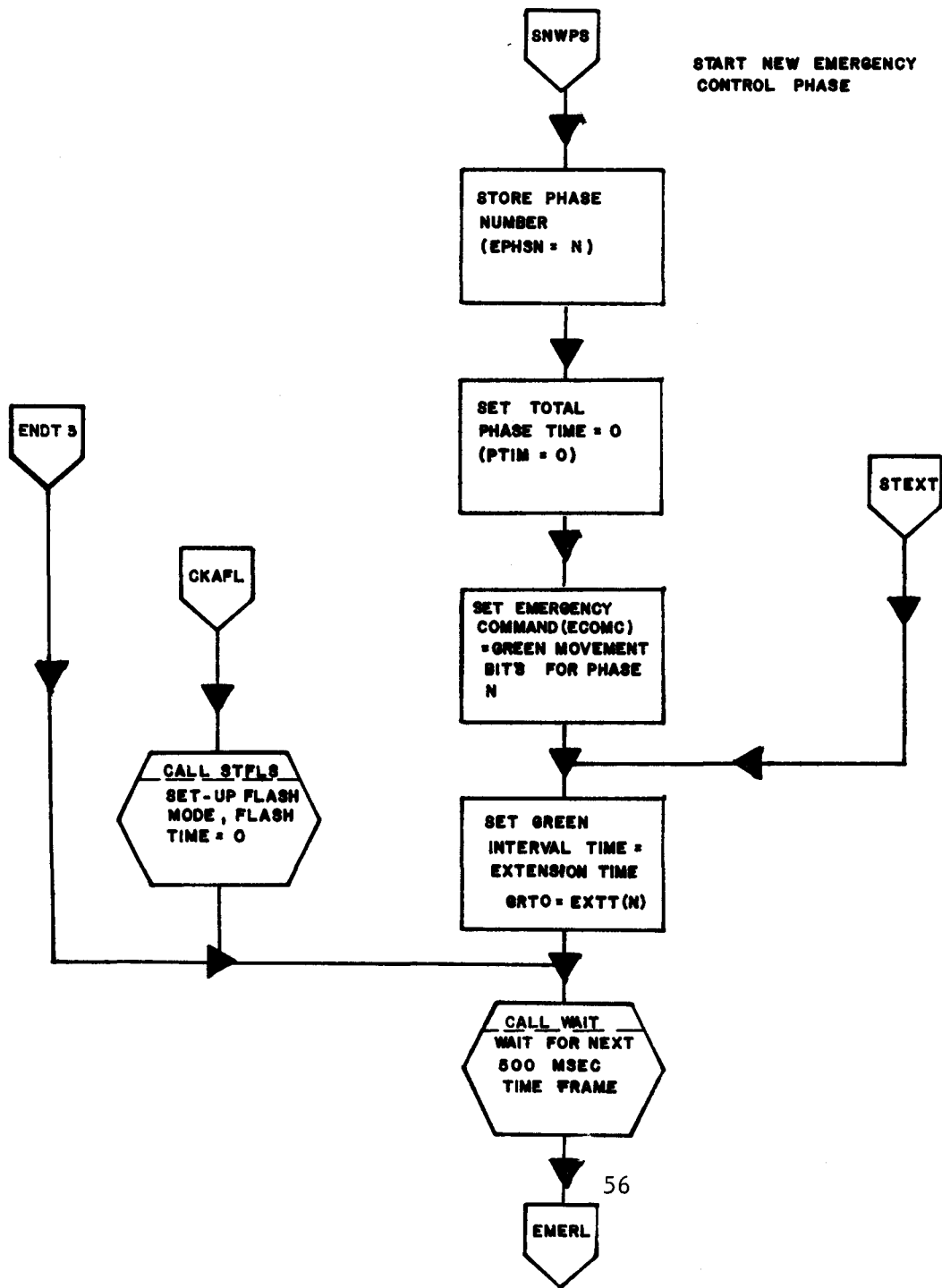




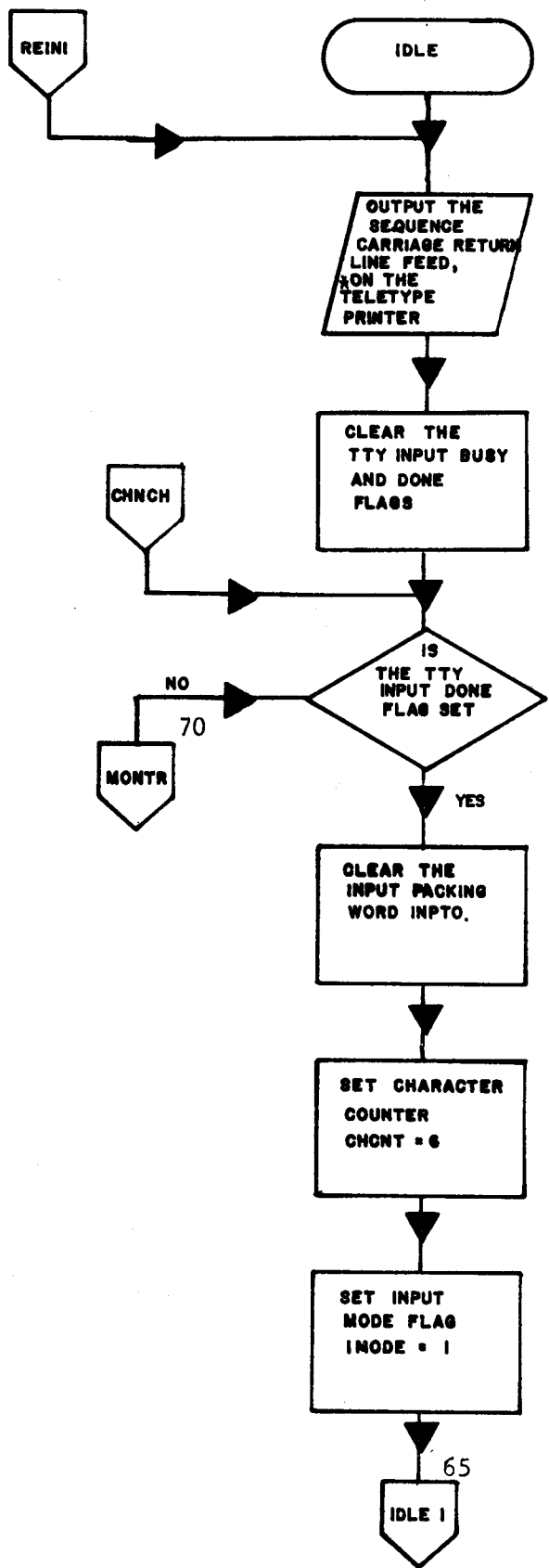






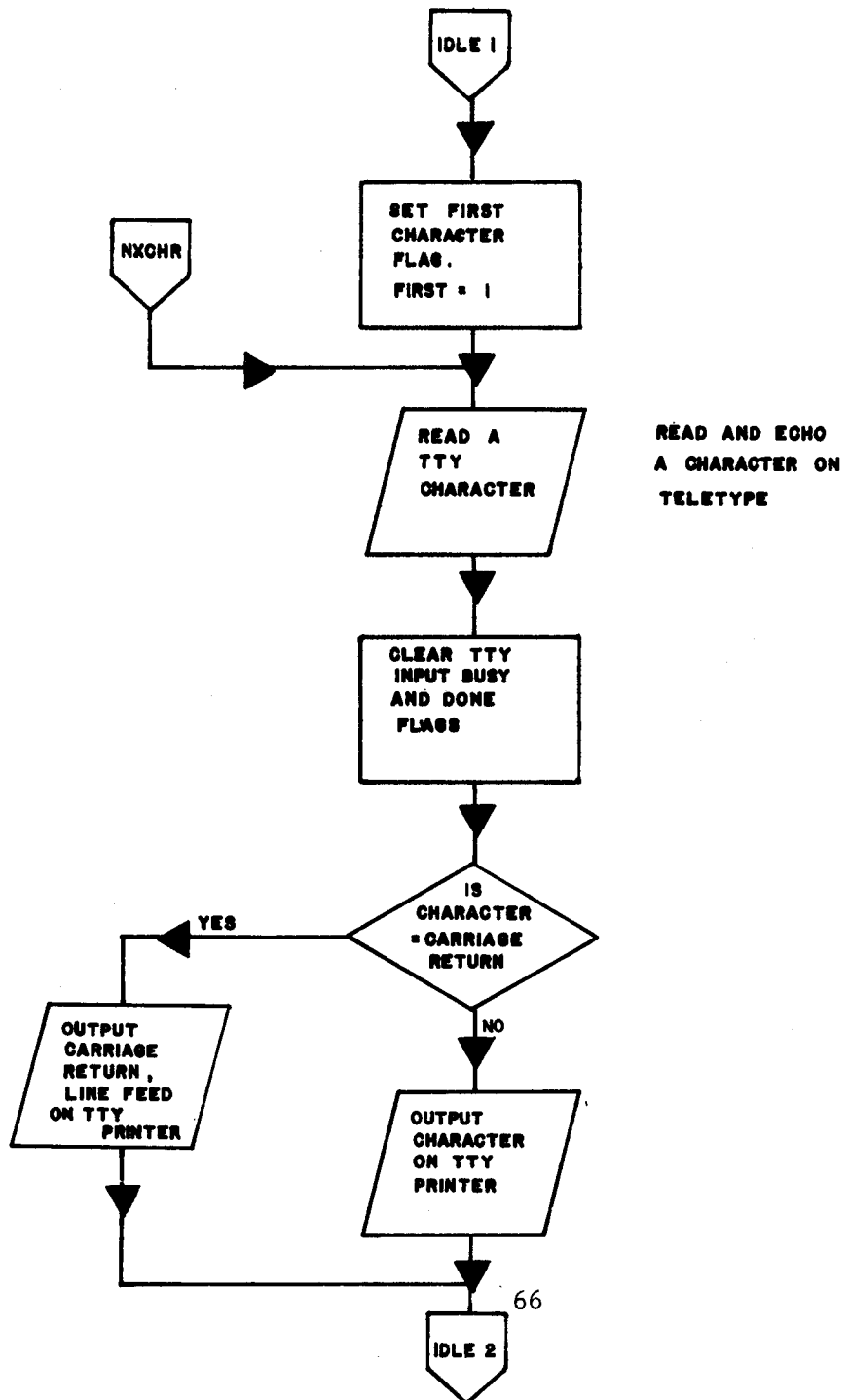


PROGRAM SEGMENT 7

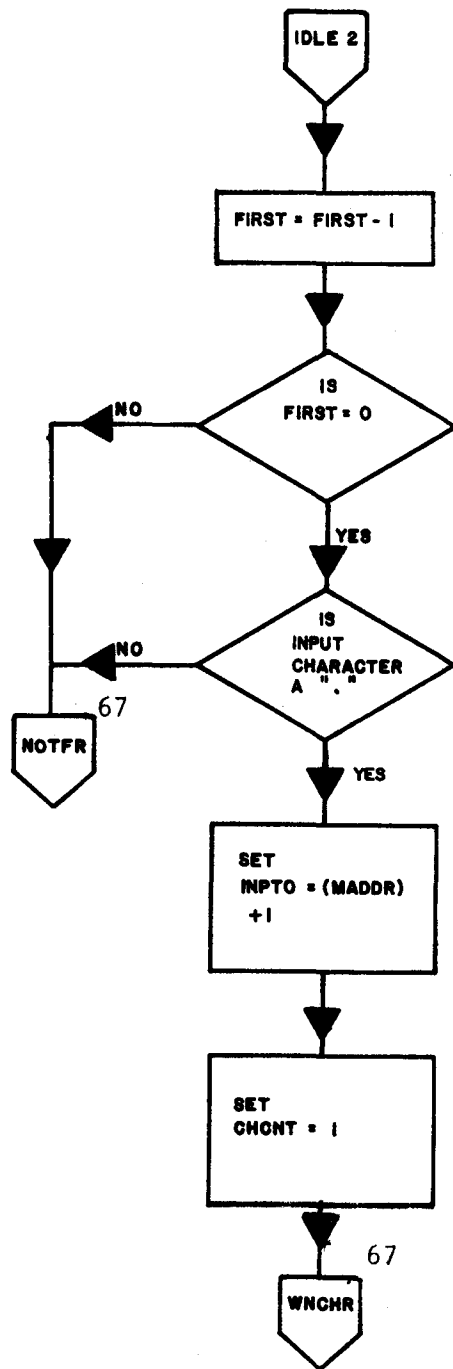


TELETYPE INTERACTION PROGRAM. EXECUTED AS A BACKGROUND TASK IN DEBUG MODE ONLY.

INITIALIZE TO PREPARE FOR TTY INPUT



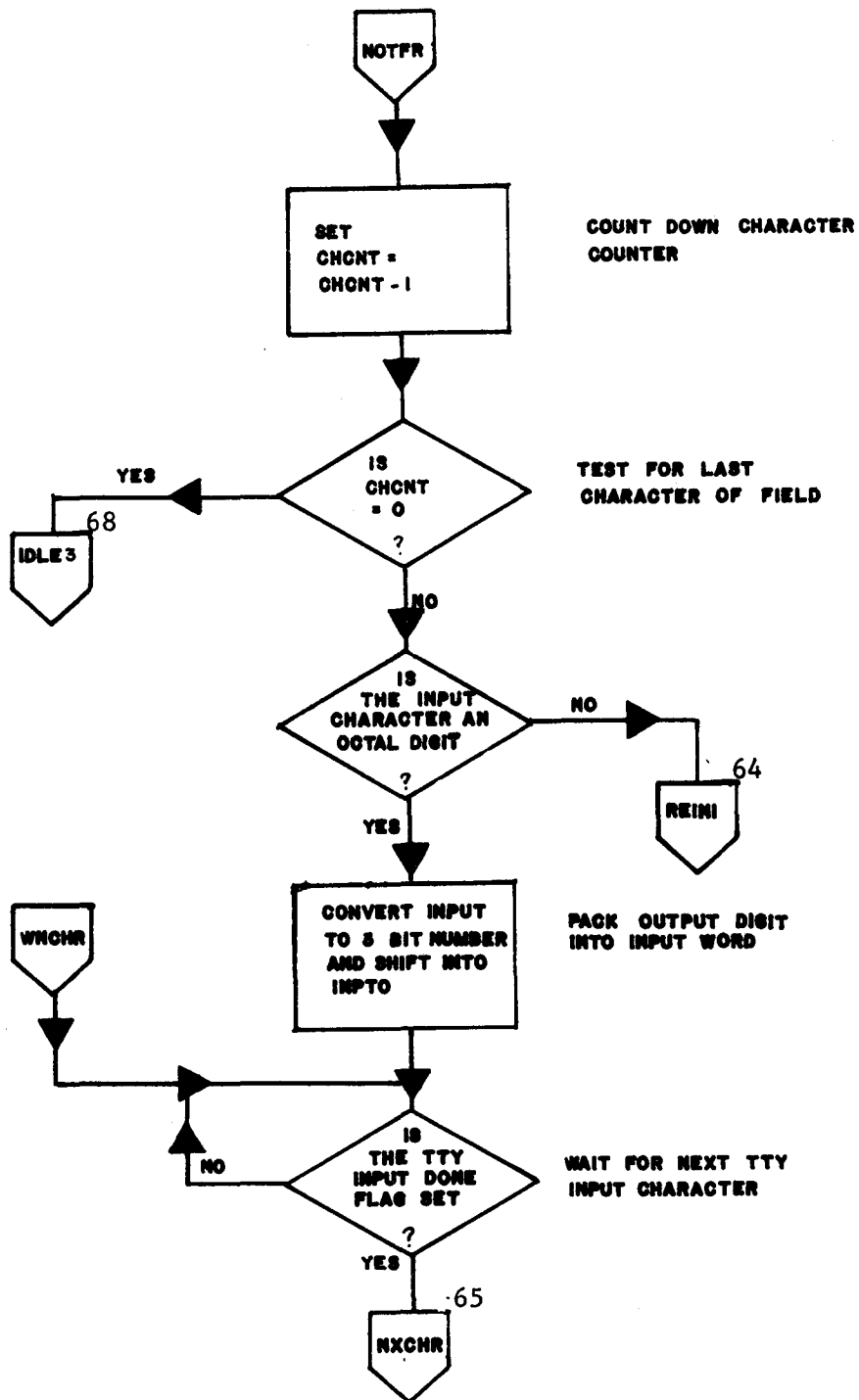
READ AND ECHO
A CHARACTER ON
TELETYPE

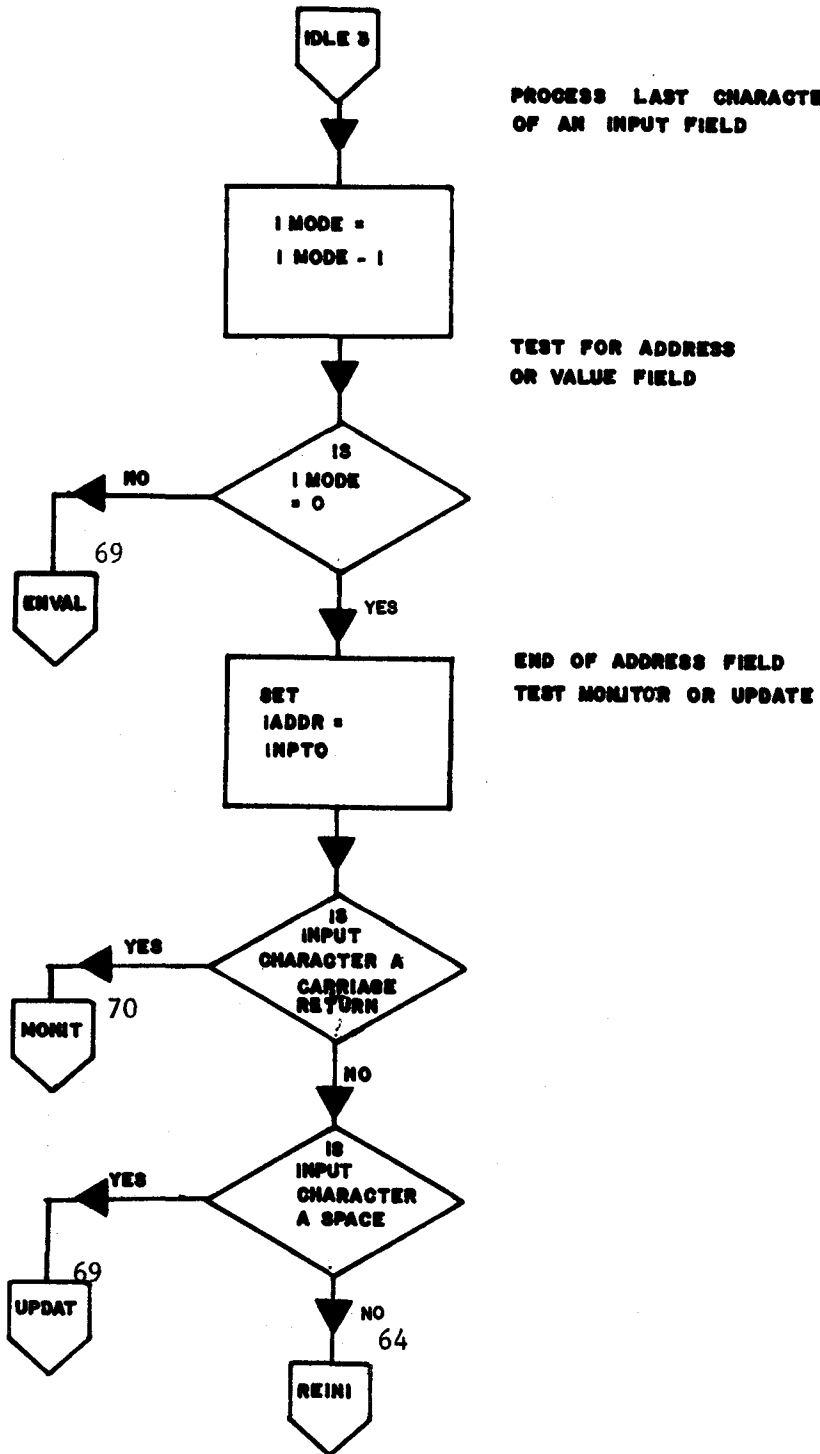


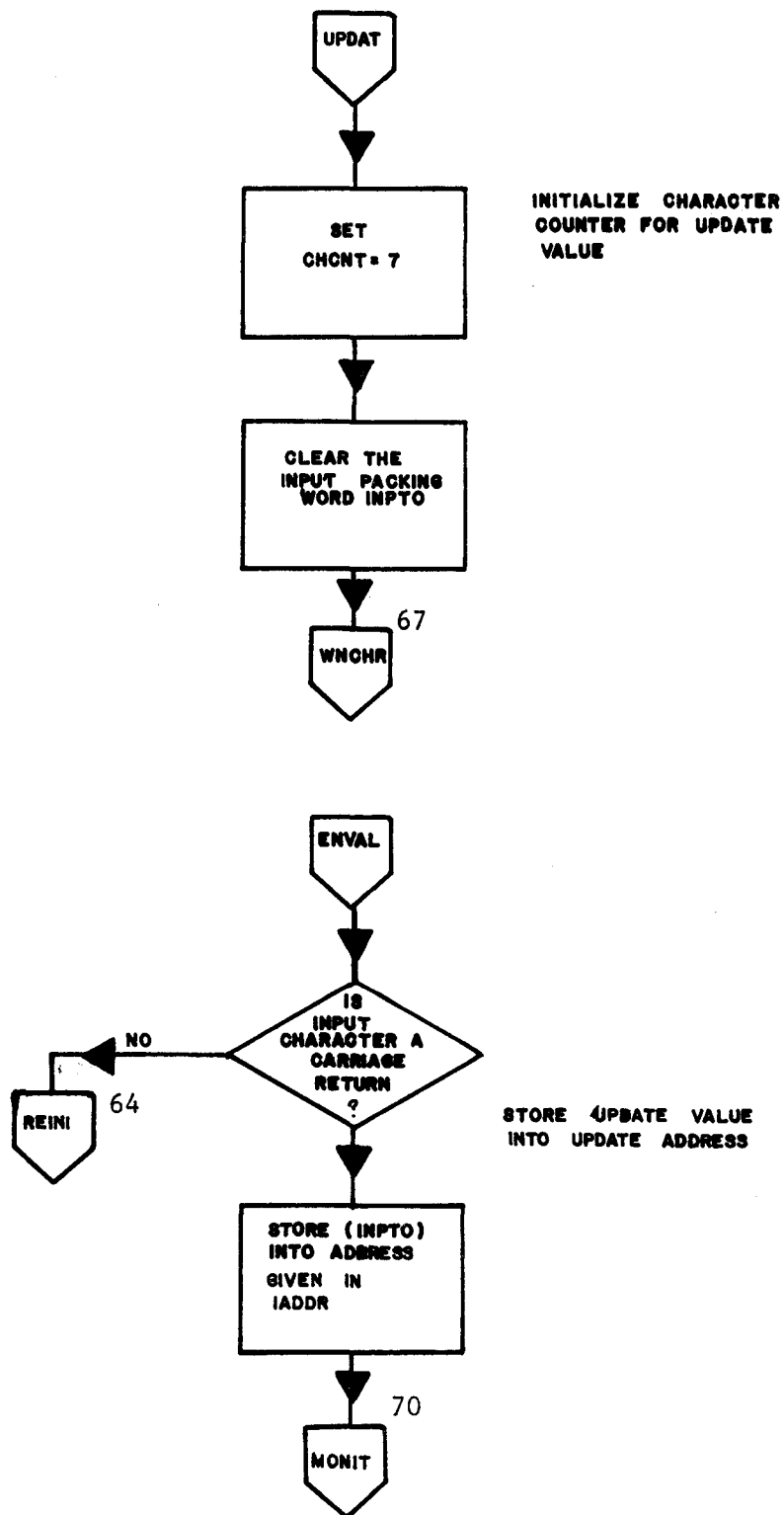
TEST FOR FIRST INPUT CHARACTER

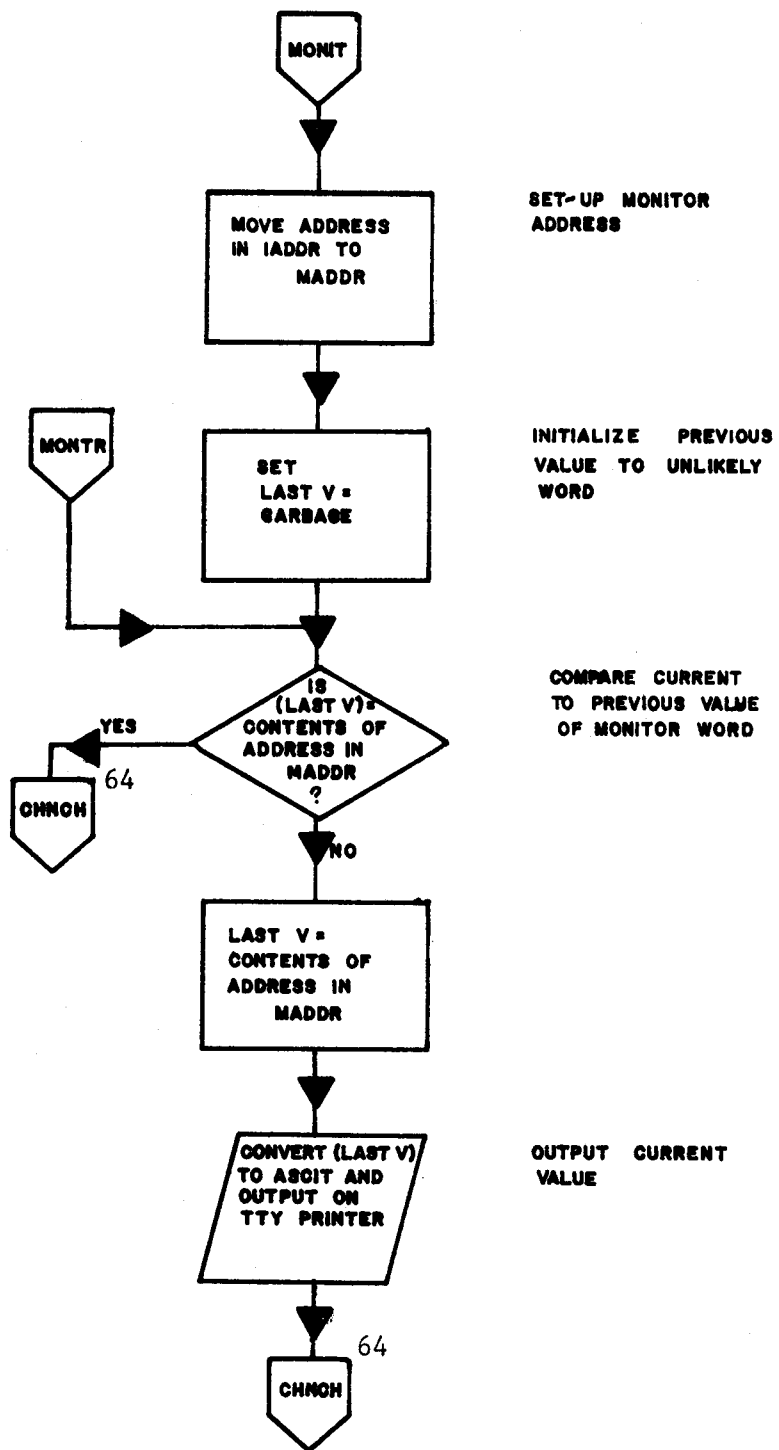
IF FIRST CHARACTER IS A PERIOD, SET THE PACKED INPUT WORD TO THE LAST VALID ADDRESS +1

SET NUMBER OF REMAINING CHARACTERS TO 1









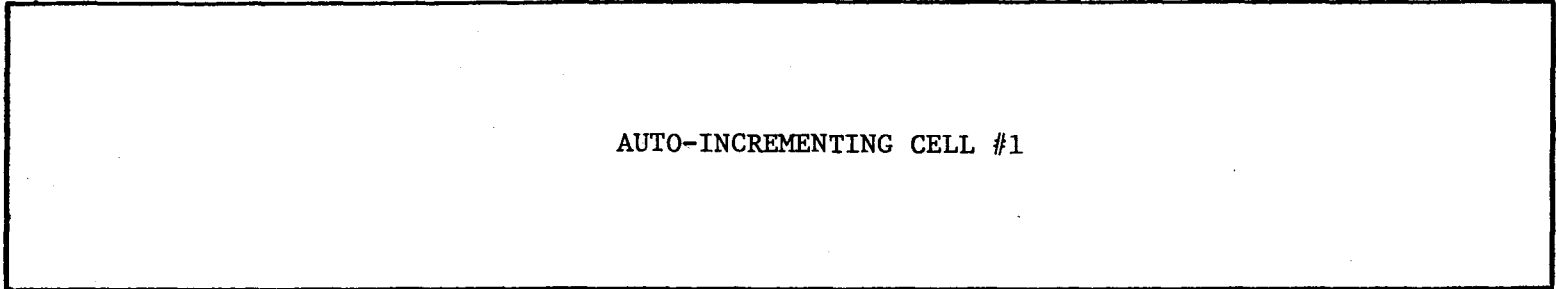
APPENDIX B

ACTUATOR PROGRAM "COMMON" DEFINITIONS

. NAME: AINC1

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...



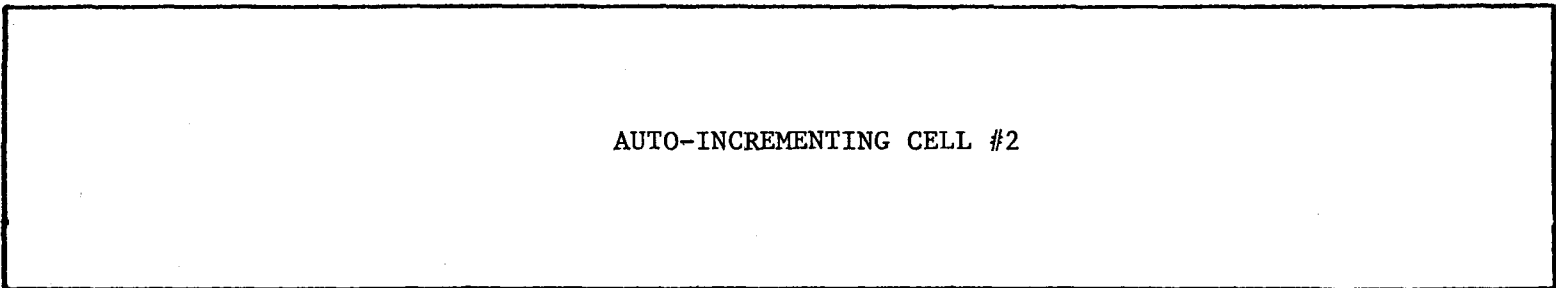
DESCRIPTION: AUTOMATIC INCREMENTING CELL #1 USED BY TASK 1.

B-1

. NAME: AINC2

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...

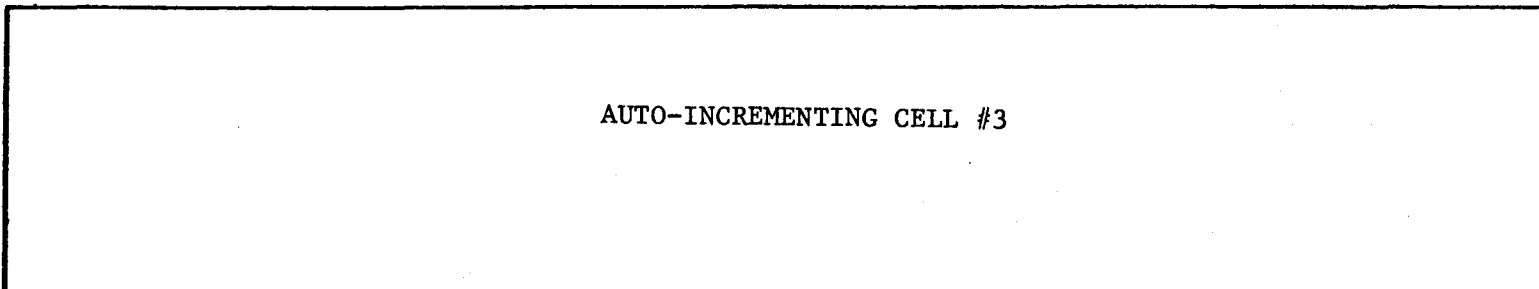


DESCRIPTION: AUTOMATIC INCREMENTING CELL #2 USED BY TASK 1.

. NAME: AINC3

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...



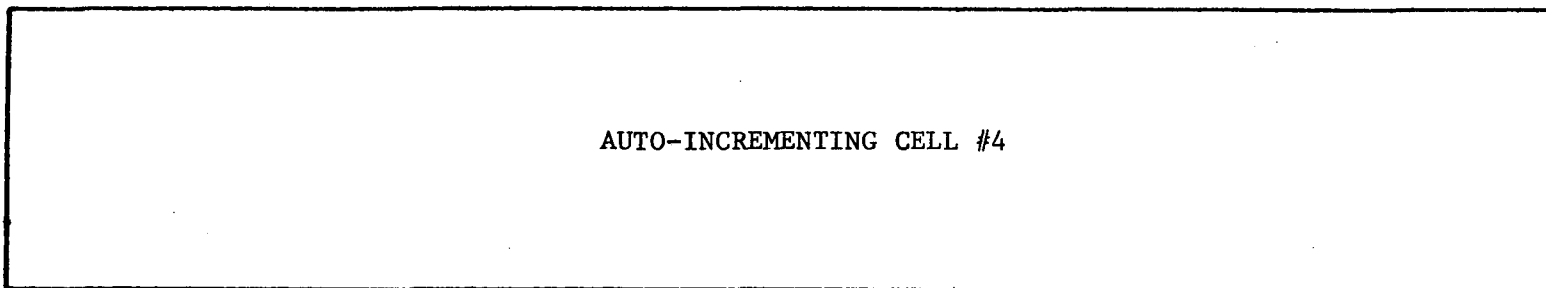
DESCRIPTION: AUTOMATIC INCREMENTING CELL #3 USED BY TASK 2.

B-2

. NAME: AINC4

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...

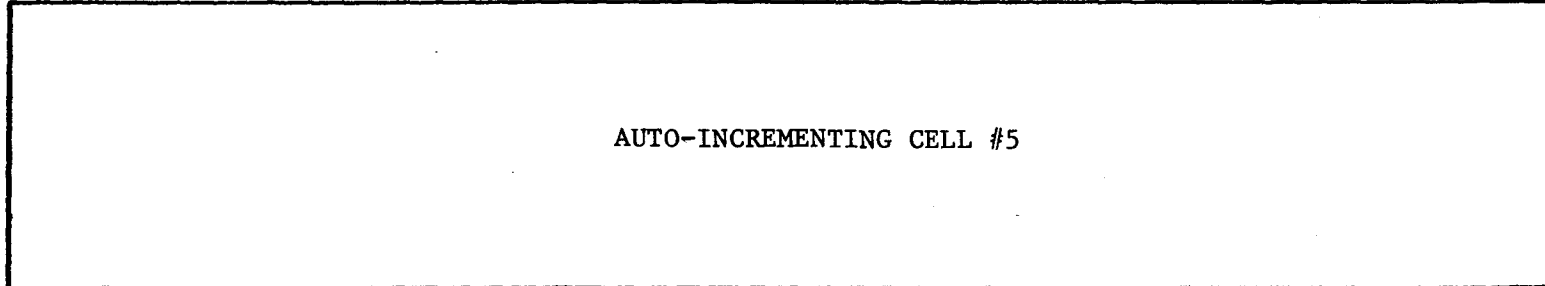


DESCRIPTION: AUTOMATIC INCREMENTING CELL #4 USED BY TASK 2.

. NAME: AINC5

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...



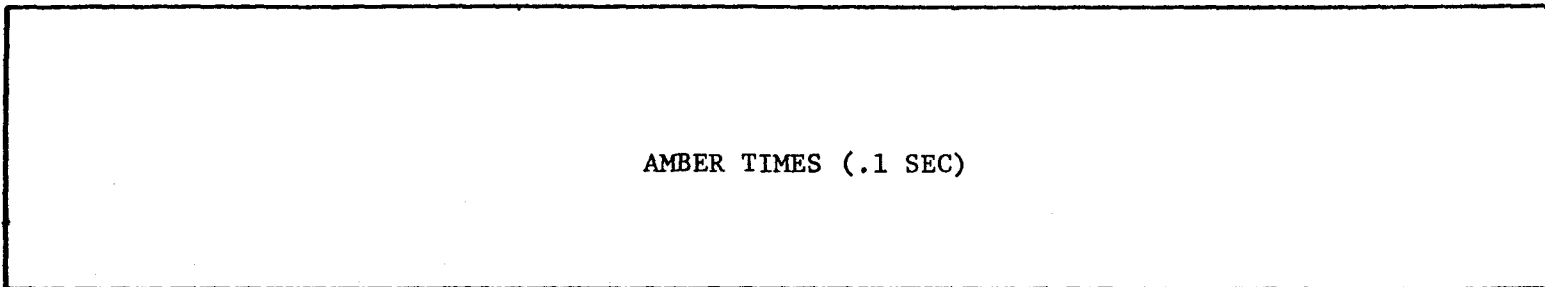
DESCRIPTION: AUTOMATIC INCREMENTING CELL #5 USED BY TASK 3.

B-3

. NAME: AMTM

SIZE: 9

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...

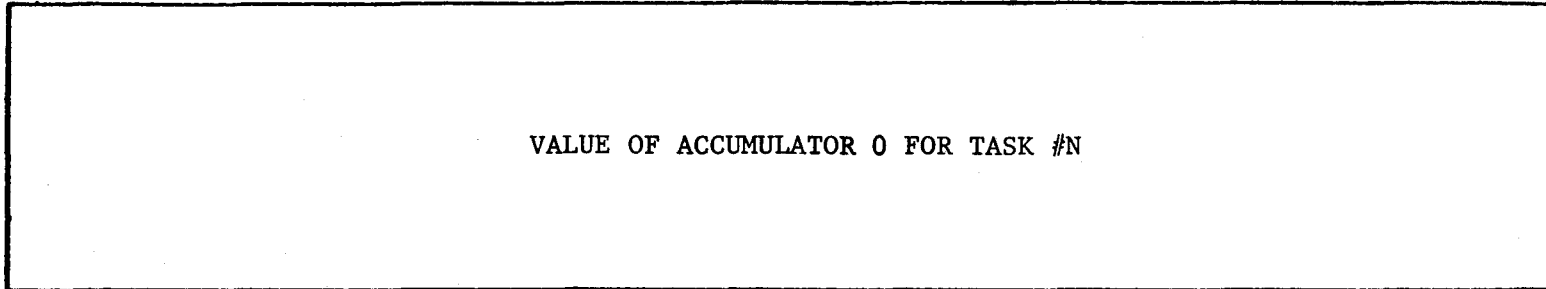


DESCRIPTION: AMBER TIMES FOR EACH MOVEMENT IN .1 SECOND UNITS. USED IN TASKS 2 AND 3. AMTM+0 IS THE TIME FOR AMBER BIT 0 (SEE APPENDIX C, OUTPUT V+1), AMTM+1 FOR AMBER BIT 1, ETC.

. NAME: AOSAV

SIZE: 4

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...



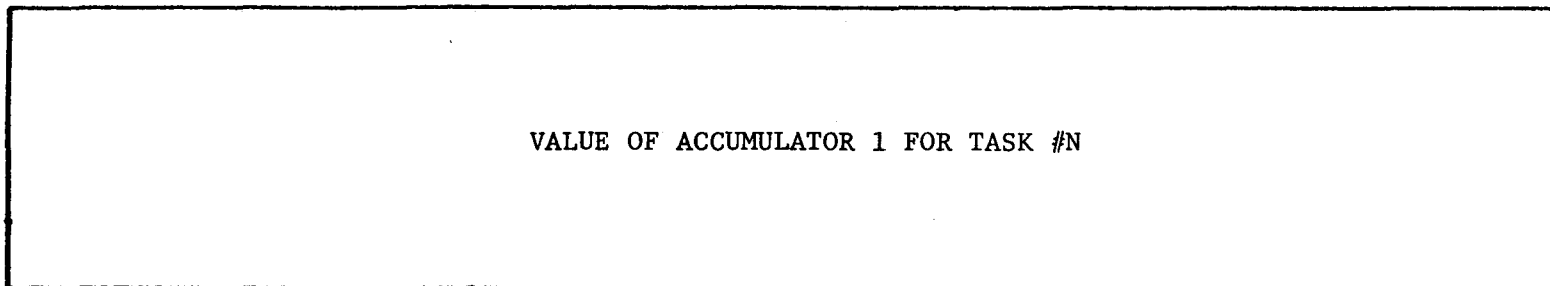
DESCRIPTION: WHEN TASK #N IS INTERRUPTED THE CURRENT VALUE OF ACCUMULATOR 0 IS SAVED IN THIS ARRAY.
WHEN TASK #N IS TO BE EXECUTED, THE VALUE FOR ACO IS OBTAINED FROM HERE.

B-4

. NAME: ALSAV

SIZE: 4

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...



DESCRIPTION: SAVED AC1. SEE AOSAV.

. NAME: A2SAV

SIZE: 4

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...

VALUE OF ACCUMULATOR 2 FOR TASK #N

DESCRIPTION: SAVED AC2. SEE A0SAV.

B-5

. NAME: A3SAV

SIZE: 4

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...

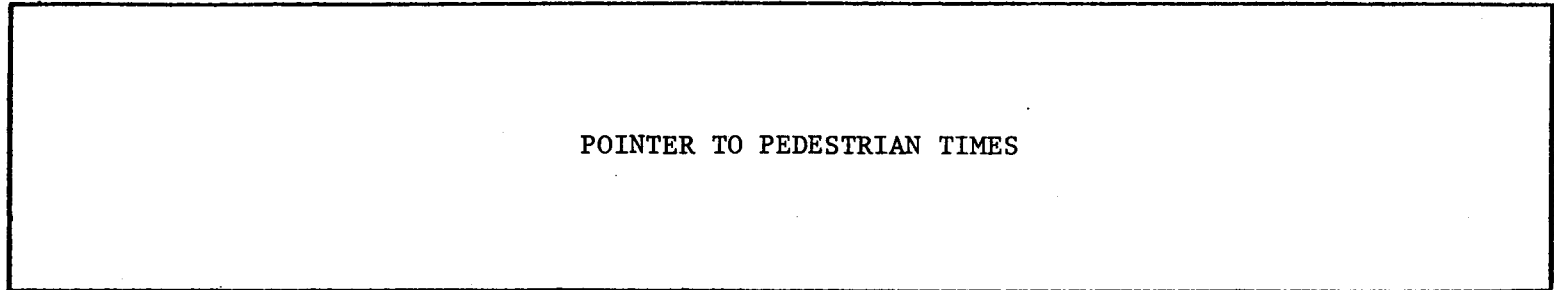
VALUE OF ACCUMULATOR 3 FOR TASK #N

DESCRIPTION: SAVED AC3. SEE A0SAV.

. NAME: APDMT

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...



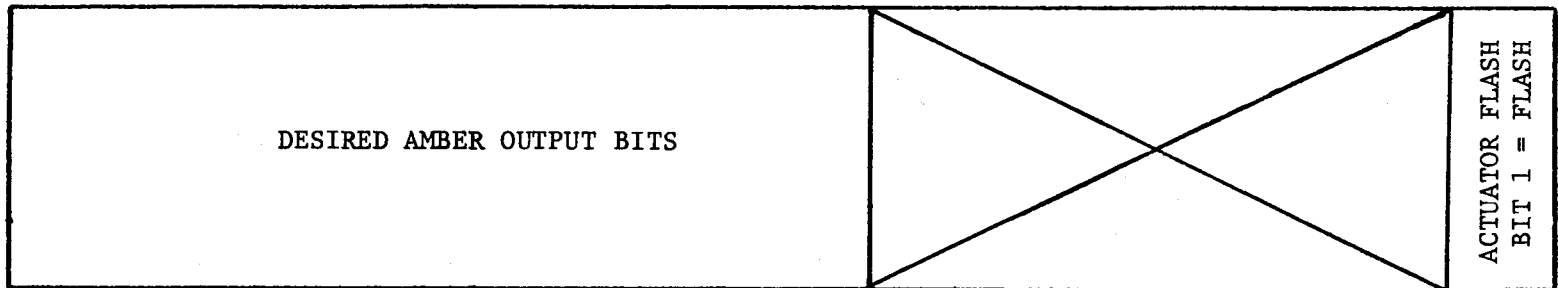
DESCRIPTION: ADDRESS OF PDMNT, WHICH CONTAINS "WALK" TIMES. USED BY TASK 2.

B-6

. NAME: ASTA

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...

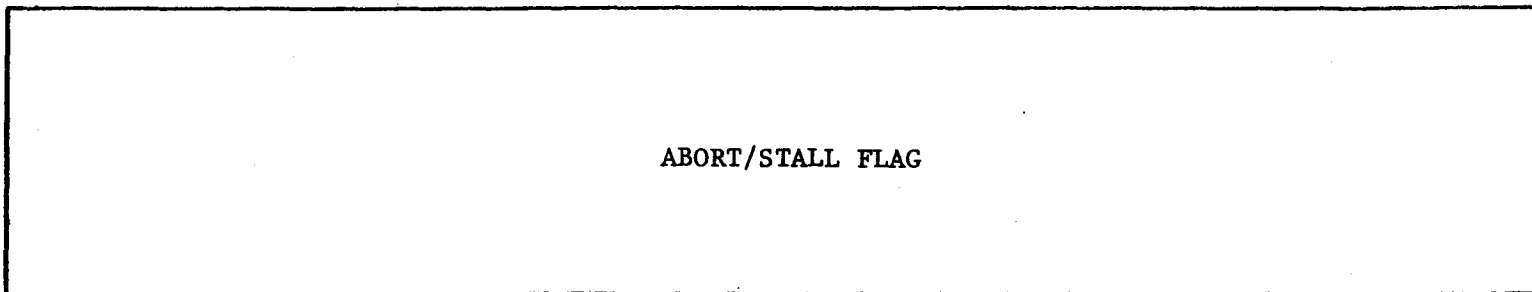


DESCRIPTION: DESIRED VALUE OF AMBERS FOR EACH MOVEMENT. 1 = AMBER LIGHT ON, 0 = AMBER LIGHT OFF.
BIT 15 CONTROLS THE ACTUATOR FLASH RELAY. USED BY TASKS 2 AND 3 AND SUBROUTINE STFLS.

. NAME: ASTLF

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...



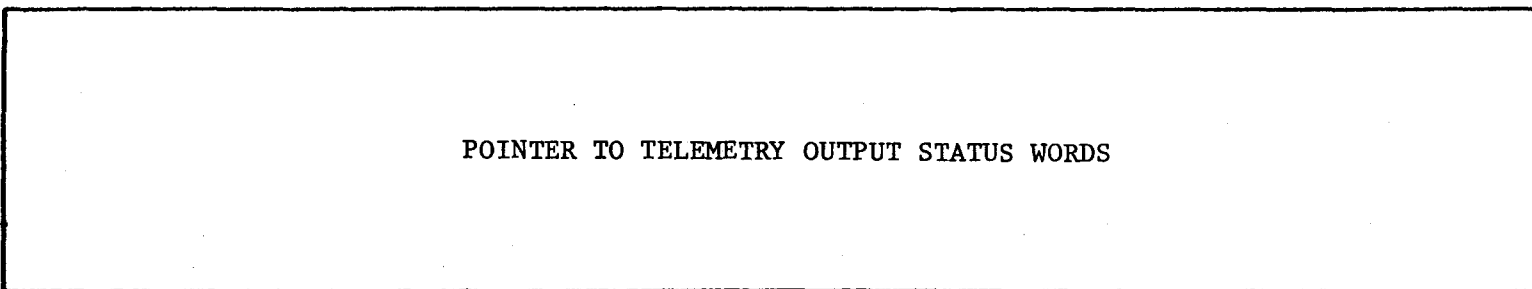
DESCRIPTION: CONTAINS ADDRESS +1 OF CALL TO SUBROUTINE ABORT IF A FATAL ERROR OR TELEMETRY ABORT IS REQUESTED. NORMALLY IS ZERO. CHECKED BY INTERRUPT ROUTINE AND RESTART.

B-7

. NAME: ASTWR

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...



DESCRIPTION: ADDRESS -1 OF FIRST TELEMETRY OUTPUT STATUS WORD (STWRD-1). USED BY TELEMETRY OUTPUT.

. NAME: BIT

SIZE: 16

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...

BIT TABLE

DESCRIPTION: A BIT TABLE WITH THE FIRST WORD CONTAINING ONLY BIT 0 SET, THE SECOND WORD ONLY BIT 1 SET, ETC. VARIOUS NUMERIC CONSTANTS AND BIT DEFINITIONS ARE EQUATED TO ENTRIES IN BIT . USED BY ALL ACTUATOR PROGRAM SEGMENTS.

B-8

. NAME: BITN

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...

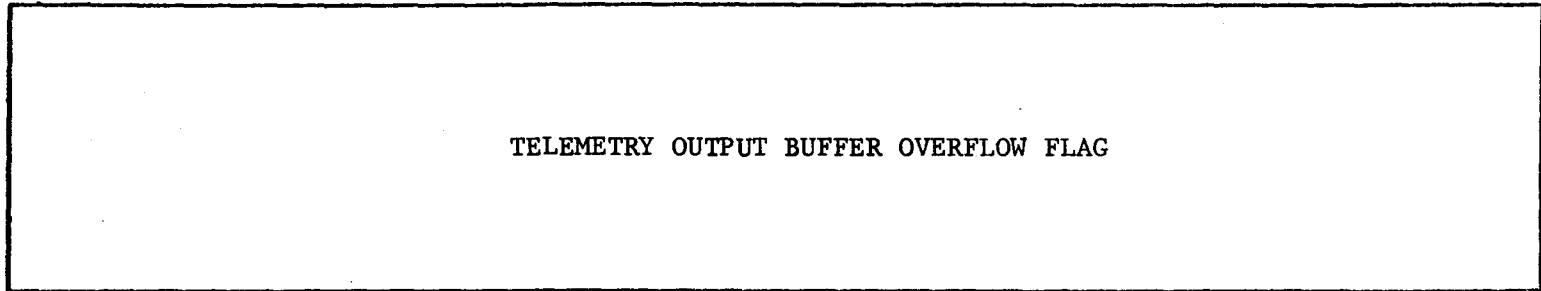
TELEMETRY OUTPUT BIT COUNTER

DESCRIPTION: A COUNTER USED BY TELEMETRY OUTPUT TO COUNT THE TELEMETRY OUTPUT BITS PER WORD.

. NAME: BOFLG

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...



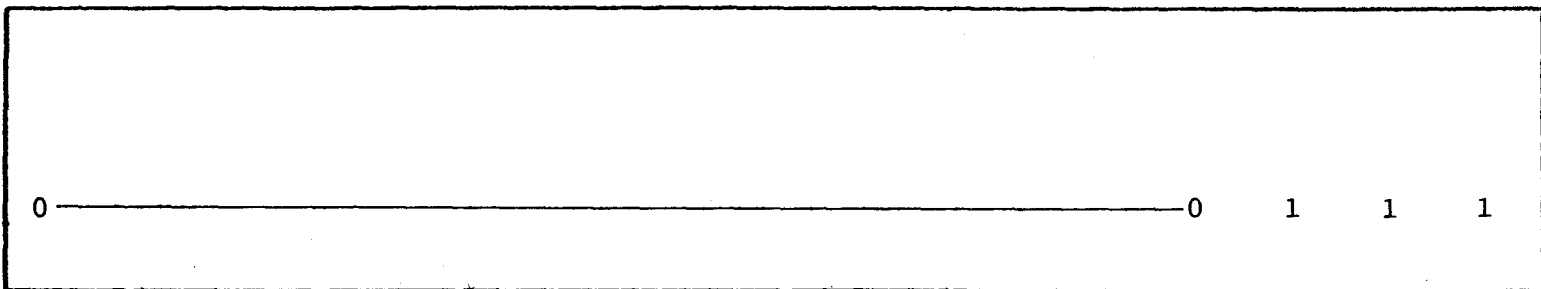
DESCRIPTION: SET WHENEVER A TASK ATTEMPTS TO PLACE A WORD IN THE TELEMETRY OUTPUT BUFFER USING SUBROUTINE PUT AND THE BUFFER IS FULL. USED BY TASKS 1 AND 2.

B-9

. NAME: C7

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...

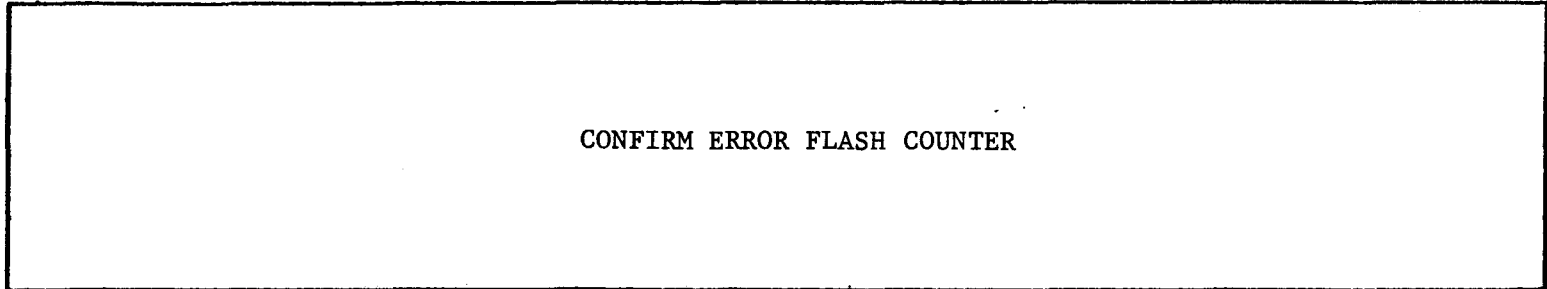


DESCRIPTION: CONSTANT 7 USED BY SEVERAL PROGRAM SEGMENTS.

. NAME: CNFLS

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...



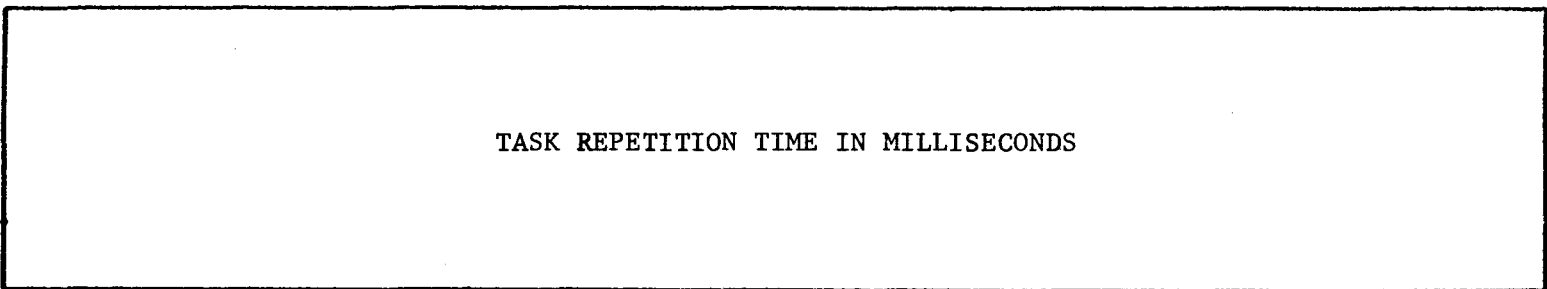
DESCRIPTION: INCREMENTED BY TASK 2 WHENEVER A SERIOUS SIGNAL CONFIRM ERROR CAUSES THE INTERSECTION TO BE PLACED ON FLASH. CHECKED BY TASK 3 TO DETERMINE IF SUCH ERRORS ARE OCCURRING FREQUENTLY ENOUGH TO JUSTIFY PERMANENT FLASH.

B-10

. NAME: CYCLE

SIZE: 4

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...



DESCRIPTION: THE NUMBER OF MILLISECONDS BETWEEN TASK EXECUTIONS FOR EACH TASK. USED BY RESTART AND TASK CONTROL.

. NAME: DATAM

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...

0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

DESCRIPTION: CONSTANT FOR MASKING TELEMETRY OUTPUT DATA BITS. USED IN TELEMETRY OUTPUT SECTION.

B-11

. NAME: ECOMC

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...

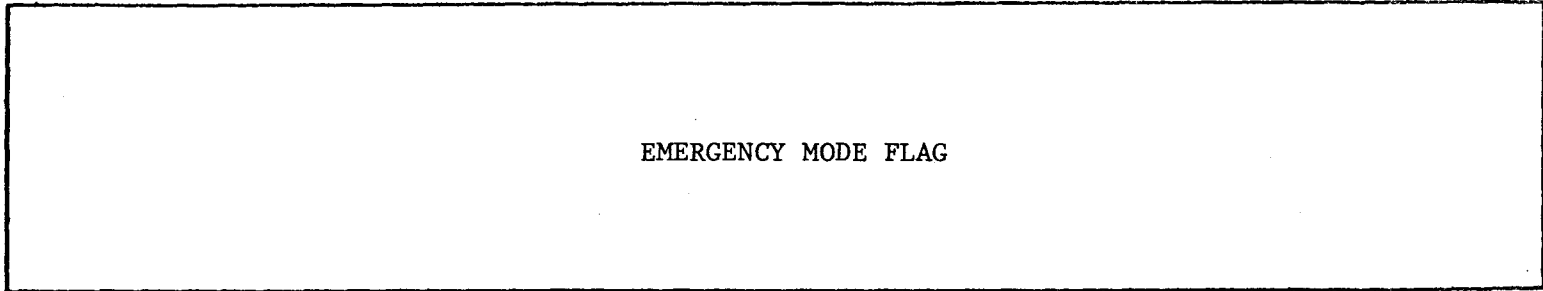
EMERGENCY COMMAND CODE															
------------------------	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

DESCRIPTION: THE SIGNAL COMMAND WORD GENERATED BY THE EMERGENCY CONTROL TASK (TASK 3). USED BY TASK 2 WHENEVER THE ACTUATOR IS UNDER BACKUP (EMERGENCY) CONTROL. THE BITS REPRESENT THE COMMANDED GREEN LIGHT STATUS FOR EACH MOVEMENT.

. NAME: EMMOD

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...



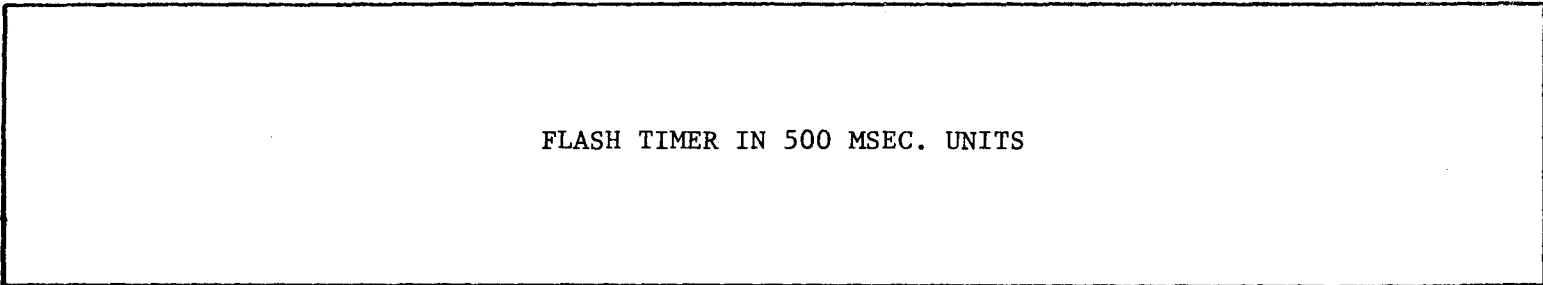
DESCRIPTION: ZERO = BACKUP (EMERGENCY) CONTROL NOT ACTIVE. NONZERO = BACKUP (EMERGENCY) CONTROL ACTIVE. SET BY TASK 3. USED BY TASK 2 TO DETERMINE WHETHER TO USE EMERGENCY CONTROL SIGNAL COMMAND (ECOMC) OR TELEMETRY SIGNAL COMMAND.

B-12

. NAME: FLTIM

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...



DESCRIPTION: A COUNTER THAT GIVES THE TIME TO REMAIN IN ACTUATOR FLASH MODE. SET BY SUBROUTINE STFLS. DECREMENTED EVERY 1/2 SECOND BY TASK 3.

. NAME: FORMM

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...

0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

DESCRIPTION: CONSTANT FOR MASKING TELEMETRY OUTPUT FORMAT BITS. USED BY SUBROUTINE PTSUB.

B-13

. NAME: GSTA

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...

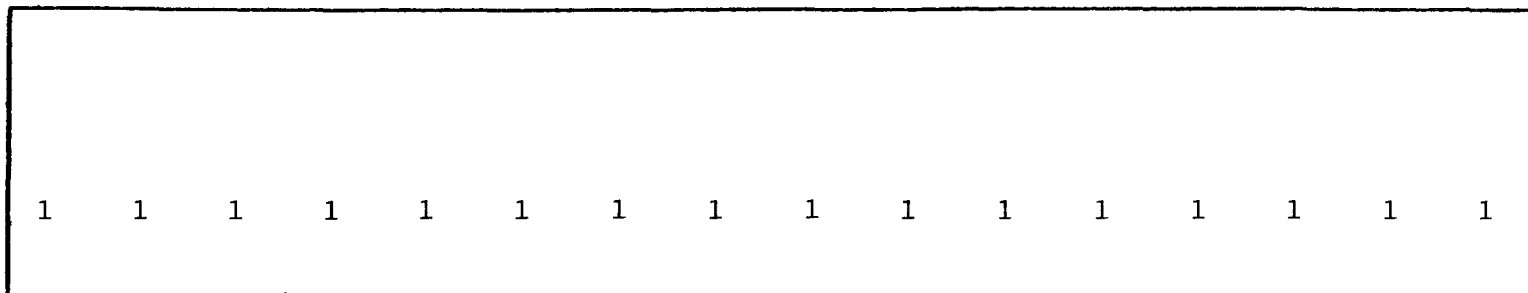
DESIRED GREEN LIGHT OUTPUT BITS	DESIRED "WALK" LIGHT OUTPUT BITS
---------------------------------	----------------------------------

DESCRIPTION: DESIRED VALUE OF GREENS AND WALK LIGHTS FOR EACH MOVEMENT. 1 = GREEN/WALK LIGHT ON, 0 = GREEN/WALK LIGHT OFF. USED BY TASKS 2 AND 3 AND SUBROUTINE STFLS.

. NAME: IMASK

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...



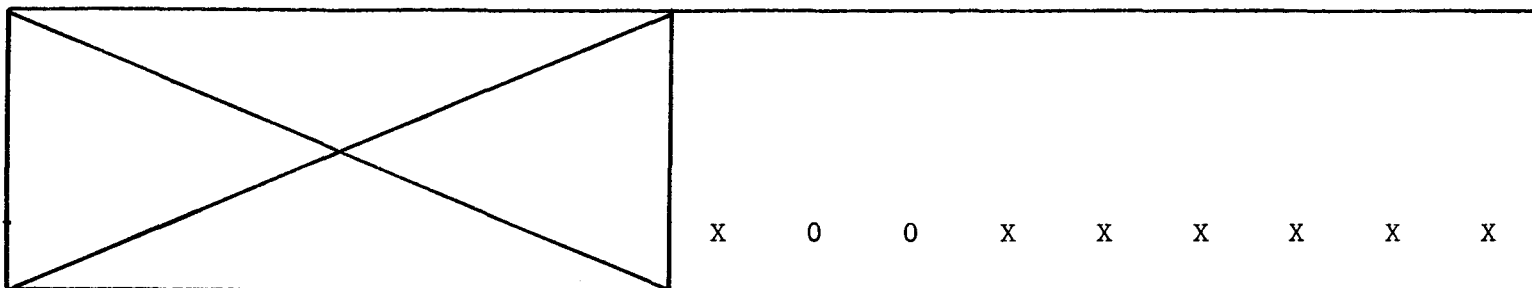
DESCRIPTION: CONSTANT USED TO MASK (INHIBIT) INTERRUPTS FROM ALL DEVICES. IT IS USED IN INTERRUPT ROUTINE AND RESTART TO INHIBIT ALL INTERRUPTS EXCEPT FOR POWER FAIL.

B-14

. NAME: LAKCO

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...

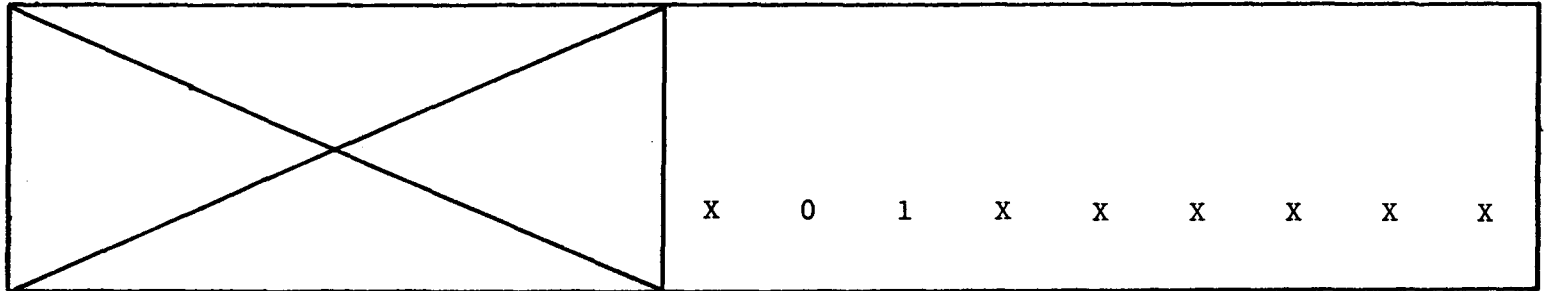


DESCRIPTION: THE LAST TELEMETRY INPUT WORD WITH BIT 9 = 0 THAT WAS ACKNOWLEDGED. SEE APPENDIX D FOR DESCRIPTION OF EACH BIT (INPUT FORMAT 0). USED BY TASKS 2 and 3.

. NAME: LAKC1

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...



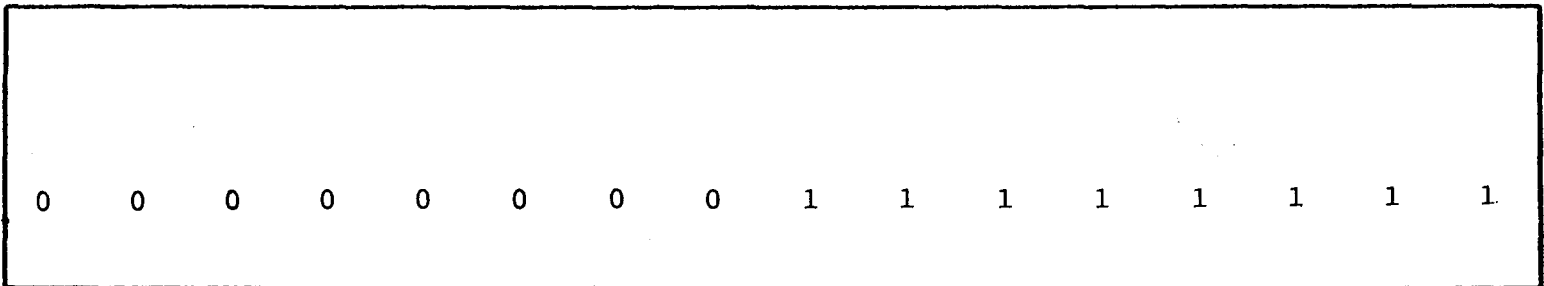
DESCRIPTION: THE LAST TELEMETRY INPUT WORD WITH BIT 9 = 1 THAT WAS ACKNOWLEDGED. SEE APPENDIX D FOR DESCRIPTION OF EACH BIT (INPUT FORMAT 1). USED BY TASK 2.

B-15

. NAME: LBYTE

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...

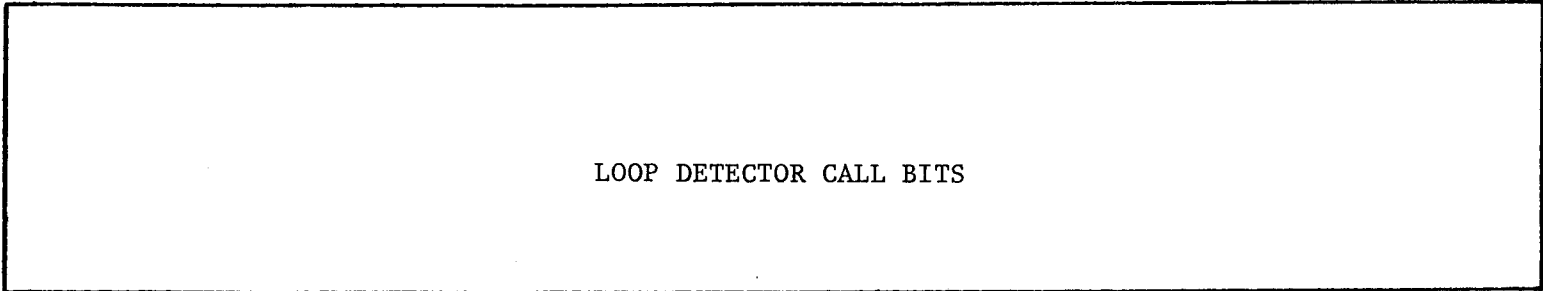


DESCRIPTION: CONSTANT USED TO MASK THE LOWER BYTE (8 BITS) OF A DATA WORD. USED BY VARIOUS TASKS.

. NAME: LOOPS

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...



LOOP DETECTOR CALL BITS

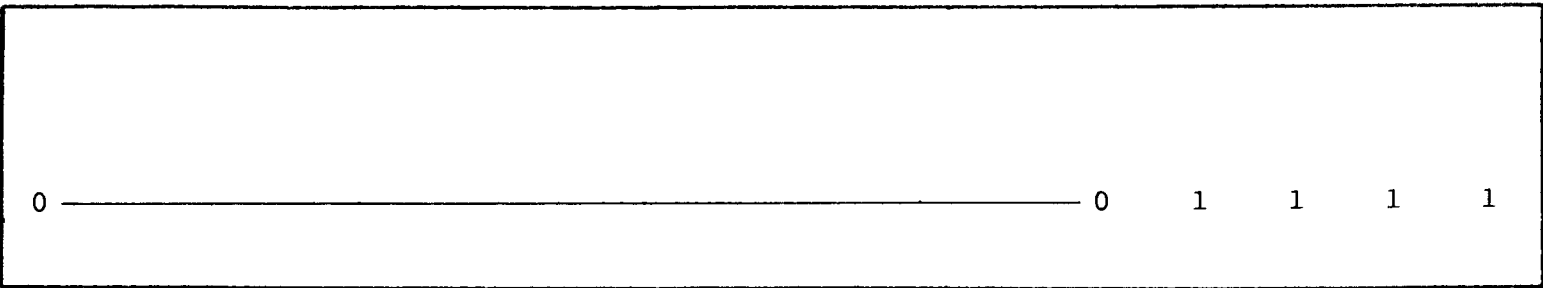
DESCRIPTION: LOOP DETECTOR ACTUATIONS THAT HAVE NOT BEEN ACKNOWLEDGED BY TASK 3 (EMERGENCY CONTROL). SEE APPENDIX C (INPUT Z+1) FOR BIT ASSIGNMENTS. SET BY TASK 2. USED BY TASK 3 TO DETERMINE PHASE EXTENSION DURING EMERGENCY CONTROL.

B-16

. NAME: LOW4M

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...



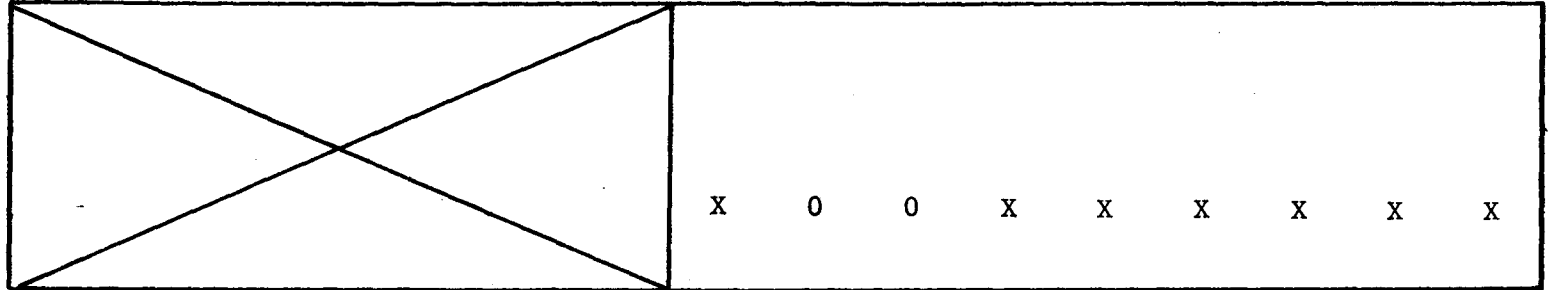
0 _____ 0 1 1 1 1

DESCRIPTION: CONSTANT USED TO MASK LOWER 4 BITS OF A DATA WORD. USED IN SUBROUTINE PTSUB.

. NAME: LRJCO

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...



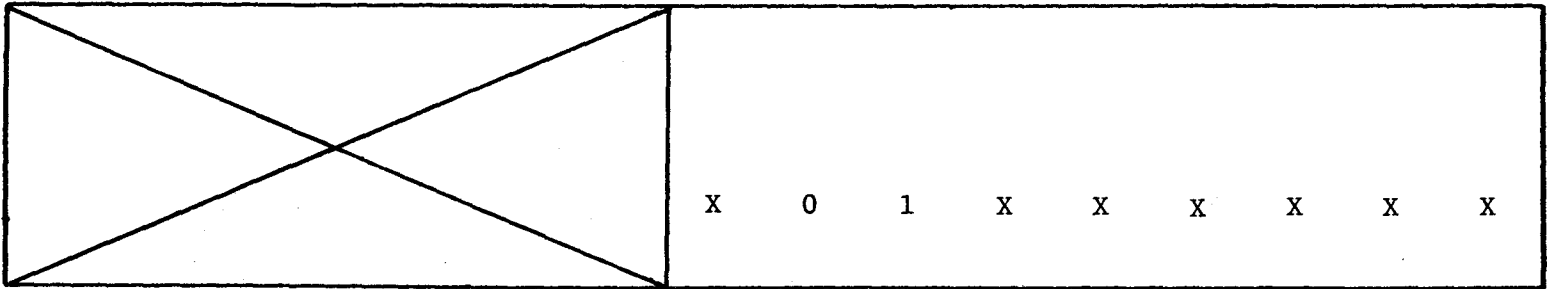
DESCRIPTION: THE LAST TELEMETRY INPUT WORD WITH BIT 9 = 0 THAT WAS REJECTED. SEE APPENDIX D FOR DESCRIPTION OF EACH BIT (INPUT FORMAT 0). USED BY TASK 2.

B-17

. NAME: LRJC1

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...

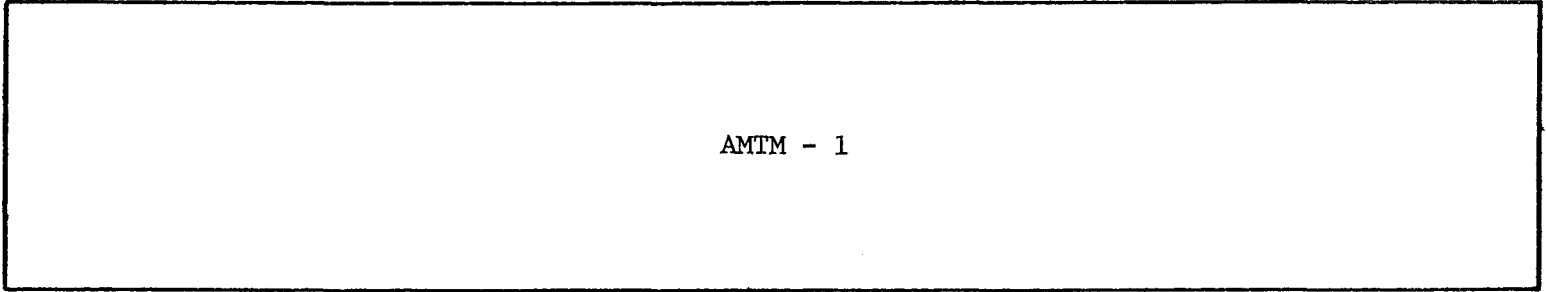


DESCRIPTION: THE LAST TELEMETRY INPUT WORD WITH BIT 9 = 1 THAT WAS REJECTED. SEE APPENDIX D FOR DESCRIPTION OF EACH BIT (INPUT FORMAT 1). USED BY TASK 2.

. NAME: MAMTM

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...



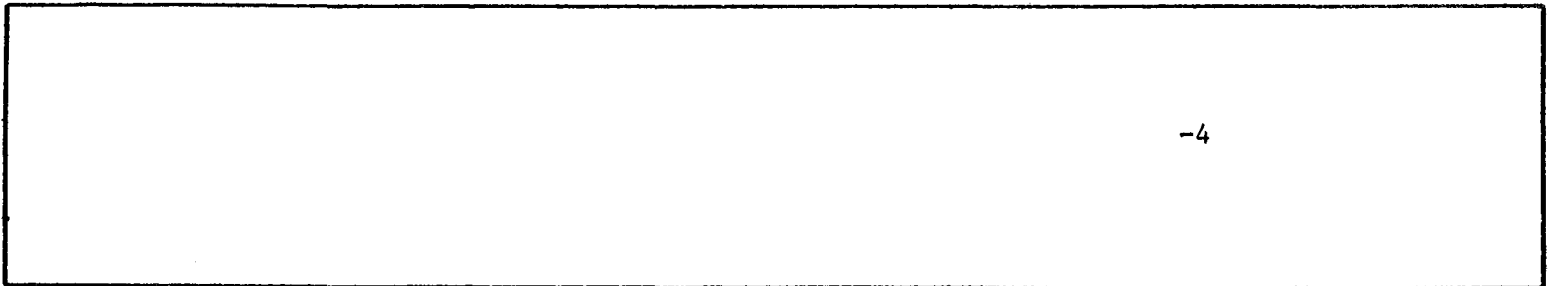
DESCRIPTION: ADDRESS -1 OF AMBER TIME TABLE (SEE AMTM). USED BY TASKS 2 AND 3.

B-18

. NAME: MHTSK

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...



DESCRIPTION: CONSTANT USED BY TASK CONTROL AND RESTART AS A LOOP COUNTER WHEN ADDRESSING THE VARIOUS TASK CONTROL BLOCKS (TIMER, STATE, ETC.).

. NAME: OWRDM

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...

0	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

DESCRIPTION: CONSTANT USED TO MASK 9 DATA BITS OF THE TELEMETRY OUTPUT WORDS.
USED IN TELEMETRY OUTPUT SECTION.

B-19

. NAME: PCINT

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...

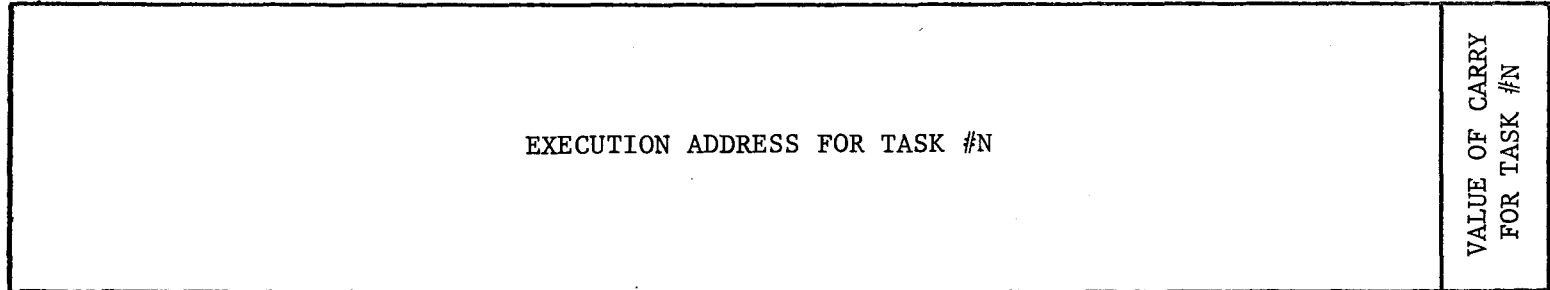
PC AT TIME OF INTERRUPT															
-------------------------	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

DESCRIPTION: ABSOLUTE LOCATION 0. CONTAINS THE VALUE OF THE PROGRAM COUNTER AFTER AN INTERRUPT. USED BY INTERRUPT ROUTINE TO DETERMINE ADDRESS TO CONTINUE INTERRUPTED PROGRAM. USED BY POWER FAIL INTERRUPT TO STORE FIRST INSTRUCTION AFTER A RESTART.

. NAME: PCSAV

SIZE: 4

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...



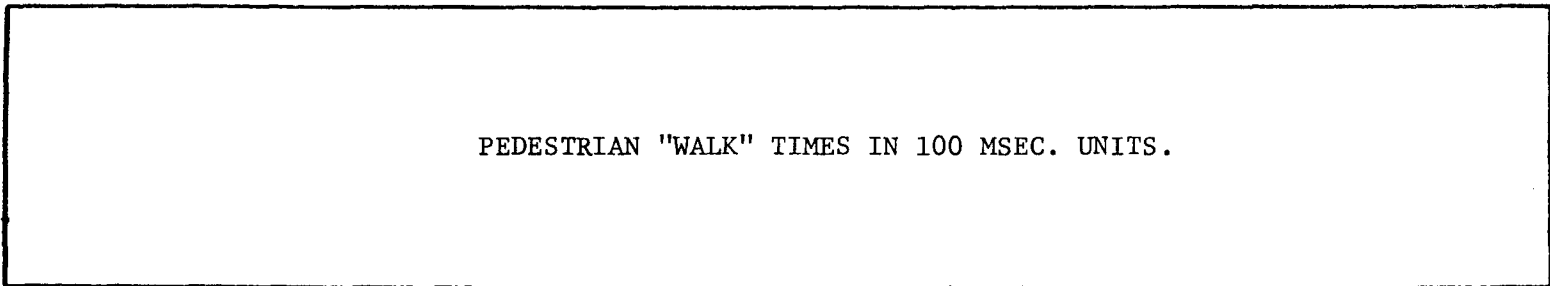
DESCRIPTION: WHEN TASK #N IS INTERRUPTED THE CURRENT VALUE OF THE PROGRAM COUNTER IS RETRIEVED FROM PCINT (LOC 0), SHIFTED LEFT TO OBTAIN THE CARRY BIT, AND STORED IN THE PCSAV ARRAY. WHEN TASK #N IS MADE READY FOR EXECUTION, THE PC AND CARRY VALUES ARE OBTAINED FROM PCSAV.

B-20

. NAME: PDMNT

SIZE: 9

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...

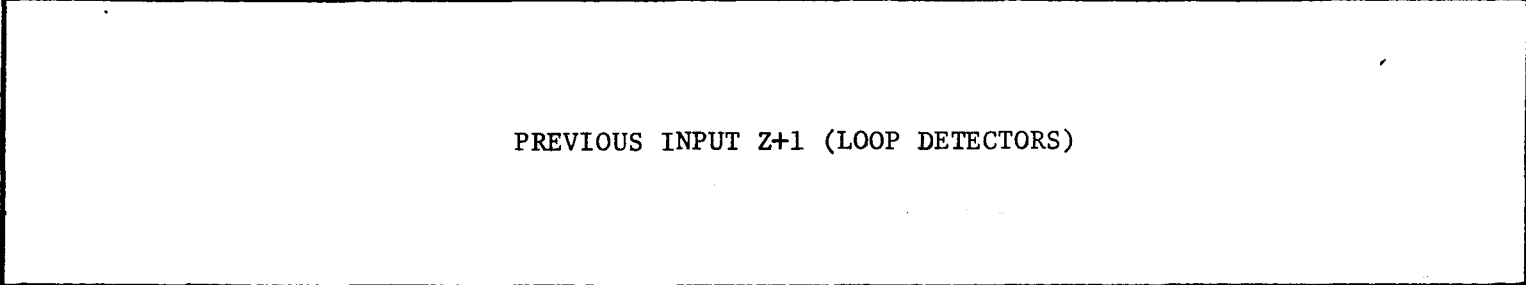


DESCRIPTION: WALK TIMES FOR EACH PEDESTRIAN MOVEMENT IN .1 SECOND UNITS. USED IN TASKS 2 AND 3. PDMNT+0 IS THE TIME FOR THE PEDESTRIAN MOVEMENT ASSOCIATED WITH GREEN BIT 0 (SEE APPENDIX C, OUTPUT V+2), PDMNT+1 FOR GREEN BIT 1, ETC.

. NAME: PRZP1

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...



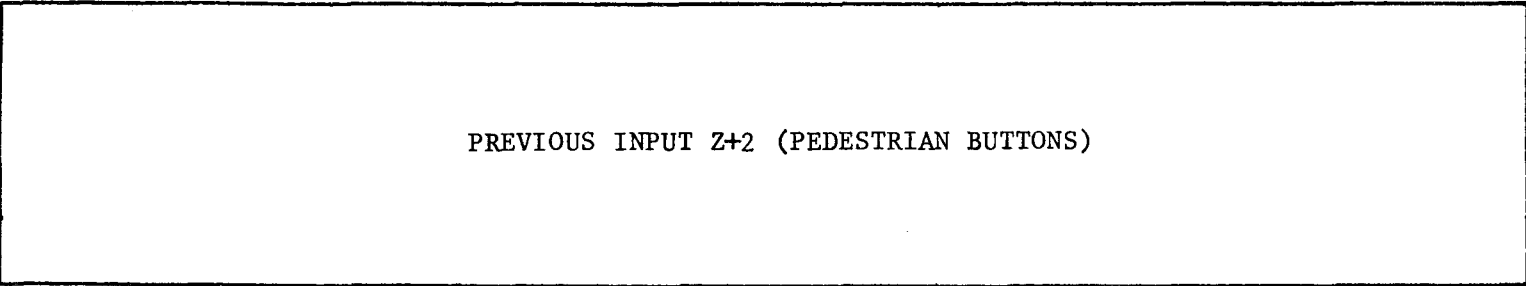
DESCRIPTION: THE LAST VALUE READ FOR INPUT WORD Z+1. USED TO COMPARE CURRENT WITH PREVIOUS DETECTOR STATES. USED BY TASKS 1 AND 2.

B-21

. NAME: PRZP2

SIZE: 1

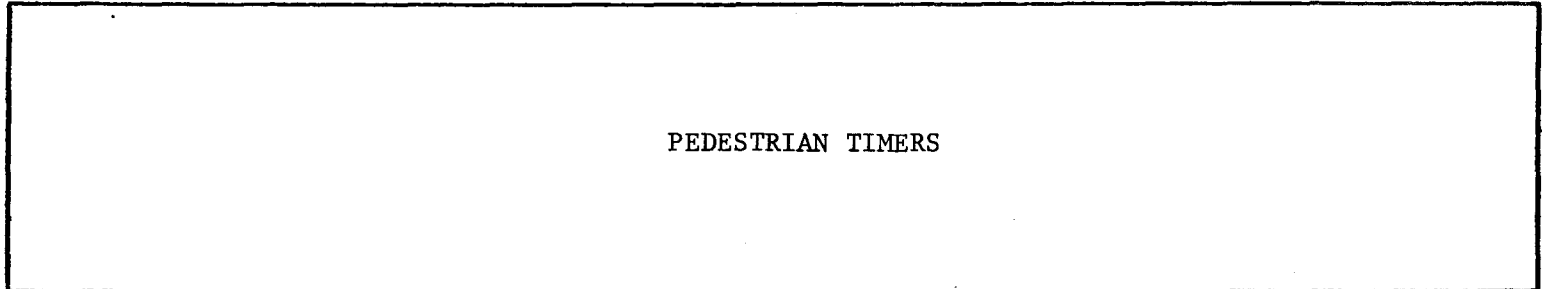
Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...



DESCRIPTION: THE LAST VALUE READ FOR INPUT WORD Z+2. USED TO COMPARE CURRENT WITH PREVIOUS PEDESTRIAN BUTTON STATES. USED BY TASK 1 AND 2.

. NAME: PTMR SIZE: 9

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...

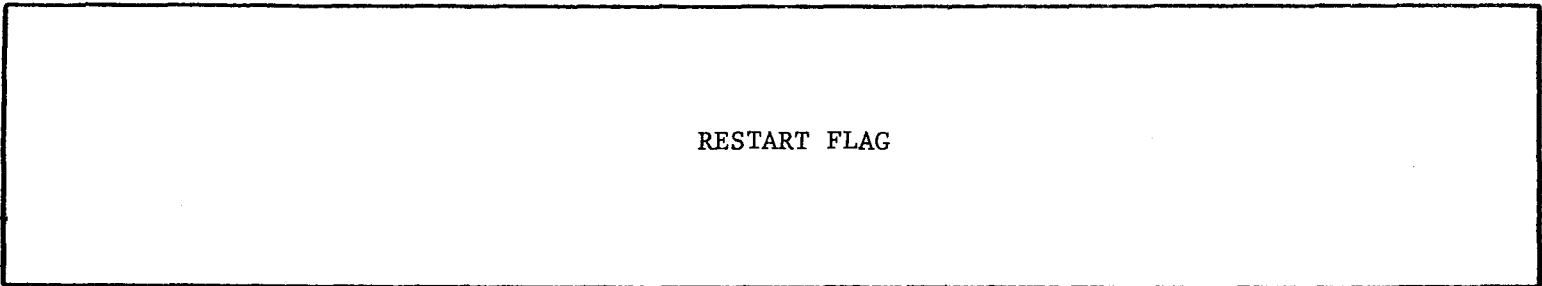


DESCRIPTION: COUNTERS USED TO TIME-OUT MINIMUM PEDESTRIAN WALK TIMES. USED IN TASK 2. INITIALIZED WITH THE VALUE FROM THE PDMNT ARRAY WHEN A "WALK" LIGHT IS TURNED ON. MUST IMMEDIATELY FOLLOW THE STMR ARRAY.

B-22

. NAME: RRFLG SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...

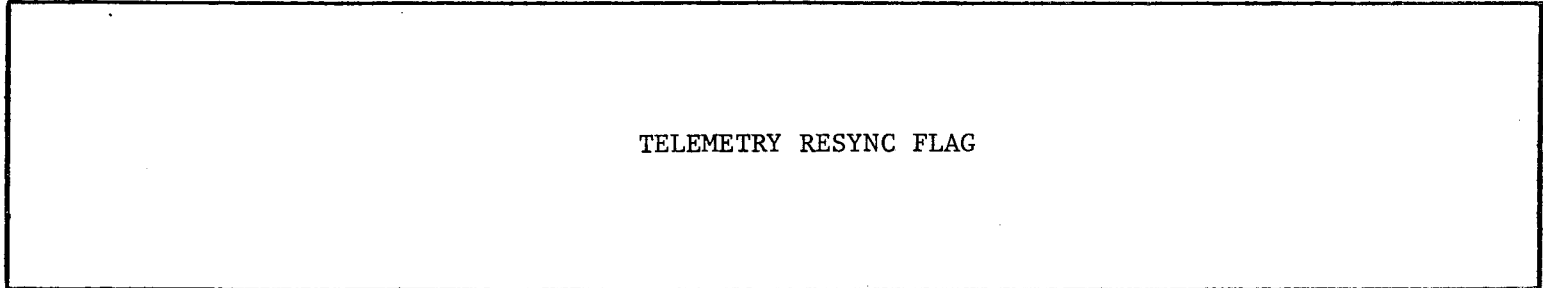


DESCRIPTION: SET BY RESTART ROUTINE WHENEVER A RESTART OCCURS. USED BY TELEMETRY OUTPUT TO SET THE RESTART BIT IN TELEMETRY OUTPUT WORD.

. NAME: RSFLG

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...



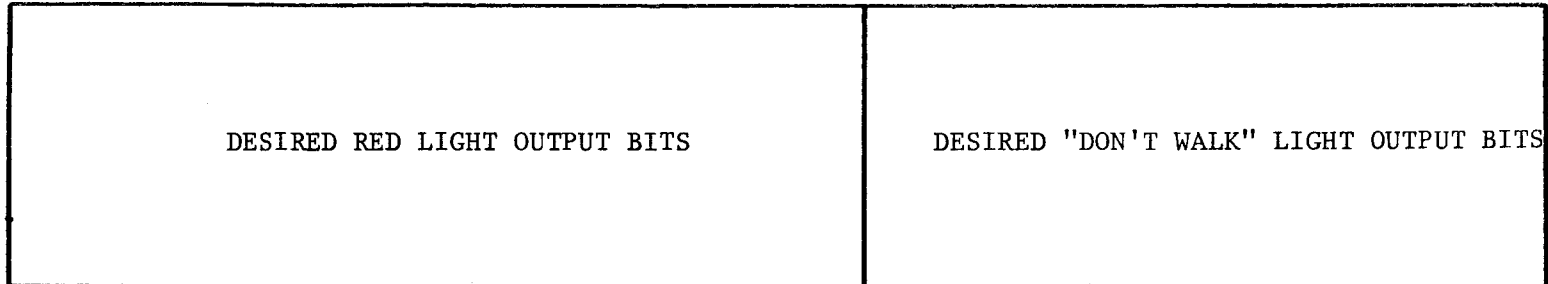
DESCRIPTION: AN INTERNAL FLAG USED BY TELEMETRY OUTPUT SECTION FOR TELEMETRY RESYNC PURPOSES.
0 = NO RESYNC. 1 = SEND RESYNC REQUEST VIA TELEMETRY, 2 = SEND CONTINUOUS MARK
(1 BIT).

B-23

. NAME: RSTA

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...



DESCRIPTION: DESIRED VALUE OF RED AND DON'T WALK LIGHTS FOR EACH MOVEMENT. 1 = RED/DON'T WALK LIGHT ON,
0 = RED/DON'T WALK LIGHT OFF. USED BY TASK 2 AND SUBROUTINE STFLS.

. NAME: SCEDF SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...

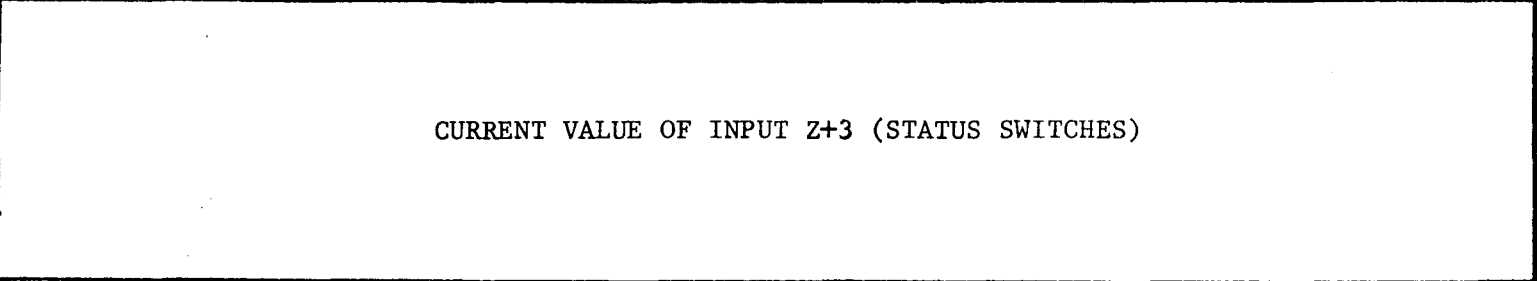


DESCRIPTION: A FLAG THAT IS SET WHENEVER THE TASK SCHEDULER IS ENTERED AND CLEARED WHEN A TASK IS EXECUTING. CHECKED BY INTERRUPT ROUTINE TO DETERMINE IF REGISTERS NEED TO BE SAVED.

B-24

. NAME: SGSTA SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...

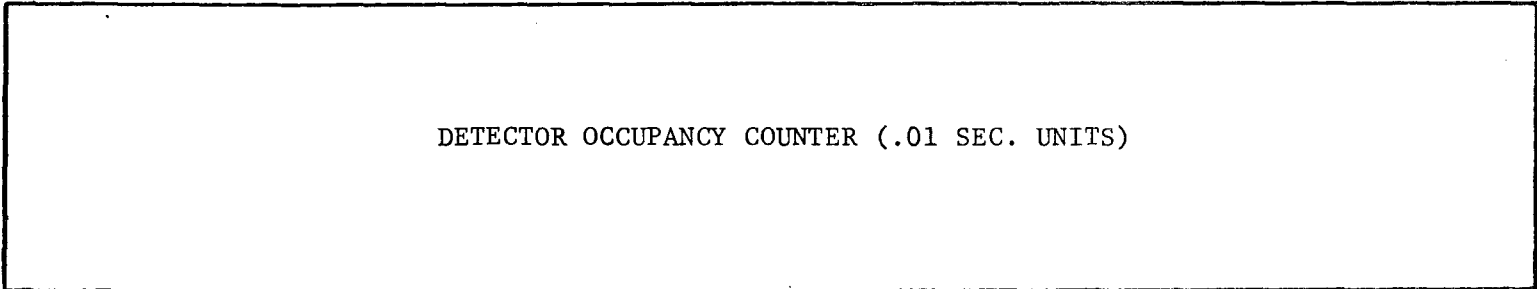


DESCRIPTION: VALUE THAT WAS READ FROM INPUT WORD Z+3. USED BY TASK 2.

. NAME: SPCEL

SIZE: 14

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...



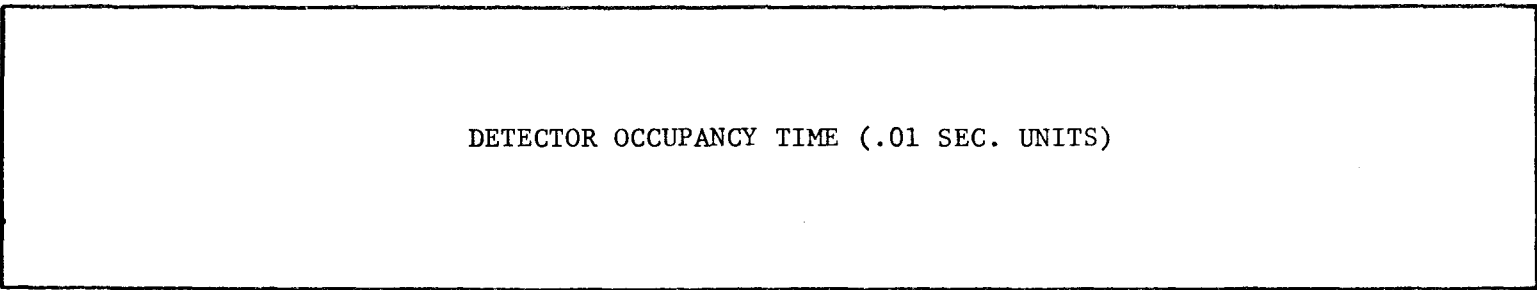
DESCRIPTION: A COUNTER USED BY TASK 0 TO ACCUMULATE AN INDIVIDUAL VEHICLE OCCUPANCY TIME OVER EACH SPEED DETECTOR (INPUT Z). INCREMENTED EVERY .01 SEC. WHEN THE DETECTOR IS OCCUPIED, TRANSFERRED TO SPTAB ARRAY AND ZEROED WHEN THE DETECTOR BECOMES UNOCCUPIED.

B-25

. NAME: SPTAB

SIZE: 14

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...

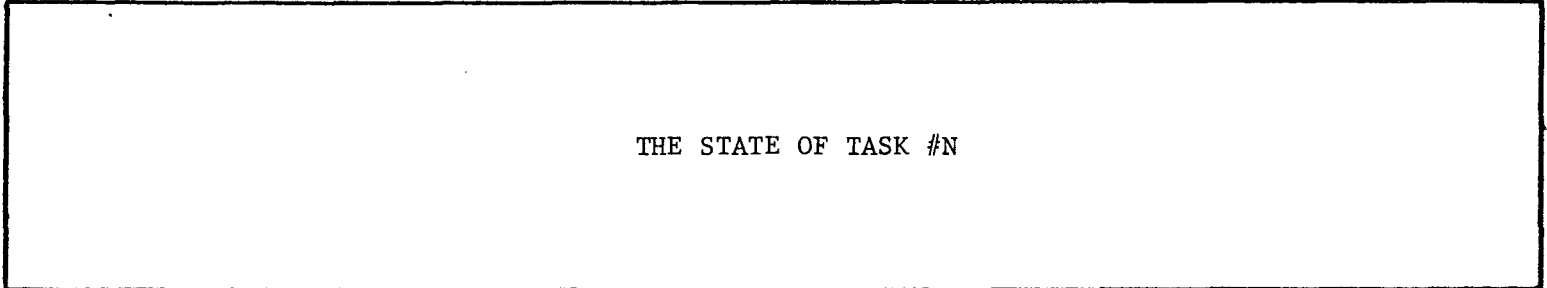


DESCRIPTION: THE TIME IN .01 SEC. UNITS THAT THE SPEED DETECTOR WAS OCCUPIED BY THE PREVIOUS VEHICLE. SET BY TASK 0. USED BY TASK 1 TO COMPUTE THE AVERAGE SPEED OVER A GROUP OF ADJACENT SPEED DETECTORS.

. NAME: STATE

SIZE: 4

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...



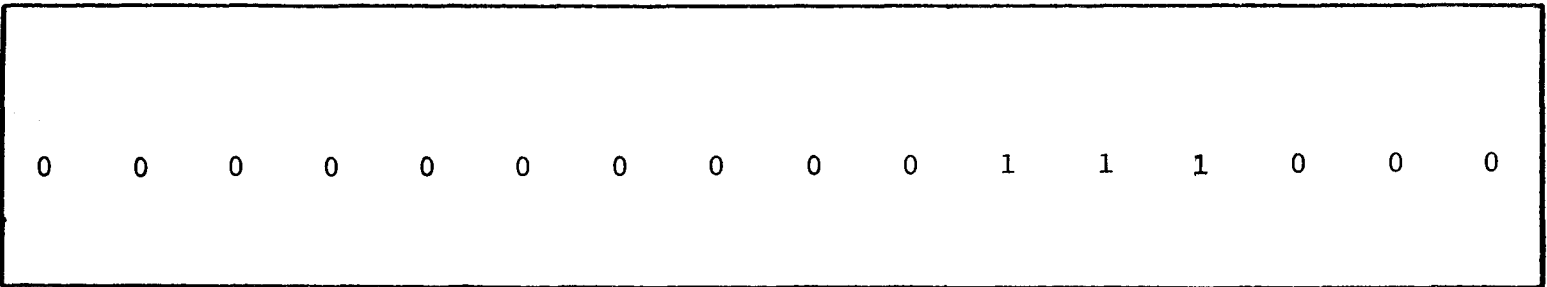
DESCRIPTION: ≤ 1 INDICATES TASK IS SUSPENDED (I.E. WAITING FOR TASK TIMER TO EXPIRE). > 1 INDICATES TASK IS PENDING EXECUTION. INCREMENTED BY TIMER CONTROL WHEN TASK TIMER EXPIRES, DECREMENTED BY SUBROUTINE WAIT WHEN CALLED BY A TASK FOR COMPLETION.

B-26

. NAME: STBTS

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...

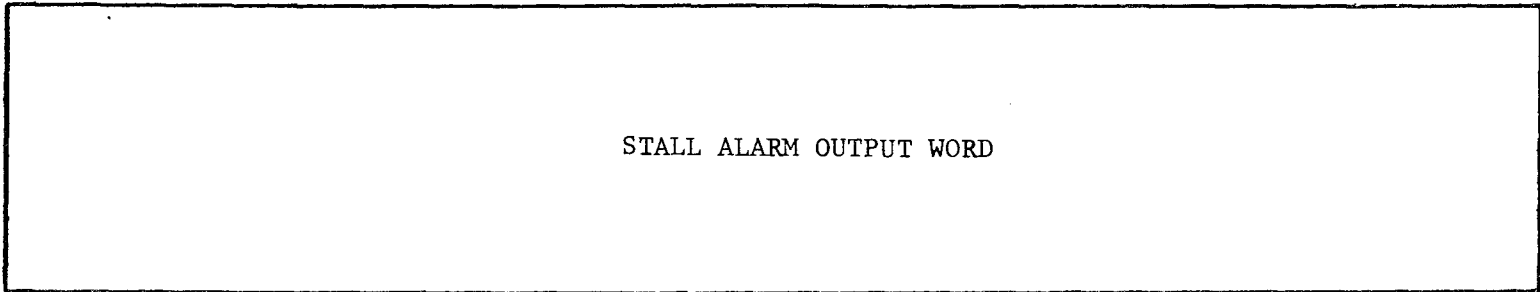


DESCRIPTION: CONSTANT CONTAINING TELEMETRY OUTPUT START/STOP BITS. USED IN TELEMETRY OUTPUT SECTION.

. NAME: STLOT

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...



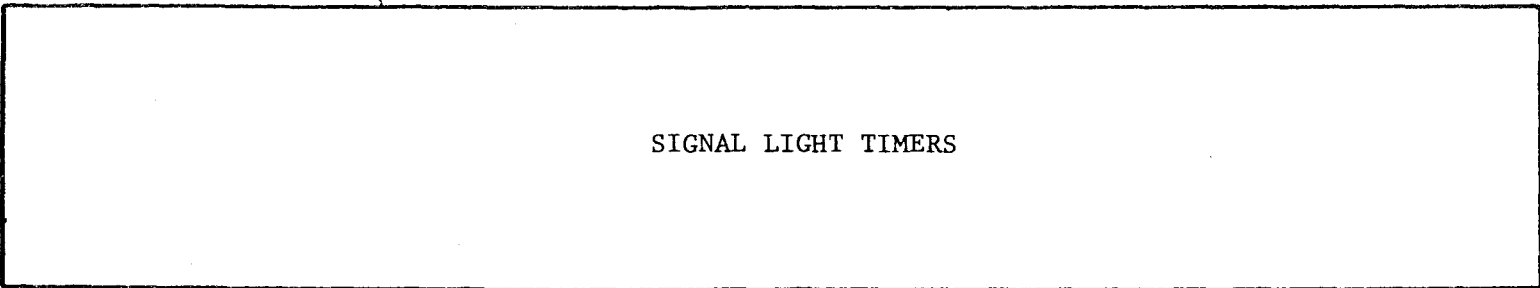
DESCRIPTION: OUTPUT WORD USED IN INTERRUPT AND RESTART ROUTINES TO PULSE THE STALL ALARM.
USED IN CHECKOUT ONLY.

B-27

. NAME: STMR

SIZE: 9

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...

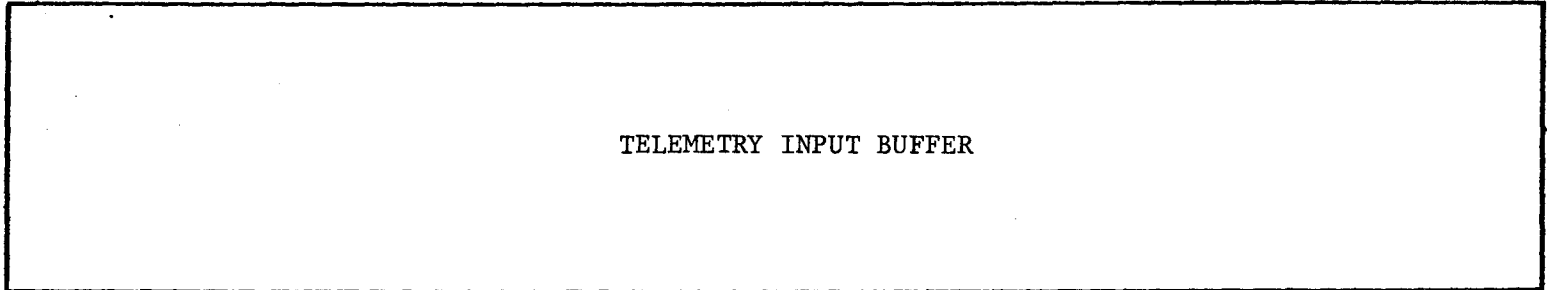


DESCRIPTION: COUNTERS USED TO TIME-OUT ABSOLUTE MINIMUM GREEN TIMES AND AMBER TIMES. USED IN
TASK 2. INITIALIZED WITH A MINIMUM GREEN TIME WHEN A GREEN LIGHT IS TURNED ON.
INITIALIZED WITH AN AMBER TIME FROM AMTM WHENEVER A GREEN LIGHT IS TURNED OFF.

. NAME: TIBUF

SIZE: 3

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...



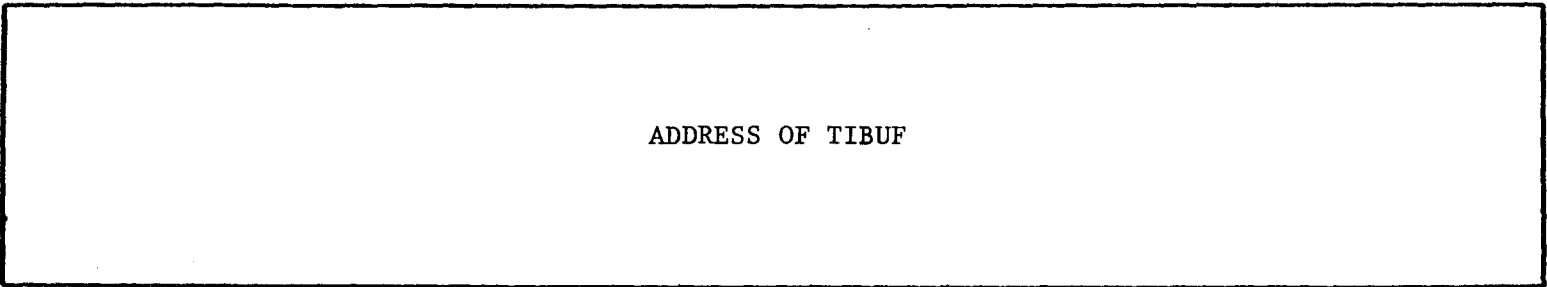
DESCRIPTION: CIRCULAR BUFFER USED TO HOLD TELEMETRY INPUT WORDS. ENTRIES ARE ADDED BY TELEMETRY INPUT SECTION AND REMOVED BY TASK 2.

B-28

. NAME: TIFIR

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...

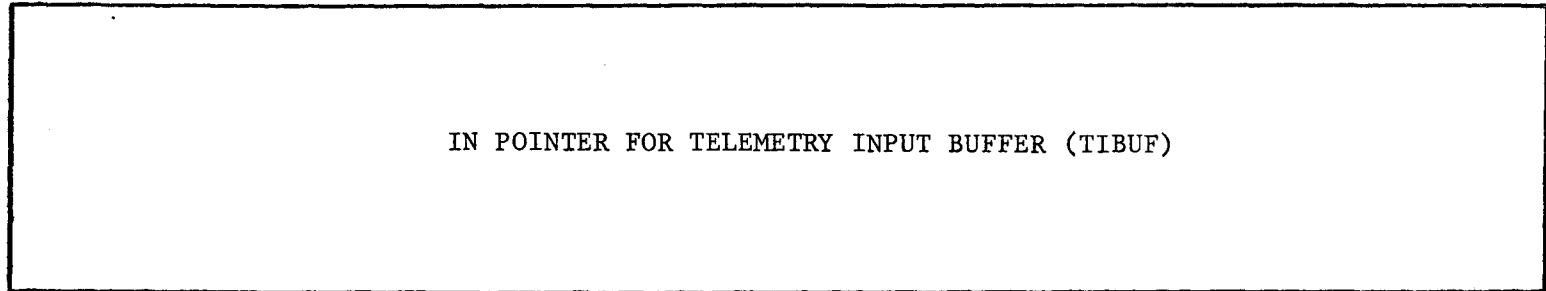


DESCRIPTION: POINTER TO FIRST WORD OF TELEMETRY INPUT BUFFER. USED TO RECYCLE TIIN AND TIOUT POINTERS BY SUBROUTINES PUT AND GET.

. NAME: TIIN

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...



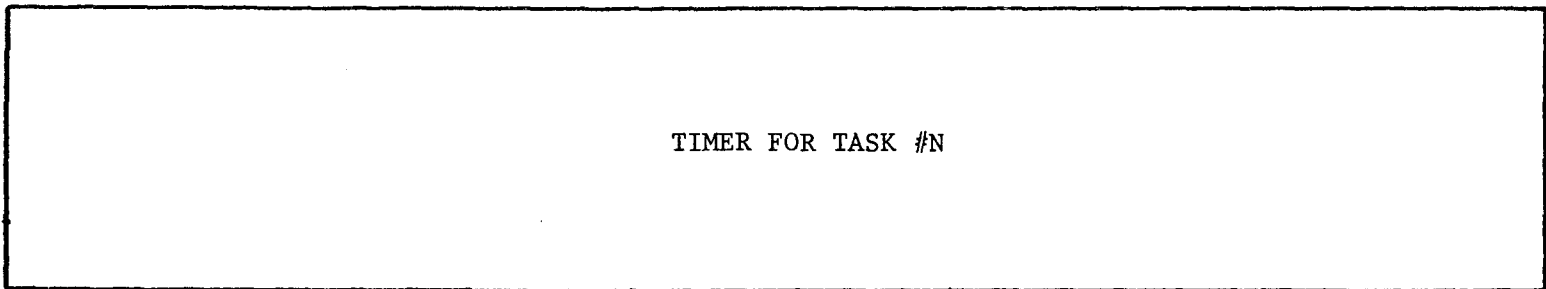
DESCRIPTION: ADDRESS OF NEXT AVAILABLE SLOT IN TIBUF FOR STORING TELEMETRY INPUT WORD. INCREMENTED EVERY TIME A NEW INPUT WORD IS STORED IN TIBUF BY SUBROUTINE PUT WHEN CALLED FROM TELEMETRY INPUT SECTION.

B-29

. NAME: TIMER

SIZE: 4

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...

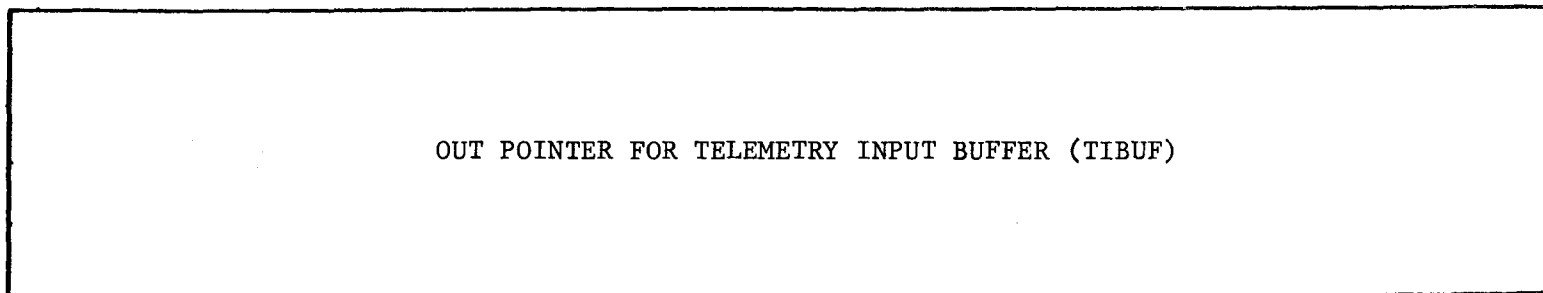


DESCRIPTION: A COUNTER THAT IS DECREMENTED EVERY MILLISECOND BY TASK TIMER CONTROL. WHEN IT COUNTS DOWN TO ZERO, IT IS RESET TO THE VALUE IN CYCLE AND THE TASK STATE IS INCREMENTED, MARKING TASK #N READY FOR EXECUTION.

. NAME: TIOUT

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...



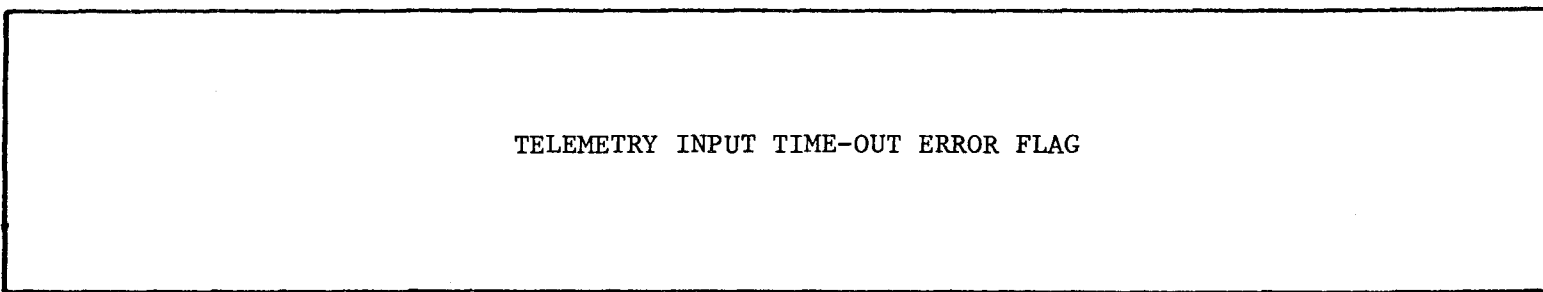
DESCRIPTION: ADDRESS OF NEXT WORD TO BE REMOVED FROM TELEMETRY INPUT BUFFER. INCREMENTED EVERY TIME A WORD IS REMOVED FROM BUFFER BY SUBROUTINE GET WHEN CALLED FROM TASK 2.

B-30

. NAME: TMERF

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...

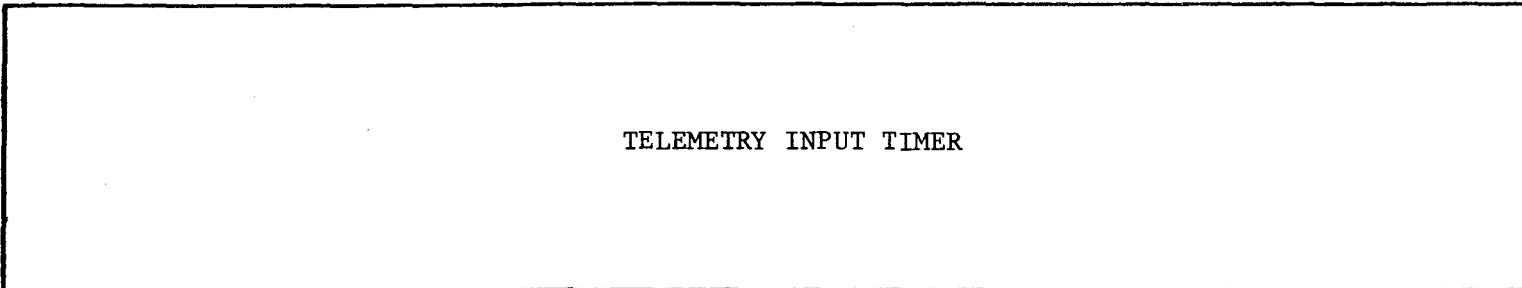


DESCRIPTION: AN ERROR FLAG SET BY TASK 2 IF A VALID INPUT FORMAT 1 WORD IS NOT RECEIVED FROM TELEMETRY INPUT IN ANY 2 SECOND PERIOD. USED BY TASK 3 TO DETERMINE IF EMERGENCY CONTROL SHOULD BE USED.

. NAME: TMINT

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...



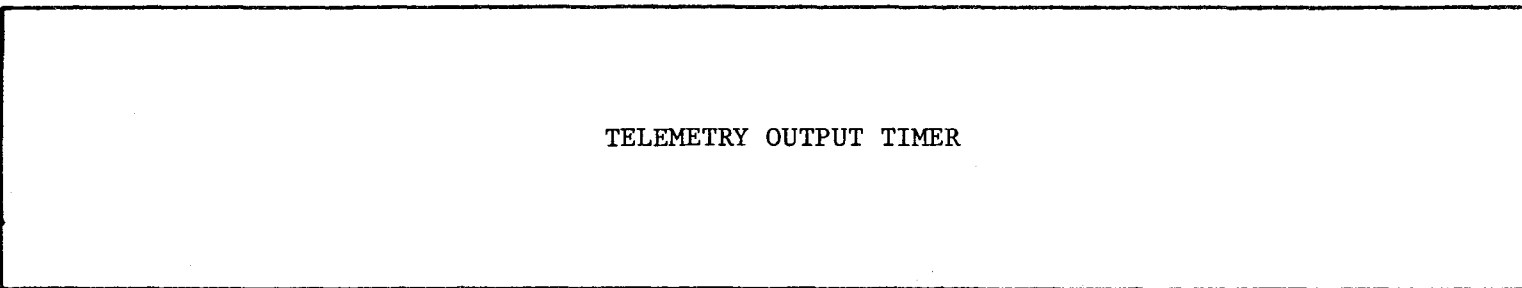
DESCRIPTION: A COUNTER USED BY TELEMETRY INPUT ROUTINE TO TIME-OUT INTERVALS BETWEEN BITS, COUNT # OF SAMPLES OF INPUT BITS, ETC.

B-31

. NAME: TMOUT

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...

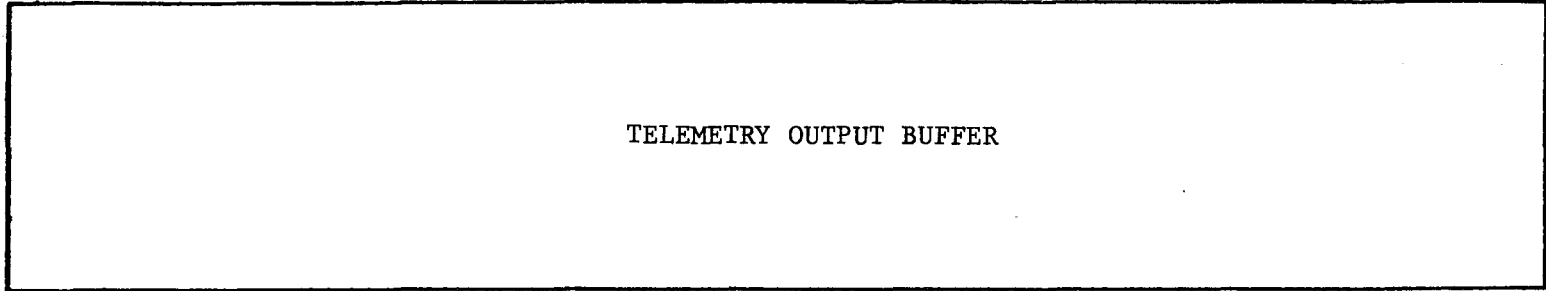


DESCRIPTION: A COUNTER USED BY TELEMETRY OUTPUT ROUTINE TO TIME OUT INTERVALS BETWEEN SUCCESSIVE OUTPUT BITS. IT IS SET TO 7 BEFORE EACH OUTPUT BIT AND COUNTED DOWN EVERY MILLISECOND.

. NAME: TOBUF

SIZE: 17

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...



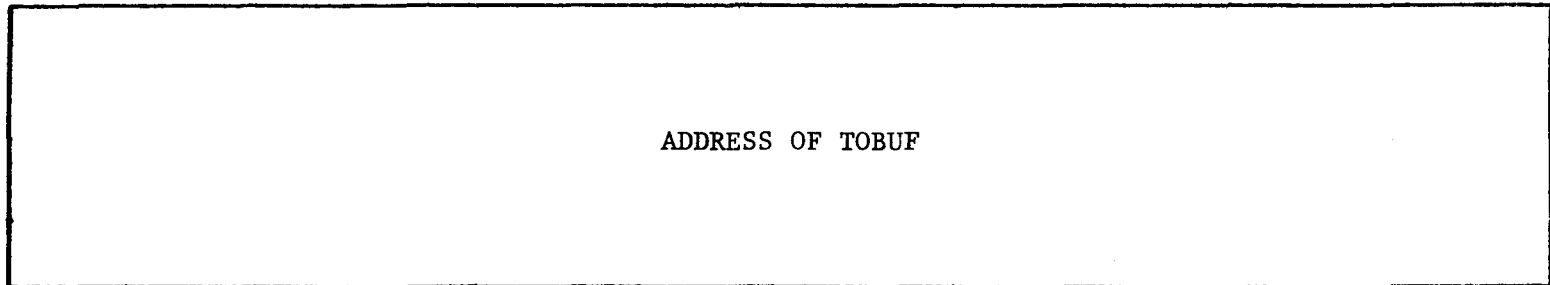
DESCRIPTION: CIRCULAR BUFFER USED TO HOLD THE QUEUE OF TELEMETRY OUTPUT WORDS. ENTRIES ARE ADDED BY TASKS 1 AND 2 WHEN THEY CALL SUBROUTINE PTSUB. TELEMETRY OUTPUT ROUTINE REMOVES WORDS FROM BUFFER.

B-32

. NAME: TOFIR

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...

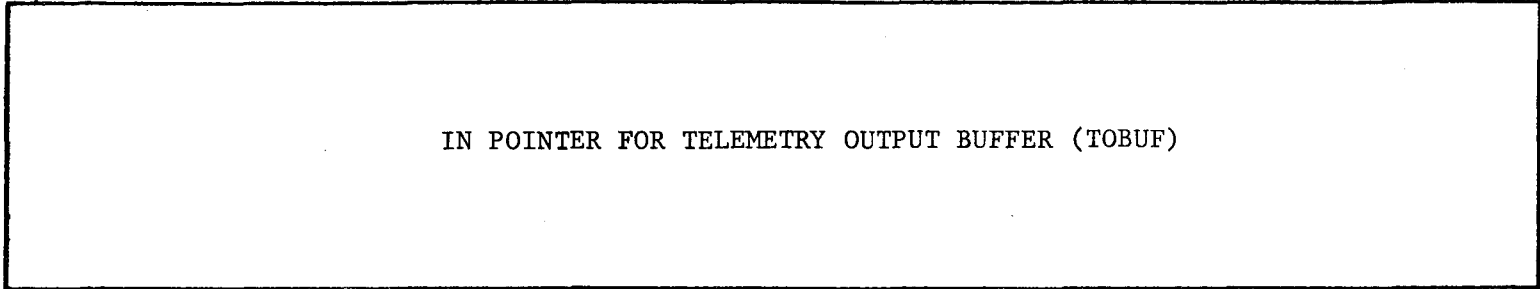


DESCRIPTION: POINTER TO FIRST WORD OF TELEMETRY OUTPUT BUFFER. USED TO RECYCLE TOIN AND TOOUT POINTERS BY SUBROUTINES PUT AND GET.

. NAME: TOIN

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...



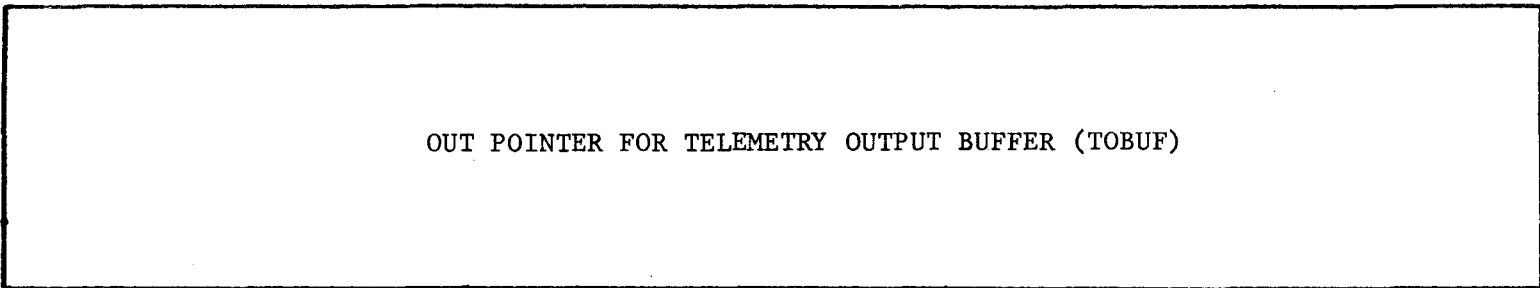
DESCRIPTION: ADDRESS OF NEXT AVAILABLE SLOT IN TOBUF FOR STORING TELEMETRY OUTPUT WORD. INCREMENTED EVERY TIME A NEW OUTPUT WORD IS STORED IN TOBUF BY SUBROUTINE PUT WHEN CALLED FROM TASKS 1 OR 2.

B-33

. NAME: TOOUT

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...

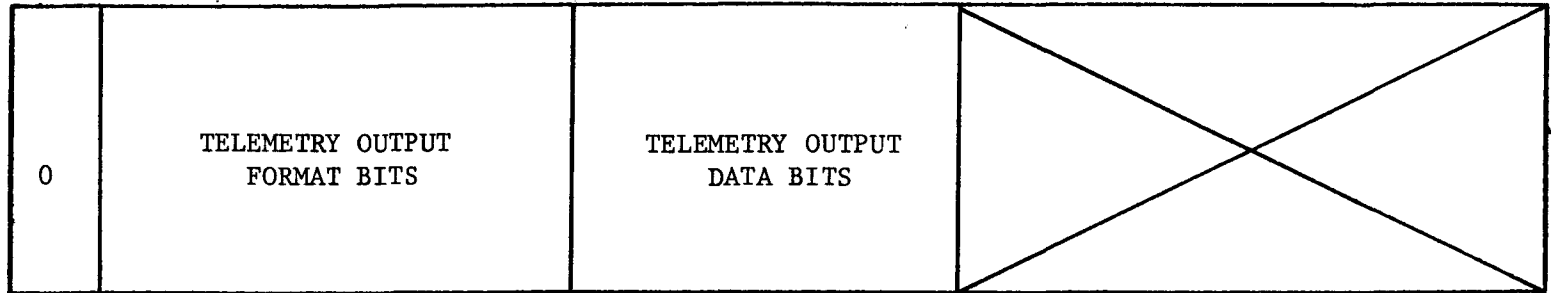


DESCRIPTION: ADDRESS OF NEXT WORD TO BE REMOVED FROM TELEMETRY OUTPUT BUFFER. INCREMENTED EVERY TIME A WORD IS REMOVED FROM BUFFER BY SUBROUTINE GET WHEN CALLED BY TELEMETRY OUTPUT SECTION.

. NAME: TOUTW

SIZE: 18

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...



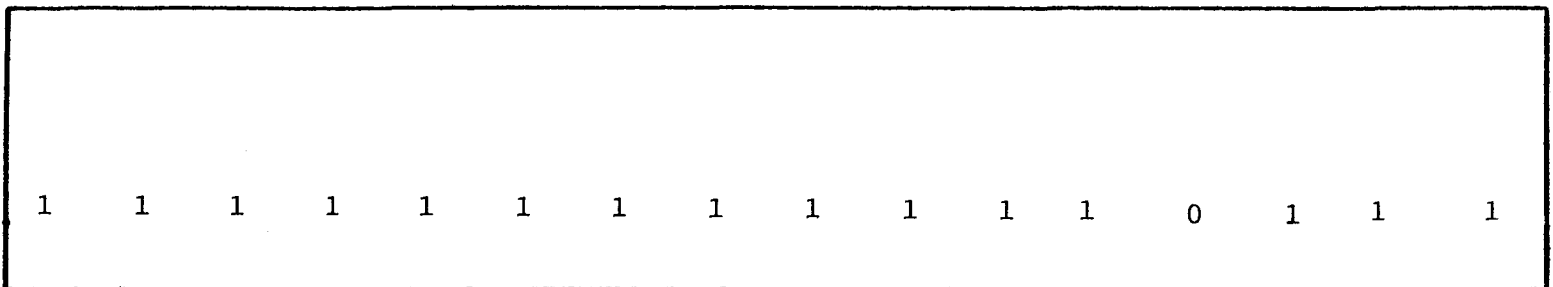
DESCRIPTION: TELEMETRY OUTPUT FORMAT TABLE. BITS 1-5 ARE THE ACTUAL FORMAT BITS. BITS 6-9 ARE USED TO HOLD THE LAST DATA BITS THAT WERE OUTPUT FOR EACH FORMAT. USED BY TASKS 1 AND 2 (SUBROUTINE PTSUB) AND BY TELEMETRY OUTPUT ROUTINE.

B-34

. NAME: UMASK

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...

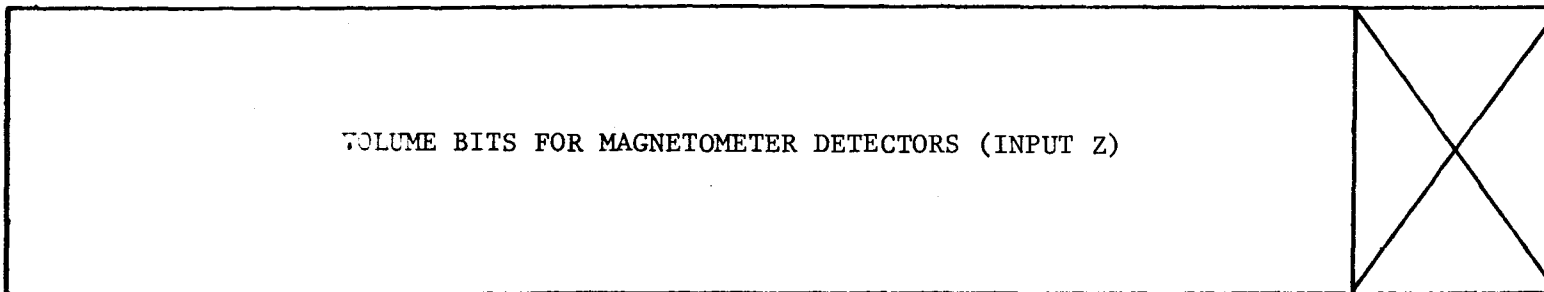


DESCRIPTION: CONSTANT USED TO UNMASK THE REAL-TIME CLOCK INTERRUPT AND MASK ALL OTHER DEVICES. IT IS USED IN TASK SCHEDULER TO ALLOW REAL-TIME CLOCK INTERRUPT DURING TASK EXECUTION.

. NAME: VOLUM

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...



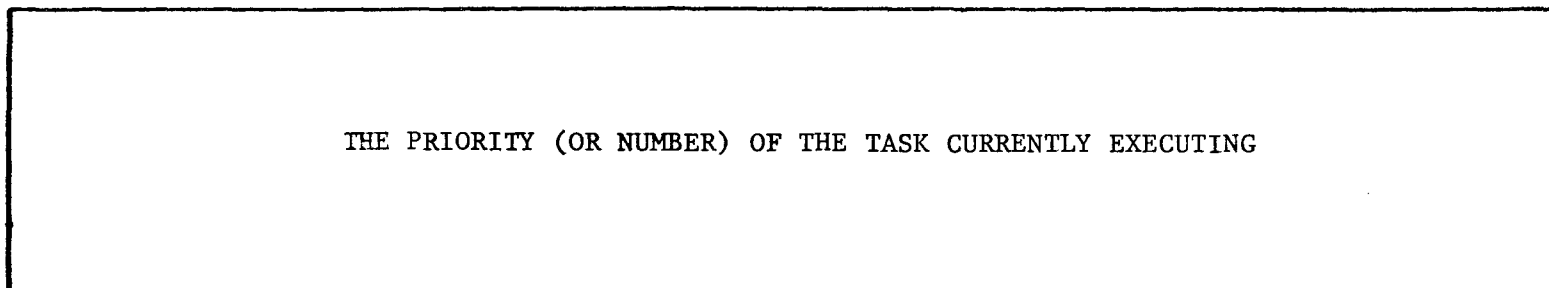
DESCRIPTION: A VOLUME BIT IS LOGICALLY OR'ED INTO THIS WORD FOR EACH 1 TO 0 TRANSITION OVER EACH MAGNETOMETER DETECTOR IN INPUT GROUP Z BY TASK 0. IT IS USED BY TASK 1 TO SEND VOLUME BITS VIA TELEMETRY AND TO DETERMINE WHEN TO UPDATE AVERAGE SPEEDS.

B-35

. NAME: XEQPR

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...

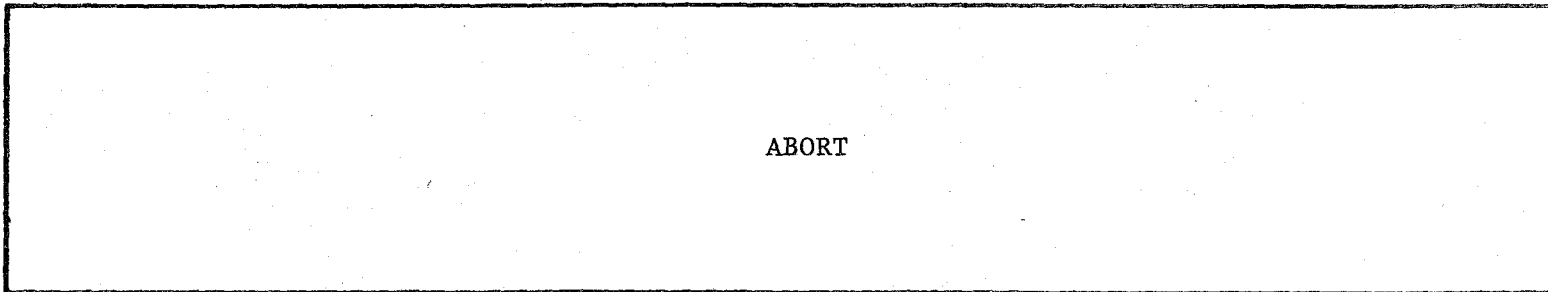


DESCRIPTION: AN INDEX VALUE USED BY TASK CONTROL TO ADDRESS TASK CONTROL BLOCKS. WHENEVER A TASK IS EXECUTING ITS INDEX WILL BE IN THIS WORD. THE VALUES ARE: TASK 0 = -4, TASK 1 = -3, TASK 2 = -2, TASK 3 = -1, IDLE JOB = 0.

• NAME: ZABRT

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...



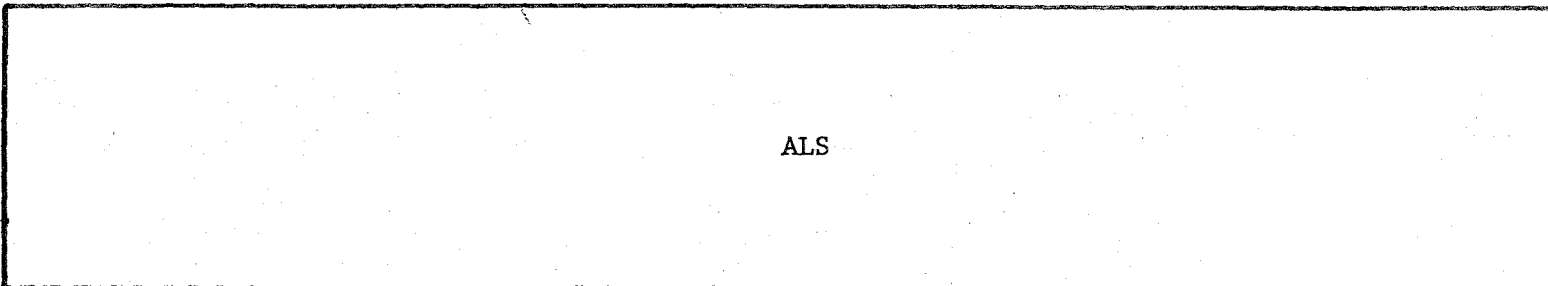
DESCRIPTION: ADDRESS OF SUBROUTINE ABORT. ALL OF THE FOLLOWING BEGIN WITH "Z" AND ARE TRANSFER VECTORS TO COMMON SUBROUTINES. SUBROUTINE ABORT IS CALLED WHENEVER A FATAL ERROR IS DETECTED.

B-36

• NAME: ZALS

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...

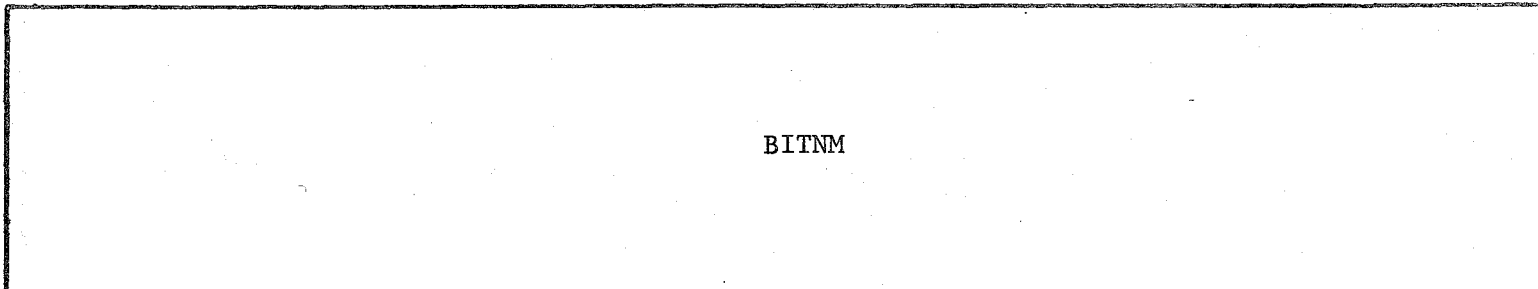


DESCRIPTION: ADDRESS OF SUBROUTINE ALS. PERFORM CIRCULAR LEFT SHIFT ON WORD IN AC0. SHIFT COUNT IN AC2.

NAME: ZBITN

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...



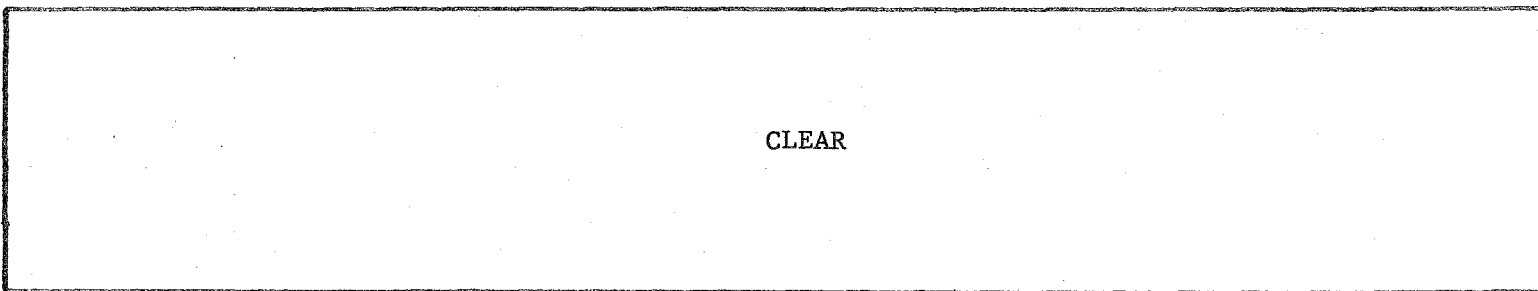
DESCRIPTION: POINTER TO SUBROUTINE BITNM. SHIFT ACO LEFT UNTIL THE FIRST "1" BIT IS REMOVED. ADD THE NUMBER OF SHIFTS REQUIRED TO ACCOMPLISH THIS TO AC2. THIS EFFECTIVELY GIVES THE BIT NUMBER OF THE LEFTMOST BIT SET IN ACO.

B-37

NAME: ZCLER

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...

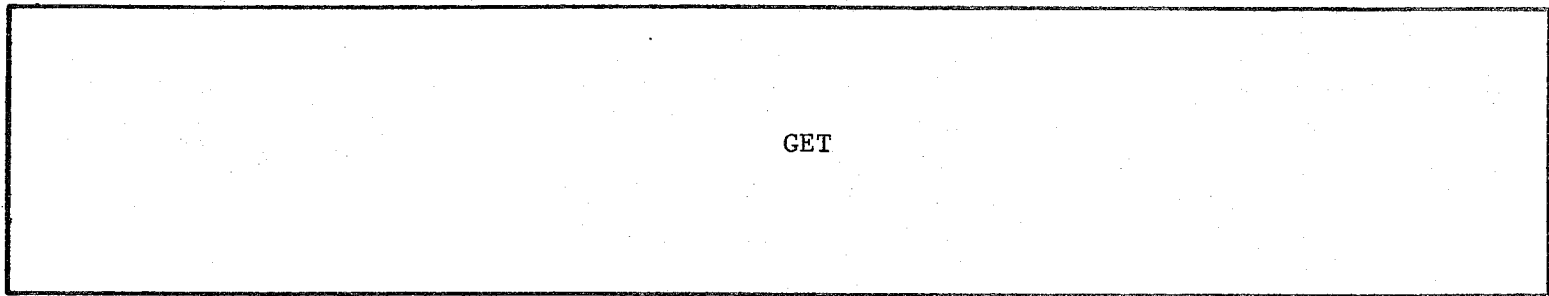


DESCRIPTION: POINTER TO SUBROUTINE CLEAR. CLEAR A GROUP OF VARIABLES WHOSE ADDRESSES APPEAR IN A LIST POINTED TO BY THE CALLING ARGUMENT. THE LIST IS TERMINATED BY A ZERO WORD.

. NAME: ZGET

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...



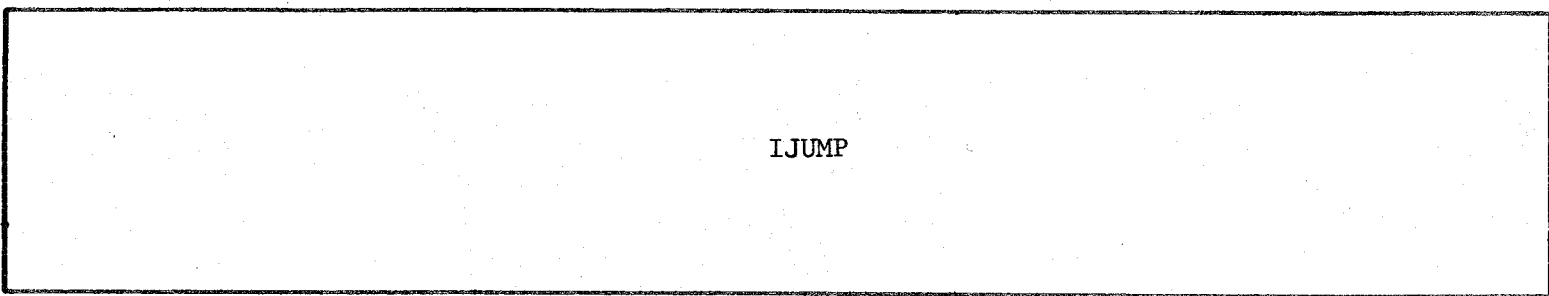
DESCRIPTION: POINTER TO SUBROUTINE GET. FETCH A WORD FROM A CIRCULAR BUFFER POINTED TO BY CALLING ARGUMENT. RETURN WORD IN ACO. TAKE ABNORMAL RETURN IF BUFFER EMPTY.

B-38

. NAME: ZIJMP

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...

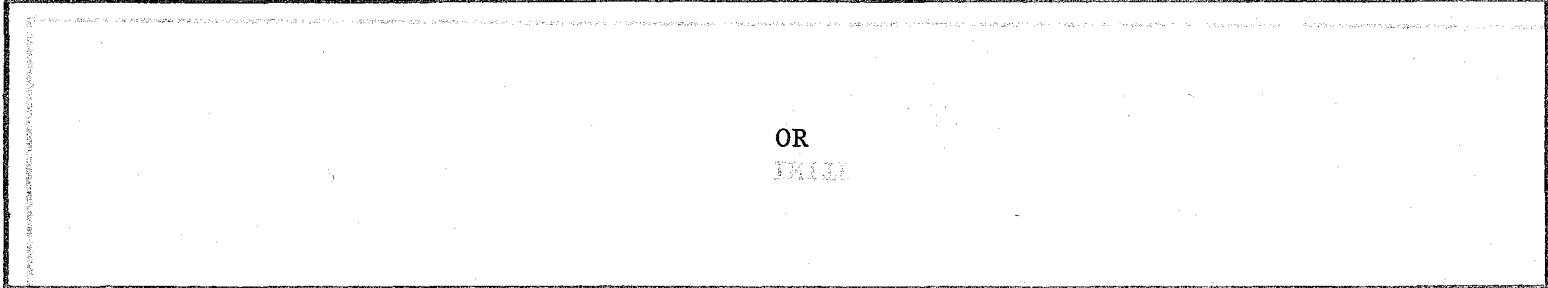


DESCRIPTION: POINTER TO SUBROUTINE IJUMP. PERFORM AN INDEXED JUMP USING AC2 ON THE LIST OF ADDRESSES IMMEDIATELY FOLLOWING THE CALL.

NAME: ZOR

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...



OR
FIELD

DESCRIPTION: POINTER TO SUBROUTINE OR. FORM LOGICAL OR OF QUANTITIES IN ACO AND AC1. LEAVE RESULT IN AC1. COMPLEMENT ORIGINAL ACO.

NAME: ZOR

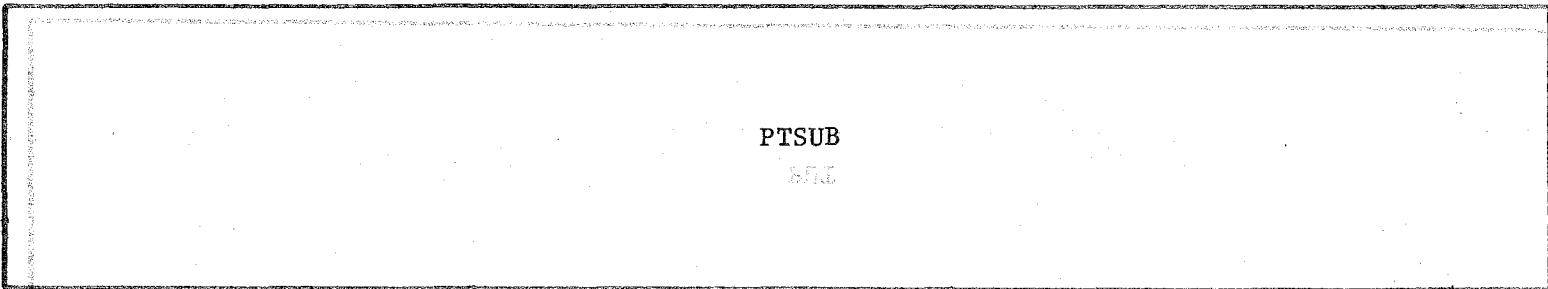
SIZE: 1

B-39

NAME: ZPTSB

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...



PTSUB
FIELD

DESCRIPTION: POINTER TO SUBROUTINE PTSUB. MERGE 4 DATA BITS AND FORMAT BITS AND PLACE RESULT IN TELEMETRY OUTPUT BUFFER EITHER UNCONDITIONALLY (ACO = 0) OR ONLY IF CHANGED

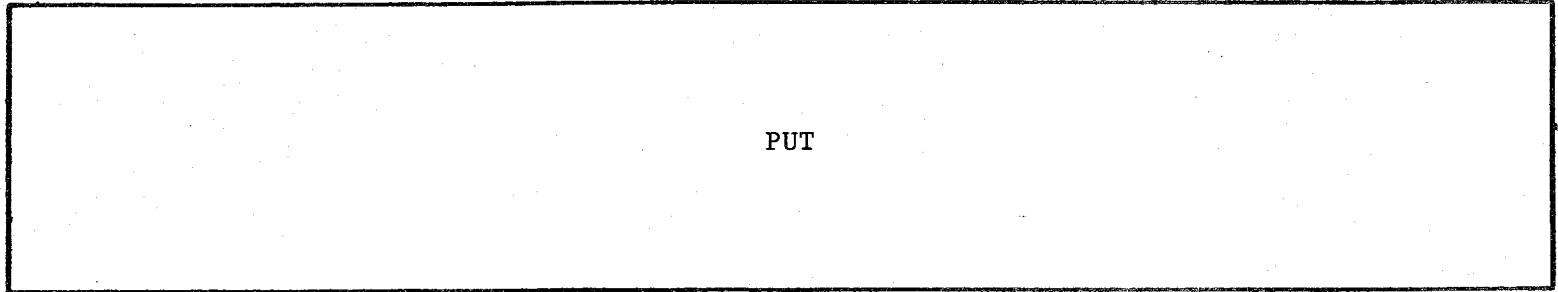
NAME: ZPTSB FROM LAST OUTPUT (ACO = -1).

SIZE: 1

NAME: ZPUT

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...



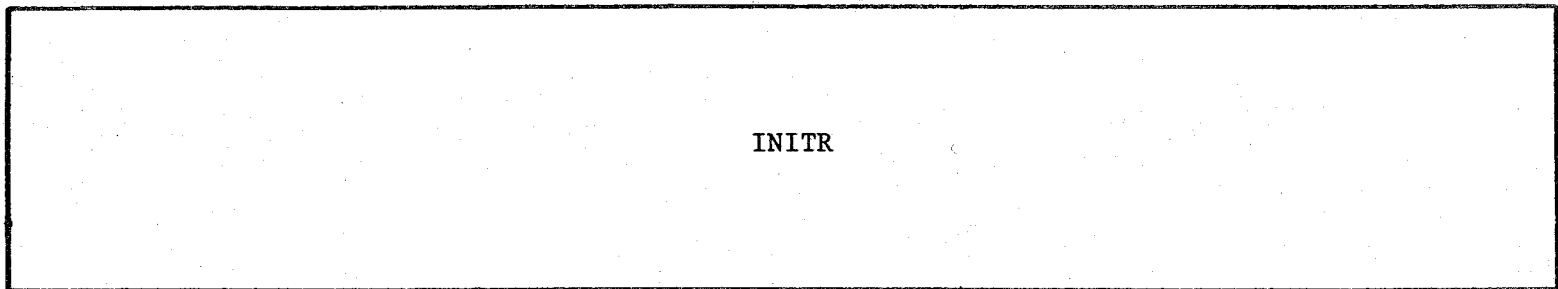
DESCRIPTION: POINTER TO SUBROUTINE PUT. PLACE WORD GIVEN BY SECOND ARGUMENT INTO CIRCULAR BUFFER POINTED TO BY FIRST ARGUMENT. TAKE ABNORMAL RETURN IF BUFFER FULL.

B-40

NAME: ZRSTR

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...

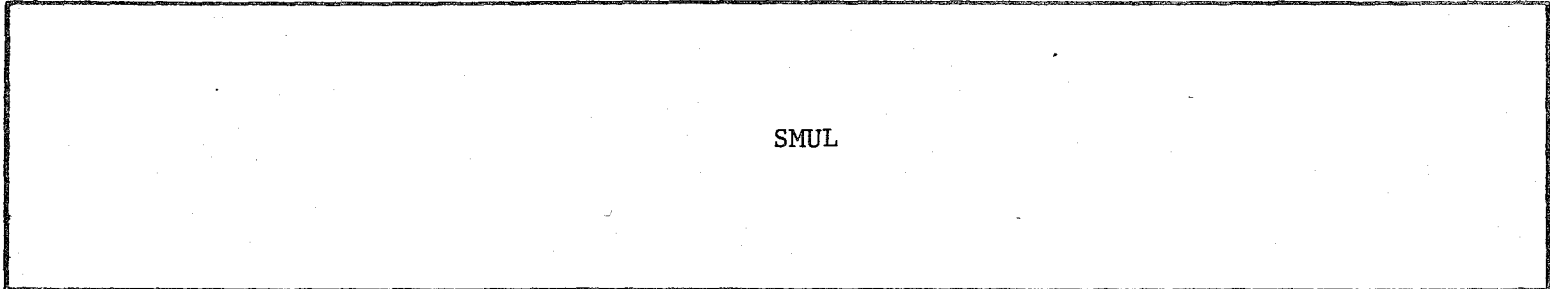


DESCRIPTION: POINTER TO INITIALIZATION AND RESTART ROUTINE. USED BY POWER FAIL ROUTINE TO INITIALIZE LOCATION 0 FOR A RESTART.

NAME: ZSMUL

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15.....



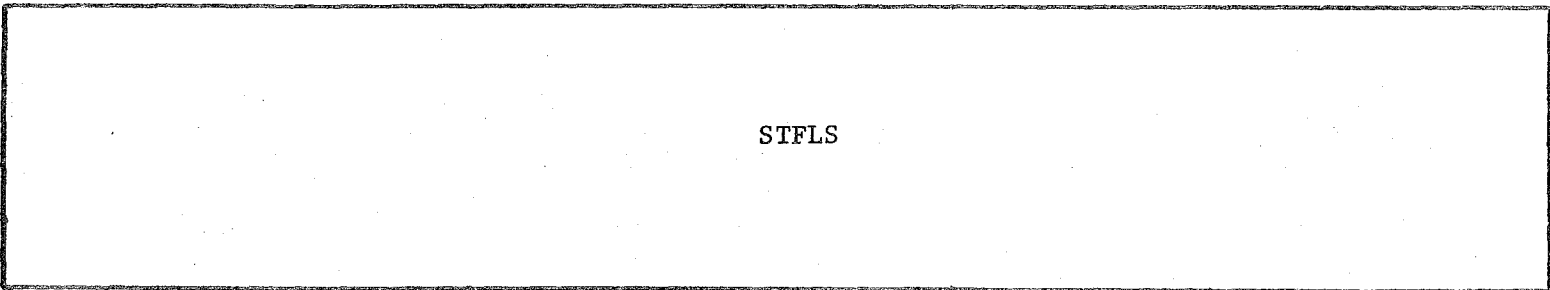
DESCRIPTION: POINTER TO SUBROUTINE SMUL. PERFORM A SIGNED MULTIPLY ON QUANTITIES IN AC1 AND AC2.
PLACE DOUBLE LENGTH PRODUCT IN AC0, AC1.

B-41

NAME: ZSTFL

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15.....

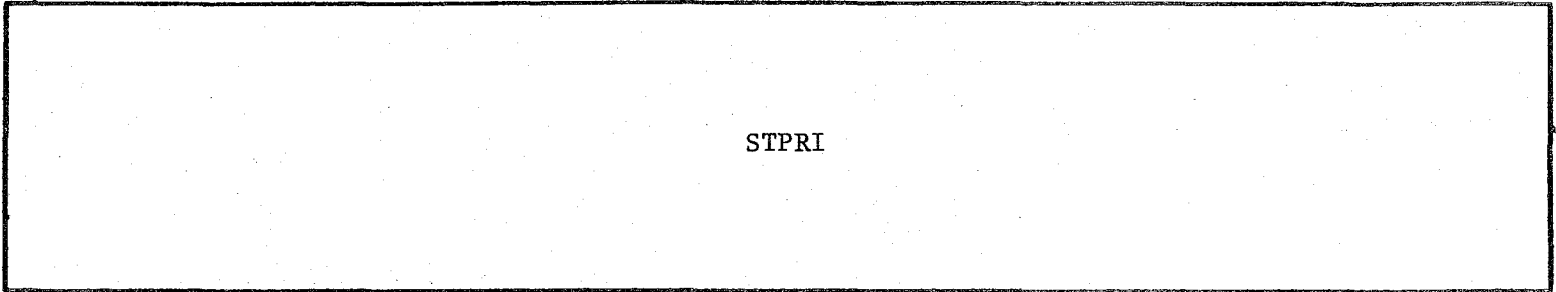


DESCRIPTION: POINTER TO SUBROUTINE STFLS. SETUP FLASH MODE. DESIRED FLASH TIME IN .5 SEC.
UNITS IN AC0. SIGNAL OUTPUTS WILL BE INITIALIZED TO FLASH CONFIGURATION.

. NAME: ZSTPR

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...



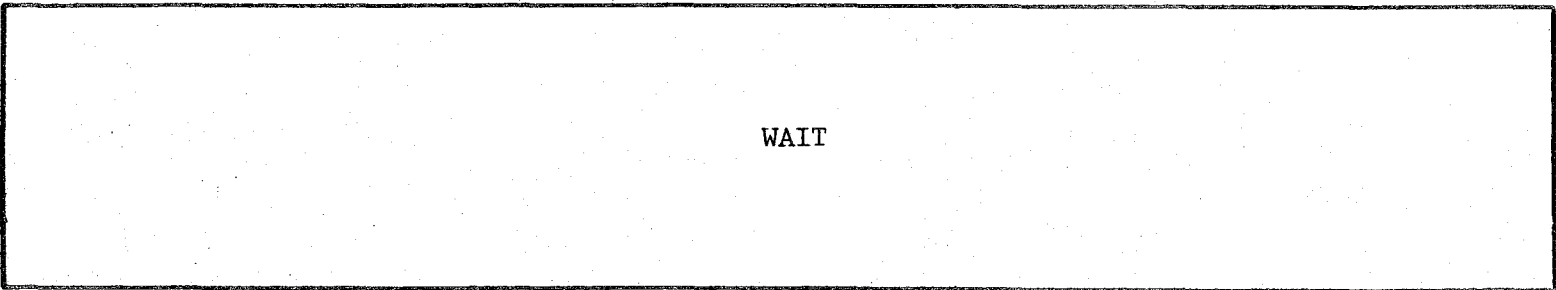
DESCRIPTION: POINTER TO SUBROUTINE STPRI. FORM LOGICAL "OR" OF ACO AND TELEMETRY OUTPUT PRIORITY WORD (OUTPUT FORMAT 15), AND STORE THE RESULT FOR IMMEDIATE OUTPUT.

B-42

. NAME: ZWAIT

SIZE: 1

Bit Number.....0.....1.....2.....3.....4.....5.....6.....7.....8.....9.....10.....11.....12.....13.....14.....15...



DESCRIPTION: POINTER TO SUBROUTINE WAIT. SAVE ADDRESS OF CALL+1 IN PCSAV AREA FOR THE EXECUTING TASK. DECREMENT THE STATE OF THE EXECUTING TASK AND ENTER THE TASK SCHEDULER. CALLED BY EACH TASK UPON COMPLETION OF EACH FRAME.

APPENDIX C

INPUT/OUTPUT BIT ASSIGNMENTS

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
INPUT Z MAGNETOMETERS	"A" SB RIGHT LANE	"A" SB CENTER LANE	"A" SB LEFT LANE	"A" EB RIGHT LANE	"A" EB CENTER LANE	"A" EB LEFT LANE	"B" NB RIGHT LANE	"B" NB CENTER LANE	"B" NB LEFT LANE	"B" WB RIGHT LANE	"B" WB CENTER LANE	"B" WB LEFT LANE	HASK'L "A" EB RIGHT LANE	HASK'L "A" EB LEFT LANE			
INPUT Z+1 LOOPS & MISC. DETECTORS	"A" SB LOOP	M'BIRD U-TURN MAG.		"A" EB LOOP	M'BIRD "A" EB RT MAG.	"B" EB LT LOOP M'BIRD MAG.	"B" NB LOOP			"B" WB LOOP	M'BIRD "B" WB RT	"A" WB LT LOOP M'BIRD MAG.	"A" EB LT LOOP				
INPUT Z+2 PED BUTTONS		"A" WEST SIDE		"A" SOUTH SIDE	"A" NORTH SIDE			"B" EAST SIDE		"B" NORTH SIDE	"B" SOUTH SIDE						
INPUT Z+3 SWITCHES	STALL ALARM	CAB. TEMP. SWITCH	SIGNAL SHUT- DOWN SWITCH	SIGNAL FLASH SWITCH	MANUAL SWITCH	AUXIL- IARY FLASH STATUS	CAB. DOOR SWITCH	FLASH INTER- CON- NECT RELAY	DIAL 2 INTER- CON- NECT RELAY	DIAL 3 INTER- CON- NECT RELAY	RESET INTER- CON- NECT RELAY						
INPUT Z+4 AC SENSORS	"A" EB LT RED	"A" WEST SIDE DON'T WALK	"A" SOUTH SIDE DON'T WALK	"A" NORTH SIDE DON'T WALK		"B" EAST SIDE DON'T WALK	"B" NORTH SIDE DON'T WALK	"B" SOUTH SIDE DON'T WALK	"A" SB RED	"A" EB RED	"A" WB RED	"A" WB LT RED	"B" NB RED	"B" WB RED	"B" EB RED	"B" EB LT RED	
INPUT Z+5 AC SENSORS	"A" EB LT AMBER							INTER- SEC- TION FLASH	"A" SB AMBER	"A" EB AMBER	"A" WB AMBER	"A" WB LT AMBER	"B" NB AMBER	"B" WB AMBER	"B" EB AMBER	"B" EB LT AMBER	
INPUT Z+6 AC SENSORS	"A" EB LT GREEN	"A" WEST SIDE WALK	"A" SOUTH SIDE WALK	"A" NORTH SIDE WALK		"B" EAST SIDE WALK	"B" NORTH SIDE WALK	"B" SOUTH SIDE WALK	"A" SB GREEN	"A" EB GREEN	"A" WB GREEN	"A" WB LT GREEN	"B" NB GREEN	"B" WB GREEN	"B" EB GREEN	"B" EB LT GREEN	

SB = SOUTHBOUND
EB = EASTBOUND

WB = WESTBOUND
NB = NORTHBOUND

"A" = WEST SIDE OF DIAMOND
"B" = EAST SIDE OF DIAMOND

MAG. = MAGNETOMETER
LT = LEFT TURN

RT = RIGHT TURN

ACTUATOR DIGITAL INPUTS

BIT #

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

INPUT Z+7 (TELEMETRY)	TLM RECVR MARK/SPACE STATUS	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
INPUT Z+8 (PIN MATRIX)	← PHASE 1 MAXIMUM TIME -1 SECOND →	← PHASE 1 MINIMUM TIME -1 SECOND →						← PHASE 1 EXTENSION TIME -1 SECOND →			X		1= READ PIN MATRIX			
INPUT Z+9 (PIN MATRIX)	← PHASE 3 MAXIMUM TIME -1 SECOND →	← PHASE 3 MINIMUM TIME -1 SECOND →						← PHASE 3 EXTENSION TIME -1 SECOND →			← PHASE 2 TIME -1 SECOND →					
INPUT Z+10 (PIN MATRIX)	← PHASE 4 MAXIMUM TIME -1 SECOND →	← PHASE 4 MINIMUM TIME -1 SECOND →						← PHASE 4 EXTENSION TIME -1 SECOND →			X					
INPUT Z+11 (PIN MATRIX)	← PHASE 6 MAXIMUM TIME -1 SECOND →	← PHASE 6 MINIMUM TIME -1 SECOND →						← PHASE 6 EXTENSION TIME -1 SECOND →			← PHASE 5 TIME -1 SECOND →					
CONSOLE DATA SWITCHES INPUT (AMBER TIMES)	← "A" SB AMBER TIME →	← "A" EB AMBER TIME →	← "A" WB AMBER TIME →	← "A" WB LEFT TURN AMBER TIME →	← "B" NB AMBER TIME →	← "B" WB AMBER TIME →	← "B" EB AMBER TIME →	← "B" EB LEFT TURN AMBER TIME →								
NOTE: ALL AMBER TIMES ARE IN .5 SECOND UNITS. THEY WILL HAVE A BASE VALUE OF 3.0 SECONDS ADDED TO THEM.																

SB = SOUTH-BOUND WB = WEST-BOUND "A" = WEST SIDE OF DIAMOND
 EB = EAST-BOUND NB = NORTH-BOUND "B" = EAST SIDE OF DIAMOND

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OUTPUT V (SIGNAL LIGHTS)	"A" SB RED	"A" EB RED	"A" WB RED	"A" WB LEFT TURN RED	"B" NB RED	"B" WB RED	"B" EB RED	"B" EB LEFT TURN RED	"A" EB LEFT TURN RED	"A" WEST SIDE DON'T WALK	"A" SOUTH SIDE DON'T WALK	"A" NORTH SIDE DON'T WALK	X	"B" EAST SIDE DON'T WALK	"B" NORTH SIDE DON'T WALK	"B" SOUTH SIDE DON'T WALK
OUTPUT V+1 (SIGNAL LIGHTS)	"A" SB AMBER	"A" EB AMBER	"A" WB AMBER	"A" WB LEFT TURN AMBER	"B" NB AMBER	"B" WB AMBER	"B" EB AMBER	"B" EB LEFT TURN AMBER	"A" EB LEFT TURN AMBER	X						INTER- SEC- TION FLASH
OUTPUT V+2 (SIGNAL LIGHTS)	"A" SB GREEN	"A" EB GREEN	"A" WB GREEN	"A" WB LEFT TURN GREEN	"B" NB GREEN	"B" WB GREEN	"B" EB GREEN	"B" EB LEFT TURN GREEN	"A" EB LEFT TURN GREEN	"A" WEST SIDE WALK	"A" SOUTH SIDE WALK	"A" NORTH SIDE WALK	X	"B" EAST SIDE WALK	"B" NORTH SIDE WALK	"B" SOUTH SIDE WALK
OUTPUT V+3 (TELEMETRY)	TLM XMTR MARK/ SPACE STATUS	X														

SB = SOUTH-BOUND WB = WEST-BOUND "A" = WEST SIDE OF DIAMOND
 EB = EAST-BOUND NB = NORTHBOUND "B" = EAST SIDE OF DIAMOND

APPENDIX D

TELEMETRY FORMAT BIT ASSIGNMENTS

BIT #	0	1	2	3	4	5	6	7	8	9	10	11
	(START)											(STOP)
INPUT 1	0	ODD PARITY	0	1	COMMAND CODE INTERSECTION*		"A"	COMMAND CODE INTERSECTION*		"B"	1	1
INPUT 0	0	ODD PARITY	0	0	1=ONLINE 0=EMERG. CONTROL	1=ACTUATOR FLASH 0=NORMAL	X	1=RESYNC 0=NORMAL	1=ABORT/ STALL 0=NORMAL	X	1	1
OUTPUT 0	0	0	0	0	0	0	1=EMERG. CONTROL 0=NORMAL	1=RESTART 0=NORMAL	1=ACTUATOR FLASH 0=NORMAL	1=PERFORM RESYNC. 0=NORMAL	1	1
OUTPUT 1	0	0	0	0	0	1	1=SIGNAL CONFIRM ERROR 0=NORMAL	INTERSECTION "A" GREEN STATUS			1	1
OUTPUT 2	0	0	0	0	1	0	1=CAB. DOOR AJAR 0=NORMAL	INTERSECTION "B" GREEN STATUS			1	1
OUTPUT 3	0	0	0	0	1	1		"A" INTERSECTION EAST- BOUND ARTERIAL AVERAGE SPEED FROM UPSTREAM DETECTORS			1	1
OUTPUT 4	0	0	0	1	0	0		"A" INTERSECTION SOUTHBOUND FRONTAGE ROAD AVERAGE SPEED FROM UPSTREAM DETECTORS			1	1
OUTPUT 5	0	0	0	1	0	1		"B" INTERSECTION WESTBOUND ARTERIAL AVERAGE SPEED FROM UPSTREAM DETECTORS			1	1
OUTPUT 6	0	0	0	1	1	0		"B" INTERSECTION NORTHBOUND FRONTAGE ROAD AVERAGE SPEED FROM UPSTREAM DETECTORS			1	1

* See Page D-4

"A" = WEST SIDE OF DIAMOND

"B" = EAST SIDE OF DIAMOND

ACTUATOR TELEMETRY FORMAT

D-1

BIT #	0 (START)	1	2	3	4	5	6	7	8	9	10	11 (STOP)
OUTPUT 7	0	0	0	1	1	1	OCCUP "A" EB LOOP	OCCUP "A" LTEB LOOP	OCCUP "A" SB LOOP	OCCUP "A" LTWB LOOP	1	1
OUTPUT 8	0	0	1	0	0	0	OCCUP "B" WB LOOP	X	OCCUP "B" NB LOOP	OCCUP "B" LTEB LOOP	1	1
OUTPUT 9	0	0	1	0	0	1	VOL 1	VOL 2	VOL 3	VOL 4	1	1
OUTPUT 10	0	0	1	0	1	0	VOL 5	VOL 6	VOL 7	VOL 8	1	1
OUTPUT 11	0	0	1	0	1	1	VOL 9	VOL 10	VOL 11	VOL 12	1	1
OUTPUT 12	0	0	1	1	0	0	VOL 13	VOL 14	VOL 15	VOL 16	1	1
OUTPUT 13	0	0	1	1	0	1	VOL 17	1=COMMAND ACKNOWLEDGE 0=NORMAL	PED. CKT. #1 - "B" NORTH SIDE	PED. CKT. #2 "B" EAST SIDE	1	1
OUTPUT 14	0	0	1	1	1	0	PED. CKT. #3 "B" SOUTH SIDE	PED. CKT. #4 "A" SOUTH SIDE	PED. CKT. #5 "A" WEST SIDE	PED. CKT. #6 "A" NORTH SIDE	1	1
OUTPUT 15	0	0	1	1	1	1	1=ABORT/ STALL BIT	1=BUFFER OVERFLOW	1=COMMAND REJECT	1=PARITY ERROR	1	1

D-2

SB = SOUTH-BOUND WB = WEST-BOUND "A" = WEST SIDE OF DIAMOND
 EB = EAST-BOUND NB = NORTHBOUND "B" = EAST SIDE OF DIAMOND
 LT = LEFT TURN

ACTUATOR TELEMETRY FORMAT

#	(START)		2				7		9	10	11 (STOP)	
OUTPUT 16	0	1	0	0	0	0	TEMP. SWITCH	SIG. SHUT- DOWN SW.	MANUAL FLASH SWITCH	MANUAL SWITCH	1	1
OUTPUT 17	0	1	0	0	0	1	FLASH INTERCON.	DIAL 2 INTERCON.	DIAL 3 INTERCON.	RESET INTERCON.	1	1

ACTUATOR TELEMETRY FORMAT

D-3

ACTUATOR INTERSECTION
COMMAND CODES

<u>A-CODE (OCTAL)</u>	<u>DECODED VALUE (BINARY)</u>	<u>GREEN MOVEMENTS</u>	<u>B-CODE (OCTAL)</u>	<u>DECODED VALUE (BINARY)</u>	<u>GREEN MOVEMENTS</u>
0	0000	ALL RED	0	0000	ALL RED
1	1000	↓	1	1000	↑
2	0100	→	2	0100	←
3	0010	←	3	0010	→
4	0001	↙	4	0001	↘
5	0110	→ ←	5	0110	← →
6	0011	↙ ↘	6	0011	↘ ↙
7*	0000BBBB1*	↙ ↘	7*	0100BBBB1*	↘ ↙

* 7 CODE IS USED AT FOREST LANE ONLY

ACODE = 7, BCODE = B

BCODE = 7, ACODE = B

APPENDIX E

ACTUATOR PROGRAM SEGMENTS

ACTUATOR PROGRAM SEGMENTS

<u>SEGMENT #</u>	<u>TAPE #</u>	<u>DESCRIPTION</u>
0 (COMMON)	1	PAGE 0 < 200
	2	PAGE 0 < 400
1 (SUBROUTINES)	1	COMMON SUBROUTINES
	2	COMMON SUBROUTINES
2 (INTERRUPT)	1	INITIALIZATION, RESTART, INTERRUPT PROCESSING
	2	TELEMETRY INPUT
	3	TELEMETRY OUTPUT, TASK MONITOR
3 (TASK 0)	1	SPEED DETECTOR PROCESSING (INPUT Z)
4 (TASK 1)	1	INITIALIZATION, INPUT Z+1, Z+2 PROCESSING
	2	SPEED CALCULATION, TELEMETRY DETECTOR OUTPUT
	3	TASK 1 SUBROUTINES
	4	TASK 1 PARAMETER DEFINITIONS
5 (TASK 2)	1	INITIALIZATION, LOOP DETECTOR PROCESSING, SIGNAL CONFIRM TESTING
	2	TIME GREENS AND AMBERS, DECODE COMMANDS
	3	PROCESS SIGNAL COMMANDS
	4	TASK 2 TELEMETRY OUTPUTS, SIGNAL OUTPUTS
6 (TASK 3)	1	INITIALIZATION, SUBROUTINES, PIN MATRIX PROCESSING
	2	FLASH PROCESSING, EMERGENCY CONTROL
7 (IDLE)	1	TELETYPE INTERACTION BACKGROUND JOB
8 (SYSTEM DEFINITIONS)	1	DEFINES "COMMON" PARAMETERS. MUST PREFIX ALL OTHER SEGMENTS EXCEPT #0.

APPENDIX F

ACTUATOR PROGRAM LISTINGS

PROGRAM LISTING

TABLE OF CONTENTS

	<u>PAGE NO.</u>
1. PROGRAM SEGMENT 0 (COMMON)	F-1
2. PROGRAM SEGMENT 1 (SUBROUTINES)	F-11
3. PROGRAM SEGMENT 2 (INTERRUPT)	F-20
4. PROGRAM SEGMENT 3 (TASK0)	F-36
5. PROGRAM SEGMENT 4 (TASK1)	F-41
6. PROGRAM SEGMENT 5 (TASK2)	F-55
7. PROGRAM SEGMENT 6 (TASK3)	F-75
8. PROGRAM SEGMENT 7 (IDLE)	F-87
9. PROGRAM SEGMENT 8 (SYSTEM DEFINITIONS)	F-94
10. LOADER MAP	F-96

PROGRAM SEGMENT 0

.TITL SEGM0

; THIS SECTION CONTAINS ALL "COMMON" DEFINITIONS FOR
 ; THE ACTUATOR PROGRAM

; ENTRY POINTS

.ENT PCINT,SPCEL,AINC1,AINC2,AINC3,AINC4
 .ENT AINC5,STMR,PTMR,CYCLE,STATE,PCSAV,A0SAV
 .ENT A1SAV,A2SAV,A3SAV,TIMER,TETIM,SPTAB
 .ENT LAKC0,LAKC1,LRJC0,LRJC1
 .ENT BIT,C32,C16,C8,C4,C2,C1
 .ENT ESPCE,ECYCL,ESTAT,EPCSA,EA0SA,EA1SA
 .ENT EA2SA,EA3SA,ETIME,ETETI
 .ENT RR0TB,PE0TB,IFBIT,IABTB,IRSYB,CR0TB
 .ENT ORSYB,I0NLB,IAFLB,OCABR,SCERB,DAFLB
 .ENT DEMRB,0ASTB,0B0FB,0CDAB,CDART,SSDNB
 .ENT TIBUF,TIFIR,TIIN,TIOUT,TOBUF,TOFIR,TOIN
 .ENT T0OUT,MHTSK,XEQPR,SCEDF,DATAM,0WRDM
 .ENT LOWM,FORMM,LBYTE,STBTS
 .ENT ZPUT,ZGET,ZOR,ZSTPR,ZSTFL,ZWAIT,ZRSTR
 .ENT ZIJMP,ZALS,ZBITN,ZABRT,ZSMUL,ZCLER,ZPTSB
 .ENT IMASK,UMASK,ASTLF,FLTIM,RSFLG,TMOUT
 .ENT TMINT,BITN,GSTA,ASTA,RSTA
 .ENT VOLUM,PRZP1,PRZP2,LOOPS,B0FLG,SGSTA
 .ENT TMERF,EMMOD,ECOMC,CNFLS,C7,MAMTM
 .ENT APDMT,TOUTW,RSWRD,PRIWR,STWRD,ASTWR
 .ENT TOW1,TOW2,TOW3,TOW7,TOW9,TOW13,TOW16
 .ENT TOW17,PDMT1,PDMT2,PDMT5,PDMT6,PDMT4
 .ENT RRFLG,PRRER,PRAER,PRGER,CLOKF
 .ENT RLMSK,ALMSK,GLMSK,AXFLB,INTCD
 .ENT PDMT,AMTM

; EXTERNALS

.EXTN INTRP,PUT,GET,OR,STPRI,STFLS,WAIT,INTR
 .EXTN IJUMP,ALS,BITN,ABORT,SMUL,CLEAR,PTSUB

; EQUIVALENCE DEFINED CONFIGURATION PARAMETERS

; TASK CONTROL

000004 HITSK = 4 ; NUMBER OF TASKS
 000003 IBFSZ = 3 ; INPUT BUFFER SIZE
 000021 0BFSZ = 17. ; OUTPUT BUFFER SIZE

; DEFINE PAGE ZERO LOCATIONS

; DEFINE VALUES BELOW LOC 200

000000 .LOC 0 ;
 000000 000000 PCINT: 0 ; INTERRUPTED PC
 000001 177777 INTRP ; INTERRUPT ROUTINE ADDRESS
 000002 000003 CLOKF: 3 ; RTC FREQ CODE
 000003 002230 JMP @ZRSTR ; RESTART

; TASK CYCLE TIMES

000004 000012 CYCLE: 10. ; TASK 0
 000005 000062 50. ; TASK 1
 000006 000144 100. ; TASK 2
 000007 000764 500. ; TASK 3
 00010 000000 PRRER: 0 ; RED ERROR BITS
 00011 000000 PRAER: 0 ; AMBER ERROR BITS
 00012 000000 PRGER: 0 ; GREEN ERROR BITS
 ; LAST TM SIGNAL INPUT CODES. RETAIN ORDER FOR INDEXING
 00013 000000 LAKC0: 0 ; LAST CODE 0 ACK
 00014 000000 LAKC1: 0 ; LAST CODE 1 ACK
 00015 000000 LRJC0: 0 ; LAST CODE 0 REJ

```

00016 000000 LRJC1:      0      ; LAST CODE 1 REJ
      000020          .LOC    20    ; DEFINE AUTO INCR LOCS
00020 000000 AINC1:      0      ; USED BY TASK 1
00021 000000 AINC2:      0      ; USED BY TASK 1
00022 000000 AINC3:      0      ; USED BY TASK 2
00023 000000 AINC4:      0      ; USED BY TASK 2
00024 000000 AINC5:      0      ; USED BY TASK 3
      000011 STMR:      .BLK    9.    ; SIGNAL TIMERS
      000011 PTMR:      .BLK    9.    ; PED TIMERS
;
; STORAGE WORDS
;
; TASK CONTROL
;
000004 STATE:      .BLK    HITSK   ; TASK STATE.
000004 PCSAV:     .BLK    HITSK+DEBUG ; PC AND CARRY
000004 A0SAV:     .BLK    HITSK+DEBUG ; AC0
000004 A1SAV:     .BLK    HITSK+DEBUG ; AC1
000004 A2SAV:     .BLK    HITSK+DEBUG ; AC2
000004 A3SAV:     .BLK    HITSK+DEBUG ; AC3
000004 TIMER:     .BLK    HITSK   ; TASK TIMER.
000000 TETIM:
      .IFN      DEBUG      ; *****
      .BLK      HITSK     ; TASK COMPLETION TIMES
      .ENDC
; SPEED CELLS
000016 NSPBT      =      14.    ; # OF SPEED BITS
000016 SPCEL:     .BLK    NSPBT   ; SPEED CELLS
000016 SPTAB:     .BLK    NSPBT   ; SPEED TABLE
; BIT DEFINITIONS
;
00137 100000 BIT:      1B0    ;
00140 040000      1B1    ;
00141 020000      1B2    ;
00142 010000      1B3    ;
00143 014000      1B4    ;
00144 002000      1B5    ;
00145 001000      1B6    ;
00146 000400      1B7    ;
00147 000200      1B8    ;
00150 000100      1B9    ;
00151 000040 C32:      1B10   ;
00152 000020 C16:      1B11   ;
00153 000010 C8:       1B12   ;
00154 000004 C4:       1B13   ;
00155 000002 C2:       1B14   ;
00156 000001 C1:       1B15   ;
      000146 RROTB      =      BIT+7 ; RESTART OUTPUT BIT
      000150 PEOTB      =      BIT+9. ; PARITY ERROR OUT
      000150 IFBIT      =      BIT+9. ; INPUT FORMAT
      000155 IABTB      =      BIT+14. ; INPUT ABORT
      000154 IRSYB      =      BIT+13. ; INPUT RESYNC
      000147 CROTB      =      BIT+8. ; COM REJ OUT
      000150 ORSYB      =      BIT+9. ; RESYNC OUT
      000151 IONLB      =      BIT+10. ; ONLINE INPUT
      000152 IAFLB      =      BIT+11. ; FLASH INPUT
      000154 OCAKB      =      BIT+13. ; COM ACK OUT +6
      000153 SCERB      =      BIT+12. ; CONFIRM ERR OUT + 6
      000155 OAFLEB      =      BIT+14. ; FLASH OUTPUT + 6
      000153 OEMRB      =      BIT+12. ; EMER CTL OUTPUT + 6
      000145 OASTB      =      BIT+6   ; ABT/STL OUT

```

```

000140 0B0FB = BIT+7 ; BUFF OFLD OUT
000153 0CDAB = BIT+12. ; CAB DOOR OUT + 6
000145 CDABT = BIT+6 ; CAB DOOR INPUT
000141 SSDNB = BIT+2 ; SIG SHUTDOWN SW INPUT
000144 AXFLB = BIT+5 ; AUX FL INPUT BIT
; DEFINE END OF TASK CONTROL ARRAYS FOR INDEXING
000121 ESPCE = SPCEL+NSPBT
000010 ECYCL = CYCLE+HITSK
000053 ESTAT = STATE+HITSK
000057 EPCSA = PCSAV+HITSK
000063 EA0SA = A0SAV+HITSK
000067 EA1SA = A1SAV+HITSK
000073 EA2SA = A2SAV+HITSK
000077 EA3SA = A3SAV+HITSK
000103 ETIME = TIMER+HITSK
000107 ETETI = TETIM+HITSK
;

```



```

; PARAMETERS BELOW LOC 400
;
; TELEMETRY I/O
000003 TIBUF: .BLK IBFSZ ; INPUT BUFFER
00162 000157 TIFIR: TIBUF ; FIRST (CONSTANT)-MUST FOLLOW I
00163 000157 TIIN: TIBUF ; IN POINTER
00164 000157 TIOUT: TIBUF ; OUT POINTER
000021 TOBUF: .BLK OBFSZ ; OUTPUT BUFFER
00206 000165 TOFIR: TOBUF ; FIRST (CONSTANT)-MUST FOLLOW O
00207 000165 TOIN: TOBUF ; IN
00210 000165 TOOUT: TOBUF ; OUT
; TASK CONTROL VALUES
00211 177774 MHTSK: -HITSK ; -HIGHEST TASK NO.
00212 000000 XEQPR: 0 ; EXECUTING TASK PRIORITY (OR I
00213 000000 SCEDF: 0 ; SCHEDULER MODE FLAG
;
; OTHER TELEMETRY DATA
;
00214 001700 DATAM: 17B9 ; OUTPUT DATA BITS MASK
00215 077700 DWRDM: 777B9 ; OUTPUT CODE+DATA BITS MASK
00216 000017 LOW4M: 17 ; LOWER 4 BIT MASK
00217 076000 FORMM: 37B5 ; OUTPUT FORMAT MASK
00220 000377 LBYTE: 377 ; LOWER BYTE MASK
00221 000070 STBTS: 7B12 ; START-STOP BITS FOR OUTPUTS.
; SUBROUTINE ADDRESSES
00222 177777 ZPUT: PUT ; PUT WORD INTO CIRCULAR BUFFER.
00223 177777 ZGET: GET ; GET WORD FROM CIRCULAR BUFFER.
00224 177777 ZOR: OR ; LOGICAL INCLUSIVE OR
00225 177777 ZSTPR: STPRI ; STORE PRIORITY OUTPUT WORD.
00226 177777 ZSTFL: STFLS ; SETUP FLASH MODE.
00227 177777 ZWAIT: WAIT ; TASK COMPLETION DELAY.
00230 177777 ZRSTR: INTR ; RESTART
00231 177777 ZIJMP: IJUMP ; INDEX 2 TABLE JUMP SUBR.
00232 177777 ZALS: ALS ; CIRCULAR LEFT SHIFT SUBR
00233 177777 ZBITN: BITNM ; BIT #
00234 177777 ZABRT: ABORT ; ABORT
00235 177777 ZSMUL: SMUL ; SIGNED MULTIPLY
00236 177777 ZCLER: CLEAR ; CLEAR LIST OF VARIABLES
00237 177777 ZPTSR: PTSUB ; PUT WORD IN TM OUTPUT BUFR
; THE FOLLOWING FOUR PARAMETERS ARE INTERSECTION
; DEPENDENT. BEFORE PUNCHING THE ACTUATOR LOAD
; TAPES THEY MUST BE ALTERED TO REFLECT THE
; SIGNAL CONFIGURATION AT EACH INTERSECTION.
00240 167000 RLMSK: 167000 ; RED LIGHT MASK
00241 167000 ALMSK: 167000 ; AMB LIGHT MASK
00242 177400 GLMSK: 177400 ; GRN LIGHT MASK
00243 000000 INTCD: 0 ; INTERSECTION CODE.0=NORMAL,2=FOR,
; ; 4=MOC,6=MCC.
; INTERRUPT VALUES
00244 177777 IMASK: 177777 ; MASK ALL DEVICES
00245 177767 UMASK: 177767 ; UNMASK REAL TIME CLOCK
;
; MISCELLANEOUS FLAGS, TEMP STORAGE, ETC.
00246 000000 ASTLF: 0 ; ABORT STALL FLAG
00247 000000 FLTIM: 0 ; FLASH TIMR
00250 000000 RSFLG: 0 ; RESYNC FLAG
00251 000000 TMOUT: 0 ; TELEMETRY OUTPUT TIMER
00252 000000 TMINT: 0 ; TELEMETRY INPUT TIMER
00253 000000 BITN: 0 ; OUTPUT BIT COUNTER - TM

```

```

00254 000000 GSTA:      0      ; GREEN OUTPUT STATUS
00255 000000 ASTA:      0      ; AMBER OUTPUT STATUS
00256 000000 RSTA:      0      ; RED OUTPUT STATUS
00257 000000 VOLUM:    0      ; Z+0 VOLUME BITS
00260 000000 PREP1:    0      ; PREVIOUS Z+1 INPUT
00261 000000 PREP2:    0      ; PREVIOUS Z+2 INPUT
00262 000000 LOPPS:    0      ; LOOP DET INPUT FOR EMER CTL
00263 000000 B7FLG:    0      ; BUFFER OFLO FLAG
00264 000000 SGSTA:    0      ; SIGNAL STATUS WORD (INPUT Z+3)
00265 000000 TMERF:    0      ; TM INPUT ERR FLAG
00266 000000 EMMOD:    0      ; EMER CTL MODE
00267 000000 ECOMC:    0      ; EMER CTL COMMAND
00270 000000 CNFLS:    0      ; CONFIRM FLASH FLAG
00271 000007 C7:        7      ; 7
00272 000010 MAMTM:    AMTM-1 ; ADDR-1 OF AMBER TIMES
00273 000000 APDMT:    PDMNT  ; ADDR OF PED TIMES

```

```

;
; TELEMETRY OUTPUT WORDS

```

```

000000 TOUTW:    ;
00274 000000 RSWRD:    0B5      ; OUTPUT FORMAT 0
00275 002000      1B5      ; 1
00276 004000      2B5      ; 2
00277 006000      3B5      ; 3
00300 010000      4B5      ; 4
00301 012000      5B5      ; 5
00302 014000      6B5      ; 6
00303 016000      7B5      ; 7
00304 020000     10B5      ; 8
00305 022000     11B5      ; 9
00306 024000     12B5      ; 10
00307 026000     13B5      ; 11
00310 030000     14B5      ; 12
00311 032000     15B5      ; 13
00312 034000     16B5      ; 14
00313 036000     17B5      ; 15
000274 STWRD  =  TOUTW  ; STATUS OUTPUTS.
00314 040000     20B5      ; 16
00315 042000     21B5      ; 17
00316 000273 ASTWR:    STWRD-1 ; ADDR-1 OF STATUS WORDS
00317 000000 RRFLG:    0      ; RESTART FLAG

```

```

.NREL

```

```

; THE FOLLOWING MUST BE IN COMMON BUT NOT
; NECESSARILY BELOW LOC 400
; PED MIN TIMES (.1 SEC)

```

```

00000'000120 PDMNT:    80.      ; BIT 0
00001'000120      80.      ; 1
00002'000120      80.      ; 2
00003'000062      50.      ; 3
00004'000120      80.      ; 4
00005'000120      80.      ; 5
00006'000120      80.      ; 6
00007'000062      50.      ; 7
00010'000062      50.      ; 8

```

```

; AMBER TIMES (.1 SEC)

```

```

00011'000050 AMTM:      40.      ; BIT 0
00012'000050      40.      ; 1
00013'000050      40.      ; 2
00014'000050      40.      ; 3
00015'000050      40.      ; 4

```

```

---
00016'000050      40.      ; 5
00017'000050      40.      ; 6
00020'000050      40.      ; 7
00021'000043      35.      ; 8
;
000275 TOW1      =      TOUTW+1
000276 TOW2      =      TOUTW+2
000277 TOW3      =      TOUTW+3
000303 TOW7      =      TOUTW+7
000305 TOW9      =      TOUTW+9.
000311 TOW13     =      TOUTW+13.
000314 TOW16     =      TOUTW+16.
000315 TOW17     =      TOUTW+17.
;
000001' PDMT1    =      PDMNT+1
000002' PDMT2    =      PDMNT+2
000004' PDMT4    =      PDMNT+4
000005' PDMT5    =      PDMNT+5
000006' PDMT6    =      PDMNT+6
.END      ; END COMMON

```

A0SAV	000057
A1SAV	000063
A2SAV	000067
A3SAV	000073
ABORT	000234 X
AINC1	000020
AINC2	000021
AINC3	000022
AINC4	000023
AINC5	000024
ALMSK	000241
ALS	000232 X
AMTM	000011
APDMT	000273
ASTA	000255
ASTLF	000246
ASTWR	000316
AXFLB	000144
BIT	000137
BITN	000253
BITNM	000233 X
BOFLG	000263
C1	000156
C16	000152
C2	000155
C32	000151
C4	000154
C7	000271
C8	000153
CDABT	000145
CLEAR	000236 X
CLOCKF	000002
CNFLS	000270
CRDTB	000147
CYCLE	000004
DATAM	01014
EA0SA	000063
EA1SA	000067
EA2SA	000073
EA3SA	000077
ECOMC	000267
ECYCL	000010
EMMOD	000266
EPCSA	000057
ESPCE	000121
ESTAT	000053
ETETI	000107
ETIME	000103
FLTIM	000247
FORMM	000217
GET	000223 X
GLMSK	000242
GSTA	000254
HITSK	000004
IABTB	000155
IAFLB	000152
IBFSZ	000003
IFBIT	000150
IJUMP	000231 X
IMASK	000244

```

---
INITR 000230 X
INTCD 000243
INTRP 000001 X
IONLB 000151
IRSYR 000154
LAKC0 000013
LAKC1 000014
LBYTE 000220
LOOPS 000262
LOW4M 000216
LRJC0 000015
LRJC1 000016
MAMTM 000272
MHTSK 000211
NSPBT 000016
OAFLB 000155
OASTB 000145
ORFSZ 000021
OROFB 000146
OCAKB 000154
OCDAB 000153
OEMRB 000153
OR      000224 X
ORSYB 000150
QWRDM 000215
PCINT 000000
PCSAV 000053
PDMVT 000000
PDMT1 000001
PDMT2 000002
PDMT4 000004
PDMT5 000005
PDMT6 000006
PEOTB 000150
PRAER 000011
PRGER 000012
PRIWR 000313
PRRER 000010
PREP1 000260
PREP2 000261
PTMR  000036
PTSUR 000237 X
PUT    000222 X
RLMSK 000240
RRFLG 000317
RROTB 000146
RSFLG 000250
RSTA  000256
RSWRD 000274
SCEDF 000213
SCERB 000153
SGSTA 000264
SMUL  000235 X
SPCEL 000103
SPTAB 000121
SSDNB 000141
STATE 000047
STBTS 000221
STFLS 000226 X
STMR  000025

```

```

---
STPRI 000225 X
STWRD 000274
TETIM 000103
TIBUF 000157
TIFIR 000162
TIIN 000163
TIMER 000077
TIOUT 000164
TMERF 000265
TMINT 000252
TMOUT 000251
TOBUF 000165
TOFIR 000206
TOIN 000207
TOOUT 000210
TOUTW 000274
TOW1 000275
TOW13 000311
TOW16 000314
TOW17 000315
TOW2 000276
TOW3 000277
TOW7 000303
TOW9 000305
UMASK 000245
VOLUM 000257
WAIT 000227 X
XEQPR 000212
ZABRT 000234
ZALS 000232
ZBITN 000233
ZCLER 000236
ZGET 000223
ZIJMP 000231
ZOK 000224
ZPTSB 000237
ZPUT 000222
ZRSTR 000230
ZSMUL 000235
ZSTFL 000226
ZSTPR 000225
ZWAIT 000227

```

PROGRAM SEGMENT 1

```
                .TITL   SEGM1
; COMMON SUBROUTINES
;
; ENTRY POINTS
  .ENT   OR,STPRI,PUT,GET,STFLS,IJUMP,ALS,BITNM
  .ENT   ABORT,SMUL,CLEAR,PTSUB
;
; 1. OR SUBROUTINE
;   LDA   0,WRD1
;   LDA   1,WRD2
;   JSR   @ZOR
; LOGICAL SUM OF AC0 AND AC1 WILL BE RETURNED IN AC1.
; AC0 WILL BE COMPLEMENTED. REENFRANT.
;
00000'100000 OR:   COM   0,0   ; NOT(AC0)
00001'107400   AND   0,1   ; RESET BITS THAT ARE 1'S IN BO
00002'106000   ADC   0,1   ; NOW ADDITION PERFORMS OR.
00003'001400   JMP   0,3   ; RETURN.
;
; 2. SET BIT IN PRIORITY TELEMETRY OUTPUT WORD.
;   LDA   0,BIT
;   JSR   @ZSTPR
; BIT WILL BE OR'D INTO PRIORITY WORD.
; NOT REENFRANT. AC0 WILL BE COMPLEMENTED. AC1=UPDATE
;
00004'024174S STPRI: LDA   1,PRIWR ; AC1=PRIORITY WORD.
00005'100000   COM   0,0   ; PERFORM OR AS ABOVE
00006'107400   AND   0,1   ;
00007'106000   ADC   0,1   ;
00010'044174S   STA   1,PRIWR ; UPDATE PRIORITY WORD.
00011'001400   JMP   0,3   ; RETURN
;
; 3. PLACE WORD IN CIRCULAR BUFFER.
;   JSR   @ZPUT   ; CALL PUT.
;         BPTRS   ; ADDRESS OF BUFFER POINTERS (
;         WORD    ; ACTUAL DATA TO BE STORED.
;   JMP   FULL    ; BUFFER WAS FULL RETURN.
;   ---      ---  ; NORMAL RETURN
; USES ALL REGISTERS. AC1=WORD STORED UNLESS BUFFER
; REENFRANT.
;
00012'031400 PUT:   LDA   2,0,3   ; AC2 = ADDR OF BUFFER POINTERS
00013'021001   LDA   0,1,2   ; AC0 = IN POINTER
00014'025002   LDA   1,2,2   ; AC1 = OUT POINTER
00015'101400   INC   0,0   ; IN+1
00016'142432   .SGT   2,0   ; TEST IN+1 = LIMIT
00017'021000   LDA   0,0,2   ; YES, IN+1 = FIRST
00020'106415   .SNE   0,1   ; TEST IN+1 = OUT
00021'001402   JMP   2,3   ; YES, BUFFER FULL.
00022'025401   LDA   1,1,3   ; AC1 = WORD TO STORE.
00023'047001   STA   1,0,2   ; STORE INTO ORIGINAL IN LOCAT
00024'041001   STA   0,1,2   ; UPDATE IN.
00025'001403   JMP   3,3   ; NORMAL RETURN
;
; 4. FETCH WORD FROM CIRCULAR BUFFER
;   JSR   @ZGET   ; CALL GET
;         BPTRS   ; ADDRESS OF BUFFER POINTERS (
;   JMP   EMPTY   ; BUFFER EMPTY RETURN.
;   ---      ---  ; NORMAL RETURN WITH WORD IN AC
; USES ALL REGISTERS.
```



```

; REENTRANT.
;
00026'031400 GET: LDA 2,0,3 ; AC2 = ADDR OF BUFFER POINTERSE
00027'021001 LDA 0,1,2 ; AC0 = IN POINTER
00030'025002 LDA 1,2,2 ; AC1 = OUT POINTER
00031'106415 .SNE 0,1 ; TEST IN=OUT
00032'001401 JMP 1,3 ; YES, BUFFER EMPTY.
00033'023002 LDA 0,02,2 ; GET WORD FROM OUT.
00034'125400 INC 1,1 ; OUT+1
00035'146432 .SGT 2,1 ; IS OUT+1 = LIMIT.
00036'025000 LDA 1,0,2 ; YES, OUT+1 = FIRST
00037'045002 STA 1,2,2 ; UPDATE OUT.
00040'001402 JMP 2,3 ; NORMAL RETURN.
;
; 5. SETUP OUTPUTS FOR FLASH MODE
; JSR @ZSTFL ; CALL STFLS
; FLASH TIME IN .5 SEC UNITS MUST BE IN
; AC0. IF 0, USES MINIMUM FLASH TIME.
; NOT REENTRANT
;
00041'101004 STFLS: .SZR 0,0 ; TEST TIME=0
00042'000405 JMP FLSNZ ; NO
00043'024153$ LDA 1,ASTA ; YES, TEST FLASH
00044'125212 MOVR# 1,1,SZC ; BIT ALREADY SET
00045'001400 JMP 0,3 ; YES, RETURN
00046'020412 LDA 0,FLMIN ; MIN TIME FOR FLASH
00047'024145$ FLSNZ: LDA 1,FLTIM ; CURRENT FLASH TIMER
00050'122432 .SGT 1,0 ; TEST NEW<CURRENT
00051'040145$ STA 0,FLTIM ; NO, USE NEW
00052'102460 .ZAC 0,0 ;
00053'040152$ STA 0,GSTA ; CLEAR GREEN AND
00054'040154$ STA 0,RSTA ; RED OUTPUTS.
00055'102520 .GP1 0,0 ;
00056'040153$ STA 0,ASTA ; SET FLASH BIT IN AMBER OUTPUT
00057'001400 JMP 0,3 ; RETURN
00060'000024 FLMIN: 20. ; MIN FLASH TIME (.5 SEC UNITS)
;
; 6. INDEXED JUMP USING AC2
; JSR @ZIJMP ; CALL IJUMP
; ADDR0 ; JUMP ADDRESS FOR AC2 = 0
; ADDR1 ; JUMP ADDRESS FOR AC2 = 1
; '
; '
; ADDR# ; JUMP ADDRESS FOR AC2 = #
; DESTROYS AC3
; REENTRANT
00061'157000 IJUMP: ADD 2,3 ; INDEX TO JUMP ADDRESS
00062'003400 JMP 00,3 ; PERFORM INDEXED JUMP
;
.EOT

```

```

;
; 7. SUBROUTINE TO PERFORM CIRCULAR LEFT SHIFT ON AC0. E
;     COUNT IN AC2. REENRANT.
;     JSR     @ZALS    ; CALL SUBR
;     ---     ----    ; NORMAL RETURN WITH SHIFTED VAL
;
00063'150400 ALS:    NEG      2,2    ; -SHIFT COUNT
00064'101122 ALSLP:  MOVZL   0,0,SZC ; B0 TO CRY, ZERO TO B15
00065'101400      INC      0,0    ; B15 = 1 IF CRY SET.
00066'151404      INC      2,2,SZR ; INCR SHIFT COUNT
00067'000775      JMP      ALSLP  ; LOOP SHIFT COUNT TIMES
00070'001400      JMP      0,3    ; EXIT WITH SHIFTED VALUE IN AC0
;
; 8. SUBROUTINE TO PLACE BIT # OF LEFTMOST BIT SET IN AC
;     INTO AC2, AND LEFT-SHIFT AC0 UNTIL THAT BIT IS
;     AC2 MUST = -1 ON INITIAL CALL. REENRANT
;     JSR     @ZBITN  ; CALL SUBR
;     ---     ----    ; RETURN HERE IF NO BITS SET IN
;     ---     ----    ; NORMAL RETURN
;
00071'101005 BITNM:  MOV      0,0,SNR ; TEST NO BITS SET
00072'001400      JMP      0,3    ; RETURN TO CALL+1
00073'151400      INC      2,2    ; INCR BIT #
00074'101123      MOVZL   0,0,SNC ; TEST ONE BIT
00075'000776      JMP      -,-2   ; NO, LOOP TILL ONE BIT FOUND
00076'001401      JMP      1,3    ; YES, RETURN TO CALL+2
;
; 9. SUBROUTINE TO SET ABORT/STALL FLAG AND HALT. NORMA
;     FOR INTERNALLY DETECTED ERRORS
;     JSR     @ZABRT
;
00077'054144$ ABORT:  STA      3,ASTLF ; SET ABORT STALL FLAG = CALL+1
00100'063077      HALT      ; STOP
;
; 10. SIGNED MULTIPLY SUBROUTINE. AC0,AC1 = SIGNED PRODUC
;     AC2 IS LEFT INTACT. ALL OTHER REGISTERS ARE CHA
;     JSR     @ZSMUL  ; CALL SUBR
;     ---     ----    ; RETURN
;     SEE NOVA MANUAL FOR DESCRIPTION
;
00101'102460 SMUL:   .ZAC      0,0    ; SIGNED MULTIPLY ENTRY. AC0=0
00102'125112      MOVL#    1,1,SZC ; N1 NEGATIVE
00103'143000      ADD       2,0    ; YES, CORRECT N2
00104'151112      MOVL#    2,2,SZC ; N2 NEGATIVE
00105'123000      ADD       1,0    ; YES, CORRECT N1
00106'040417      STA      0,CFAC  ; SAVE CORR FAC
00107'102460 UMUL:   .ZAC      0,0    ; ENTRY FOR UNSIGNED MULTIPLY
00110'054413 MPY:    STA      3,MLRET ; ENTRY FOR AC0,AC1=AC0+AC1*AC2
00111'034413      LDA      3,M16   ; GET STEP COUNT
00112'125203 MULOP:  MOVR     1,1,SNC ; CHECK NEXT MULTIPLIER BIT
00113'101201      MOVR     0,0,SKP ; 0,SHIFT
00114'143220      ADDZ    2,0    ; 1,ADD MULTIPLICAND AND SHIFT
00115'175404      INC      3,3,SZR ; COUNT STEP,COMPL CRY ON LAST 0
00116'000774      JMP      MULOP  ; ITERATE LOOP
00117'125260      MOVCR   1,1    ; SHIFT IN LAST LOW BIT(WHICH WE
00120'034405      LDA      3,CFAC  ; CORR FAC
00121'162400      SUB      3,0    ; CORRECT HIGH ORDER
00122'002401      JMP      @MLRET ; BY FINAL COUNT) AND RESTORE CX
00123'000000 MLRET:  0          ; RETURN ADDR
00124'177760 M16:   -16.      ; 16 STEPS

```

```

---
00125'000000 CFAC:      0      ; CORR FAC
;
; 11. SUBROUTINE TO CLEAR LOCATIONS GIVEN IN A LIST
;      JSR      @ZCLER ; CALL SUBR
;      LIST    ; ADDR OF LIST CONTAINING ADDRES
;      ; OF LOCS TO BE CLEARED, TERMINATE
;      REENTRANT. USES ALL REGS.
00126'031400 CLEAR: LDA      2,0,3 ; ADDR OF FIRST OF LIST
00127'102460 .ZAC      0,0      ; CLEAR AC0
00130'025000 CLRLP: LDA      1,0,2 ; GET ADDR FROM LIST
00131'125005 .SNZ      1,1      ; TEST END OF LIST
00132'001401 JMP       1,3      ; YES, RETURN
00133'043000 STA      0,00,2 ; CLEAR LOC
00134'151400 INC       2,2      ; NEXT LIST ADDR
00135'000773 JMP       CLRLP   ; LOOP
;
; 12. SUBR TO PLACE WORD IN TM OUTPUT BUFFER
;      AC0=0 PLACE WORD IN BUFFER UNLESS ZERO, CLEAR BF
;      AC0=-1 PLACE WORD IN BUFFER IF CHANGED FROM LAST
;      JSR      @ZPTSB
;      ARG0    ; ADDR OF OUTPUT WORD(LAST 4 BITS)
;      ARG1    ; OUTPUT FORMAT ADDR
;      USES ALL REGS. NON-REENTRANT
;
00136'040444 PTSUB: STA      0,ENFLG ; SAVE NTRY FLG
00137'023400 LDA      0,00,3 ; OUTPUT WORD TO AC0
00140'024120$ LDA      1,LOW4M ; LOW 4 BITS
00141'123700 ANDS     1,0      ; KEEP LOW 4 , PUT IN UPPER BYTE
00142'101220 MOVZR   0,0      ; POSITION FOR OUTPUT
00143'101220 MOVZR   0,0      ;
00144'027401 LDA      1,01,3 ; OUTPUT FORMAT WORD TO AC1
00145'030121$ LDA      2,FORMM ; FORMAT MASK
00146'133400 AND      1,2      ; FORMAT BITS
00147'113000 ADD      0,2      ; FORM FORMAT+ DATA BITS IN AC2
00150'050413 STA      2,POTWD ; STORE NEW OUTPUT WORD
00151'010431 ISZ     ENFLG   ; TEST ENTRY FLAG
00152'000404 JMP      PTUNC   ; NO COMPARE
00153'132415 .SNE    1,2      ; COMPARE CURRENT=LAST OUTPUT
00154'001402 JMP      2,3      ; SAME, RETURN
00155'000403 JMP      PTOUT   ; DIFFERENT, OUTPUT
00156'101005 PTUNC: .SNZ   0,0      ; TEST DATA BITS=0
00157'001402 JMP      2,3      ; YES, RETURN
00160'054423 PTOUT: STA      3,PTRET ; SAVE CALL+1
00161'006124$ JSR      @ZPUT   ; PUT WORD IN
00162'000110$ TOFIR   ; TM OUTPUT BUFR
00163'000000 POTWD: 0        ; OUTPUT WORD
00164'000413 JMP      BFRET   ; BUFR FULL
00165'034416 LDA      3,PTRET ; NORMAL RETN, GET CALL+1
;
; NOTE, STORED WORD IN AC1
00166'047401 STA      1,01,3 ; UPDATE LAST OUTPUT VALUE
00167'014413 DSZ     ENFLG   ; TEST ENTRY FLAG
00170'001402 JMP      2,3      ; COMPARE, EXIT
00171'023400 LDA      0,00,3 ; DATA WORD
00172'024120$ LDA      1,LOW4M ;
00173'124000 COM      1,1      ; .NOT. LOW 4 BITS
00174'123400 AND      1,0      ; CLEAR LOW 4 BITS AND
00175'043400 STA      0,00,3 ; RESTORE DATA WORD
00176'001402 JMP      2,3      ; RETURN
00177'034404 BFRET: LDA      3,PTRET ; IF BUFFER FULL,
00200'010161$ ISZ     BOFLG   ; INCR BUFR OFLO FLAG

```

00201'001402 JMP 2,3 ; RETURN
00202'000000 ENFLG: 0 ; ENTRY FLAG
00203'000000 PTRET: 0 ; RETURN ADDR
 .END ; END OF COMMON AREA

A0SAV 000015SX
A1SAV 000016SX
A2SAV 000017SX
A3SAV 000020SX
ABORT 000077'
AINC1 000003SX
AINC2 000004SX
AINC3 000005SX
AINC4 000006SX
AINC5 000007SX
ALS 000063'
ALSLP 000064'
AMB 000041
AMTM 177777 X
APDMT 000171SX
ASTA 000153SX
ASTLF 000144SX
ASTWR 000176SX
BFRET 000177'
BIT 000052SX
BITN 000151SX
BITNM 000071'
BOFLG 000161SX
C1 000060SX
C16 000054SX
C2 000057SX
C32 000053SX
C4 000056SX
C7 000167SX
C8 000055SX
CDABT 000101SX
CFAC 000125'
CLEAR 000126'
CLRLP 000130'
CNFLS 000166SX
CROTB 000066SX
CYCLE 000012SX
DATAM 000116SX
DEBUG 000000
EA0SA 000030SX
EA1SA 000031SX
EA2SA 000032SX
EA3SA 000033SX
ECOMC 000165SX
ECYCL 000025SX
EMMOD 000164SX
ENFLG 000202'
EPCSA 000027SX
ESPCE 000024SX
ESTAT 000026SX
ETETI 000035SX
ETIME 000034SX
FLMIN 000060'
FLSNE 000047'
FLTIM 000145SX
FORMM 000121SX
GET 000026'
GRN 000042
GSTA 000152SX
HITSK 000004

```

---
IABTB 000064$X
IAFLB 000071$X
IFBIT 000063$X
IJUMP 000061'
IMASK 000142$X
IONLB 000070$X
IRSYB 000065$X
LAKC0 000046$X
LAKC1 000047$X
LBYTE 000122$X
LOOPS 000160$X
LOW4M 000120$X
LRJC0 000050$X
LRJC1 000051$X
M16 000124'
MAMTM 000170$X
MHTSK 000113$X
MLRET 000123'
MPY 000110'
MULOP 000112'
NSPET 000016
OAFLB 000074$X
OASTB 000076$X
OBOFB 000077$X
OCAKB 000072$X
OCDAB 000100$X
OEMRE 000075$X
OR 000000'
ORSYB 000067$X
OWRDM 000117$X
PCINT 000001$X
PCSAV 000014$X
PDMNT 177777 X
PDMT1 177777 X
PDMT2 177777 X
PDMT4 177777 X
PDMT5 177777 X
PDMT6 177777 X
PEOTB 000062$X
POTWD 000163'
PRIWR 000174$X
PREP1 000156$X
PREP2 000157$X
PTMR 000011$X
PTOUT 000160'
PTRET 000203'
PTSUB 000136'
PTUNC 000156'
PUT 000012'
RED 000040
RFLG 000177$X
RROTB 000061$X
RSFLG 000146$X
RSTA 000154$X
RSWRD 000173$X
SCEDF 000115$X
SCERB 000073$X
SGSTA 000162$X
SMUL 000101'
SPEL 000002$X

```

```

---
SPTAB 000023SX
SSDNB 000102SX
STATE 000013SX
STBTS 000123SX
STFLS 000041'
STL 000057
STMR 000010SX
STPRI 000004'
STWRD 000175SX
TETIM 000022SX
TIBUF 000103SX
TIFIR 000104SX
TIIN 000105SX
TIMER 000021SX
TIN 000027
TIOUT 000106SX
TMERF 000163SX
TMINT 000150SX
TMOUT 000147SX
TOBUF 000107SX
TOFIR 000110SX
TOIN 000111SX
TOOUT 000112SX
TOUT 000043
TOUTW 000172SX
TOW1 000036SX
TOW13 000043SX
TOW16 000044SX
TOW17 000045SX
TOW2 000037SX
TOW3 000040SX
TOW7 000041SX
TOW9 000042SX
UMASK 000143SX
UMUL 000107'
VOLUM 000155SX
VOUT 000040
XEQPR 000114SX
ZABRT 000136SX
ZALS 000134SX
ZBITN 000135SX
ZCLER 000140SX
ZGET 000125SX
ZIJMP 000133SX
ZIN 000020
ZOR 000126SX
ZPTSB 000141SX
ZPUT 000124SX
ZRSTR 000132SX
ZSMUL 000137SX
ZSTFL 000130SX
ZSTPR 000127SX
ZWAIT 000131SX

```

PROGRAM SEGMENT 2


```

---
      .TITL      SEGM2
      .EXTN      TASK0, TASK1, TASK2, TASK3, IDLE
      .ENT        INTR, INTRP, WAIT
000015  ARTC    =      RTC+1-DEBUG      ; DEFINE ACTUATOR RTC
; INITIALIZATION AND RESTART
000000'062677  INTR:  IORST          ; RESTE ALL IO
000001'020142$  LDA      0, IMASK ; INHIBIT ALL INTERRUPTS
000002'062177  DOBS      0, CPU   ; EXCEPT POWER FAIL
000003'020144$  LDA      0, ASTLF ; ABORT STALL FLAG
000004'101004  .SZR      0, 0   ; SKIP IF NO ABORT
000005'063077  HALT          ; STOP.
000006'126000  .GM1      1, 1   ; AC1 = -1
000007'176520  .GPI      3, 3   ; AC3 = +1
; DELAY TO LET RTC STABILIZE
00010'030054$  LDA      2, C16 ; 16 OUTER LOOPS
      .IFN      DEBUG      ; *****
LDA      2, C32 ; 32 IN DEBUG MODE
      .ENDC          ; *****
000000  RDLOP:          ;
000000  .IFE      DEBUG ; DEBUG MODE *
00011'075057  DOA      3, STL  ; RESET STALL ALARM *
      .ENDC          ;
00012'101404  INC      0, 0, SZR ; 65K INNER LOOPS
00013'000776  JMP      RDLOP ;
000000  .IFE      DEBUG ; NO OUTPUT FOR DEBUG RUNS *
00014'061042  DOA      0, GRN  ; ALL GREENS OFF
00015'075041  DOA      3, AMB  ; FLASH ON
00016'061040  DOA      0, RED  ; ALL REDS OFF *****
      .ENDC          ;
00017'133004  ADD      1, 2, SZR ; DECREMENT AC2.
00020'000771  JMP      RDLOP ; LOOP.
; EMPTY I/O BUFFERS
00021'020104$  LDA      0, TIFIR ; STORE (FIRST) FOR TELEMETRY IE
00022'040105$  STA      0, TIIN  ; INTO IN AND
00023'040106$  STA      0, TIOUT ; OUT.
00024'020110$  LDA      0, TOFIR ; SAME FOR TELEMETRY OUTPUT BUFB
00025'040111$  STA      0, TOIN  ;
00026'040112$  STA      0, TOOUT ;
; INITIALIZE TASK CONTROL PARAMETERS
00027'020421  LDA      0, AIADD ; ADDR-1 OF TASK ADDRS
00030'040004$  STA      0, AINC2 ; STORE IN AUTO INCR LOC 2
00031'030113$  LDA      2, MHTSK ; AC2 =-HIGHEST TASK #
00032'102520  ILOOP: .GPI      0, 0   ; INITIALIZE STATE,
00033'041026$  STA      0, ESTAT, 2 ;
00034'022004$  LDA      0, @AINC2 ; STARTING ADDRESS (PC)
00035'101120  MOVZL     0, 0   ; CLEAR CARRY,
00036'041027$  STA      0, EPCSA, 2 ;
00037'021025$  LDA      0, ECYCL, 2 ; PLACE CYCLE TIME
00040'041034$  STA      0, ETIME, 2 ; IN TIMER
00041'151404  INC      2, 2, SZR ; INCR TASK INDEX
00042'000770  JMP      ILOOP ; BACK FOR NEXT TASK.
      .IFN      DEBUG      ; *****
LDA      0, ZIDLE ; PLACE ADDR OF IDLE
MOVZL     0, 0   ; JOB IN PC SAVE IN
STA      0, EPCSA ; DEBUG MODE
      .ENDC          ; *****
00043'102460  .ZAC      0, 0   ; CLEAR
00044'042403  STA      0, @ATMOD ; TLM INPUT MODE.
00045'002401  JMP      @ZGSCD ; GO TO SCHEDULER
      .IFN      DEBUG      ; *****

```

```

---
      ZIDLE:          IDLE      ; ADDR OF IDLE
      .ENDC          ;
00046'000453' ZGSCD:      GSCED      ; ADDR OF SCHEDULER
00047'000256' ATMOD:      TMODE      ; ADDR OF TLM MODE.
00050'000050' AIADD:      IADDR-1    ; ADDR-1 OF IADDR
      ; TASK INITIAL VALUES
      ; STARTING LOCS
00051'177777 IADDR:      TASK0      ;
00052'177777          TASK1      ;
00053'177777          TASK2      ;
00054'177777          TASK3      ;
;
;
; TELEMETRY VALUES
000007 IBTTM = 7 ; BIT TIME IN CLOCK PULSES
000014 ITNBT = 12. ; TOTAL NO. OF BITS IN TLM FRAME
000005 IBDLA = 5 ; INTER-BIT DELAY
000124 IFRTM = IBTTM*ITNBT; FRAME TIME
; INTERRUPT PROCESSING
;
; POWER FAIL INTERRUPT
00055'020411 PFINT: LDA 0,JRSTR ; PLACE JMP TO RESTART ROUTINE
00056'040001$ STA 0,PCINT ; IN LOC 0.
00057'102460 .ZAC 0,0 ; CLEAR ACO.
00060'126520 .GPI 1,1 ; AC1 = +1
      000000 .IFE DEBUG ; DEBUG MODE *
00061'061042 DOA 0,GRN ; TURN OFF GREENS
00062'065041 DOA 1,AMB ; FLASH ON.
00063'061040 DOA 0,RED ; ALL REDS OFF.
      .ENDC ; *
00064'062677 IORST ; RESET ALL IO
00065'063077 HALT ; HALT FOR POWER SHUTDOWN
00066'002132$ JRSTR: JMP @ZRSTR ; NO, PERFORM RESTART.
;
; UNDEFINED INTERRUPT
00067'054413 UNINT: STA 3,A3RSV ; SAVE AC3
00070'034413 LDA 3,.NIOC ; NIOC 0 INSTRUCTION
00071'071477 INTA 2 ; GET INTERRUPT DEVICE CODE
00072'157000 ADD 2,3 ; FORM CLEAR DONE AND BUSY FLAGS
00073'054401 STA 3,+.1 ; INSTRUCTION AND STORE FOR NEXT
00074'060200 NIOC 0 ; CLEAR DEVICE.
00075'030404 LDA 2,A2RSV ; RESTORE AC2
00076'034404 LDA 3,A3RSV ; AND AC3.
00077'060177 INTEN ; ENABLE INTERRUPT
00100'002001$ JMP @PCINT ; RETURN TO INTERRUPTED PC.
00101'000000 A2RSV: 0 ; A2 INTERRUPT SAVE
00102'000000 A3RSV: 0 ; A3 INTERRUPT SAVE
00103'060200 .NIOC: NIOC 0 ; NIOC FOR CLEARING UNDEFINED DE
;
; INTERRUPT ENTRY POINT
;
00104'063777 INTRP: SKPDZ CPU ; TEST POWER FAILURE.
00105'000750 JMP PFINT ; YES.
00106'050773 STA 2,A2RSV ; NO, SAVE AC2.
00107'030144$ LDA 2,ASTLF ; IS ABORT/STALL FLAG SET.
00110'151004 .SZR 2,2 ;
00111'063077 HALT ; YES, ABORT.
00112'063615 SKPDN ARTC ; TEST RT CLOCK
00113'000754 JMP UNINT ; NO, UNDEFINED DEVICE
;

```

```

; REAL TIME CLOCK INTERRUPT
;
00114'060115      NIOS      ARTC      ; SET BUSY, CLEAR DONE FOR RTC.
00115'030115$    LDA        2,SCEDF ; SCHEDULER FLAG
00116'151004      .SZR      2,2      ; TEST SCHEDULER MODE
00117'000412      JMP        NOSAV   ; DON'T SAVE REGS.
00120'030114$    LDA        2,XEQPR ; AC2 = TASK INDEX
00121'041030$    STA        0,EA0SA,2 ; SAVE REGISTERS IN TASK
00122'045031$    STA        1,EA1SA,2 ; CONTROL AREA.
00123'020756      LDA        0,A2RSV ;
00124'041032$    STA        0,EA2SA,2 ;
00125'055033$    STA        3,EA3SA,2 ;
00126'020001$    LDA        0,PCINT ; SAVE PC AND CARRY.
00127'101100      MOVL      0,0      ;
00130'041027$    STA        0,EPCSA,2 ;
00131'020142$ NOSAV: LDA        0,IMASK ; MASK FOR RTC AND LOWER PRIORITY
00132'062177      DOBS      0,CPU   ; ENABLE HIGHER PRIORITY INTERRUPT
          000000      .IFE      DEBUG ; DEBUG MODE *
00133'061057      DOA        0,STL   ; RESET STALL ALARM *
          .ENDC      ; *
;

```

111

```
; TELEMETRY INPUT SECTION
;
      000000      .IFE      DEBUG      ; NOT DEBUG MODE      *
00134'060427    DIA      0,TIN      ; INPUT TELEMETRY BIT      *
      .ENDC      ;      *
      .IFN      DEBUG      ; DEBUG MODE      *
LDA      0,SINPT ; SIMULATED TELEMETRY INPUT      *
      .ENDC      ;      *
00135'030521    LDA      2,TMODE ; TELEMETRY INPUT MODE
00136'006133$   JSR      @ZIJMP ; BRANCH TO APPROPRIATE INPUT RE
00137'000154'   INTLM ; MODE 0
00140'000162'   WSTAR ; 1
00141'000171'   DSTAR ; 2
00142'000205'   FDATA ; 3
00143'000213'   DDATA ; 4
00144'000245'   PSTOP ; 5
00145'000271'   DSTOP ; 6

; TELEMETRY VALUES
00146'000005    SSTAR:      IBTTM-2 ; # OF START SAMPLES TO CHECK.
00147'000005    BDLAY:      IBDLA ; INTER-BIT DELAY.
00150'000011    NBITS:      ITNBT-3 ; # OF DATA BITS.
00151'000002    SDATA:      IBTTM-IBDLA; # OF DATA SAMPLES
00152'000003    STPDL:      3 ; STOP BIT DELAY
00153'000013    SSTOP:      IBTTM*2-3 ; # OF STOP BIT SAMPLES

; INITIALIZE TLM
00154'020061$  INTLM: LDA      0,RROTB ; SET RESTART
00155'040177$   STA      0,RRFLG ; FLAG.
00156'020510    LDA      0,NCCOM ; SET # OF CONSECUTIVE
00157'040510    STA      0,ICNTR ; COMMANDS TO REQUIRE.
00160'040503    STA      0,EXRSY ; SET EXTN RESYNC FLAG.
00161'000470    JMP      IOTPT ; GO OUTPUT

; WAITING FOR START BIT
00162'101112    WSTAR: MOVL# 0,0,SZC ; TEST START BIT
00163'000555    JMP      OUTPT ; NO, KEEP WAITING
00164'024762    LDA      1,SSTAR ; SET TMINT TO # OF START
00165'044150$   STA      1,TMINT ; SAMPLES TO CHECK (BIT TIME -2)
00166'126460    .ZAC      1,1 ; AC1=0.
00167'044470    STA      1,IGNOR ; CLEAR IGNORE FLAG
00170'000461    JMP      IOTPT ; GO OUTPUT

; DURING START BIT
00171'101112    DSTAR: MOVL# 0,0,SZC ; TEST START BIT
00172'010465    ISZ      IGNORE ; SET IGNORE FLAG IF NOT START B
00173'014150$   DSZ      TMINT ; IS THIS LAST START BIT SAMPLE.
00174'000544    JMP      OUTPT ; NO, GO OUTPUT.
00175'126460    .ZAC      1,1 ; YES, AC1=0.
00176'044520    STA      1,TINPT ; CLEAR INPUT WORD.
00177'044461    STA      1,CPARY ; CLEAR COMPUTED PARITY WORD.
00200'024747    LDA      1,BDLAY ; SET TIME OUT WORD TO
00201'044150$   STA      1,TMINT ; INTER-BIT DELAY TIME.
00202'024746    LDA      1,NBITS ; INITIALIZE BITS REMAINING
00203'044456    STA      1,BITRM ; TO # OF DATA BITS.
00204'000445    JMP      IOTPT ; GO OUTPUT.

; FIRST SAMPLE OF EACH DATA BIT
00205'014150$   FDATA: DSZ      TMINT ; TIME TO SAMPLE.
00206'000532    JMP      OUTPT ; NO. GO OUTPUT
00207'024742    LDA      1,SDATA ; YES, STORE # OF SAMPLES
00210'044150$   STA      1,TMINT ; INTO TIME-OUT WORD.
00211'040451    STA      0,DVALU ; SAVE FIRST SAMPLE.
00212'000437    JMP      IOTPT ; GO OUTPUT
```

```

; DURING DATA BIT
00213'024447 DDATA: LDA 1,DVALU ; IS CURRENT INPUT
00214'106414 .SEQ 0,1 ; VALUE = FIRST SAMPLE OF DATA E
00215'010442 ISZ IGNOR ; NO, SET IGNORE FLAG.
00216'014150$ DSZ TMINT ; IS THIS LAST SAMPLE OF DATA BT
00217'000521 JMP OUTPT ; NO. GO OUTPUT.
00220'024440 LDA 1,CPARY ; YES, UPDATE COMPUTED
00221'107000 ADD 0,1 ; PARITY FOR
00222'044436 STA 1,CPARY ; THIS WORD.
00223'101100 MOVL 0,0 ; MOVE DATA BIT TO CARRY.
00224'020472 LDA 0,TINPT ; LOAD CURRENT INPUT WORD.
00225'101100 MOVL 0,0 ; SHIFT NEW DATA BIT TO BIT 15.
00226'040470 STA 0,TINPT ; UPDATE INPUT WORD.
00227'014432 DSZ BITRM ; IS THIS THE LAST DATA BIT IN E
00230'000411 JMP NXBIT ; NO.
00231'125004 .SZR 1,1 ; YES, TEST COMPUTED PARITY.
00232'000404 JMP PAROK ; IF NON-ZERO, OK.
00233'020062$ LDA 0,PEOTB ; BAD PARITY. SET PARITY ERROR
00234'006127$ JSR @ZSTPR ; BIT IN PRIORITY OUTPUT WORD.
00235'010422 ISZ IGNOR ; SET IGNORE FLAG.
00236'024714 PAROK: LDA 1,STPDL ; SET TIME-OUT TO
00237'044150$ STA 1,TMINT ; STOP BIT DELAY TIME.
00240'000411 JMP IOTPT ; GO OUTPUT.
00241'014415 NXBIT: DSZ TMODE ; SET TMODE = 3.
00242'024705 LDA 1,BDLAY ; SET TIME-OUT TO
00243'044150$ STA 1,TMINT ; INTER-BIT DELAY TIME.
00244'000474 JMP OUTPT ; GO OUTPUT.

; PREPARE FOR STOP BITS
00245'014150$ PSTOP: DSZ TMINT ; TIME TO BEGIN STOP BIT SAMPLE.
00246'000472 JMP OUTPT ; NO, GO OUTPUT.
00247'024704 LDA 1,SSTOP ; SET TIME-OUT TO NUMBER OF
00250'044150$ STA 1,TMINT ; STOP BIT SAMPLES. (BIT TIME*2)
00251'010405 IOTPT: ISZ TMODE ; INCR MODE
00252'000466 JMP OUTPT ; GO OUTPUT

; TELEMETRY VALUES
00253'000374 RSTIM: 3*IFRTM ; RESYNC DELAY TIME
00254'000015 NBTSO: 13. ; TOTAL # OF BITS IN OUTPUT FRAE
00255'000007 BITTM: IBTTM ; BIT TIME
00256'000000 TMODE: 0 ; TELEMETRY INPUT MODE
; .IFN DEBUG ; *****
SINPT: 0 ; SIMULATED TELEMETRY INPUT *
; .ENDC ; *****

00257'000000 IGNOR: 0 ; IGNORE TELEMETRY INPUT FLAG
00260'000000 CPARY: 0 ; COMPUTED PARITY-TM INPUT
00261'000000 BITRM: 0 ; # OF REMAINING INPUT BITS
00262'000000 DVALU: 0 ; DATA INPUT VALUE - TM
00263'000000 EXRSY: 0 ; EXTERNAL RESYNC FLAG
00264'000000 OUTWD: 0 ; OUTPUT WORD - TM
00265'000000 STFLG: 0 ; STATUS SEND FLAG
00266'000001 NCCOM: 1 ; # OF CONSEC COMDS-1
00267'000001 ICNTR: 1 ; CONSEC COMD CNTR
00270'177767 LINPT: -9. ; LAST COMMAND INPUT

; DURING STOP BITS
00271'101112 DSTOP: MOVL# 0,0,SZC ; TEST STOP BIT.
00272'000406 JMP STPOK ; STOP BIT OK
00273'126520 .GPI 1,1 ; STOP BIT ERROR
00274'044146$ STA 1,RSFLG ; SET RESYNC FLAG
00275'020066$ LDA 0,CROTB ; SET COMD REJ
00276'006127$ JSR @ZSTPR ; BIT IN TLM OUTPUT
00277'000437 JMP SMOD1 ; SET TMODE=1.

```

```

---
      000000 STPOK:
00300'014150$ DSZ   TMINT   ; YES. LAST SAMPLE OF STOP BIT.
00301'000437  JMP   OUTPT   ; NO.
00302'024755  LDA   1,IGNOR ; YES, WAS THERE AN ERROR IN THE
00303'125004   .SZR   1,1   ;
00304'000426  JMP   BADWD   ; YES, IGNORE INPUT WORD.
00305'020411  LDA   0,TINPT ; INPUT WORD
00306'024762  LDA   1,LINPT ; LAST INPUT WORD
00307'040761  STA   0,LINPT ; UPDATE LAST WITH CURRENT
00310'106414   .SEQ   0,1   ; TEST SAME
00311'000423  JMP   RCCOM   ; NO, IGNORE
00312'014755  DSZ   ICNTR   ; DECR CONSEC COMD CNTR
00313'000423  JMP   SMOD1   ; NOT EXPIRED, IGNORE
00314'006124$ JSR   @ZPUT   ; PLACE WORD INTO INPUT BUFFER.
00315'000104$ TIFIR ; BUFFER POINTER LOCATION.
00316'000000 TINPT:  0   ; INPUT WORD.
00317'000413  JMP   BADWD   ; BUFFER FULL RETURN.
00320'020063$ LDA   0,IFBIT ; INPUT FORMAT BIT.
00321'123404  AND   1,0,SZR ; TEST FORMAT BIT = 0 OR 1.
00322'000412  JMP   RCCOM   ; =1.
00323'020064$ LDA   0,IABTB ; =0.
                                .IFN   DEBUG   ; *****
                                JMP   TSRSY   ; *****
                                .ENDC
00324'123404  AND   1,0,SZR ; TEST ABORT BIT.
00325'006136$ JSR   @ZABRT ; ABORT
00326'020065$ TSRSY: LDA   0,IRSYB ; TEST INPUT RESYNC BIT.
00327'123404  AND   1,0,SZR ;
00330'010733  ISZ   EXRSY   ; SET. SET EXTERNAL RESYNC FLAG.
00331'000403  JMP   RCCOM   ; GO SET MODE.
00332'020066$ BADWD: LDA   0,CROTB ; BAD INPUT. SET COMMAND
00333'006127$ JSR   @ZSTPR ; REJECT BIT FOR TM OUTPUT.
00334'020732  RCCOM: LDA   0,NCCOM ; RESTART CONSEC COMD
00335'040732  STA   0,ICNTR ; CNTR.
00336'126520  SMOD1: .GPI   1,1   ; AC1=1
00337'044717  SMODE: STA   1,TMODE ; SET TMODE
;

```

111

; TELEMETRY OUTPUT SECTION

```
;
00340'024723  OUTPT: LDA      1,EXRSY ; EXTERNAL RESYNC FLAG.
00341'125005      .SNZ      1,1 ; TEST SET.
00342'000413      JMP      NOEXR  ; NO.
00343'102460      .ZAC      0,0 ; ACO=0
00344'105520      INGZL     0,1 ; AC1=2
00345'040716      STA      0,EXRSY ; CLEAR EXTERNAL RESYNC FLAG.
00346'040146$     STA      0,RSFLG ; CLEAR INTERNAL RESYNC FLAG.
00347'044151$     STA      1,BITN  ; SET OUTPUT BIT # = 2.
00350'126000      .GM1      1,1 ; AC1=-1
00351'044713      STA      1,OUTWD ; SET OUTPUT WORD = ALL 1'S.
00352'040713      STA      0,STFLG ; CLR STATUS SEND FL
00353'024700      LDA      1,RSTIM ; SET TIME OUT=RESYNC TIME
00354'000404      JMP      STIMO  ; GO SET TIME-OUT
00355'014147$ NOEXR: DSZ      TMOUT  ; TEST OUTPUT TIMER.
00356'000465      JMP      TIMRS  ; NOT TIME FOR OUTPUT.
00357'024676      LDA      1,BITTM ; MOVE BIT TIME TO
00360'044147$ STIMO: STA      1,TMOUT ; OUTPUT TIMER.
00361'020703      LDA      0,OUTWD ; OUTPUT WORD.
           000000      .IFE      DEBUG  ; NOT DEBUG *
00362'061043      DOA      0,TOUT  ; OUTPUT BIT 0.
           .ENDC      ; *
           .IFN      DEBUG  ; DEBUG *
           SUBZR     1,1 ; SET BIT 0 MASK.
           LDA      2,TMINW ; TM PARALLEL INPUT
           AND      2,1 ; PICK OFF BIT 0
           MOVZL    2,2 ; POSITION NEXT BIT
           STA      2,TMINW ; AND RESTORE
           STA      1,SINPT ; STORE AS SIMULATED INPUT.
           .ENDC      ; *
00363'101120      MOVZL    0,0 ; SHIFT NEXT OUTPUT BIT TO BIT 0
00364'040700      STA      0,OUTWD ; RESTORE OUTPUT WORD.
00365'014151$     DSZ      BITN  ; DECREMENT OUTPUT BIT COUNT.
00366'000455      JMP      TIMRS  ; MORE BITS, CONTINUE.
00367'024665      LDA      1,NBTSO ; NO MORE BITS IN THIS WORD. SEE
00370'044151$     STA      1,BITN  ; COUNT TO TOTAL # OF OUTPUT BIT$
00371'030146$     LDA      2,RSFLG ; RESYNC FLAG.
00372'151005      .SNZ      2,2 ; TEST FOR RESYNC NEEDED
00373'000407      JMP      NORSY  ; NO
           ; RESYNC FLAG = 1
00374'102460      ORSYW: .ZAC      0,0 ; CLEAR
00375'040146$     STA      0,RSFLG ; RESYNC FLAG
00376'024173$     LDA      1,RSWRD ; RESYNC OUTPUT WORD.
00377'020067$     LDA      0,ORSYB ; RESYNC OUTPUT BIT.
00400'006126$     JSR      @ZOR  ; SET RESYNC BIT.
00401'000435      JMP      SOUTW  ; GO STORE RESYNC OUTPUT WORD.
           ; RESYNC FLAG = 0 -- NORMAL OUTPUT
           000000  NORSY:
           ;
00402'024174$     TPRI0: LDA      1,PRIWR ; CHECK PRIORITY OUTPUT WORD.
00403'020116$     LDA      0,DATAM ; DATA BITS MASK.
00404'110000      COM      0,2 ;
00405'133400      AND      1,2 ;
00406'050174$     STA      2,PRIWR ; OUTPUT WORD.
00407'123404      AND      1,0,SZR ; CHECK DATA BITS
00410'000426      JMP      SOUTW  ; GO OUTPUT PRIORITY WORD
           ; NORMAL OUTPUT
00411'006125$     NOPRF: JSR      @ZGET ; GET WORD FROM OUTPUT BUFFER
```

```

---
00412'000110$      TOFIR   ; BUFFER POINTERS
00413'000404      JMP     OBFEM   ; BUFFER EMPTY RETURN.
00414'105000      MOV     0,1     ; PLACE
00415'000421      JMP     SOUTW   ; WORD IN AC1.
00416'000003      MXSTT: 3       ; LAST STATUS WORD #.
00417'030646      OBFEM: LDA    2,STFLG ; GET STATUS SEND INDEX
00420'151400      INC     2,2     ; AND ADVANCE IT
00421'024775      LDA    1,MXSTT ; LAST FILLER FRAME
00422'132032      .SGE    1,2     ; SENT?
00423'152520      .GPI    2,2     ; YES, RECYCLE STATUS INDEX
00424'050641      STA    2,STFLG ; UPDATE STATUS INDEX.
00425'034176$     OUTST: LDA    3,ASTWR; AC1 = STATUS WORD.
00426'157000      ADD     2,3     ;
00427'025400      LDA    1,0,3    ;
00430'151234      MOVZR# 2,2,SZR ; TEST ST WRD 1
00431'000405      JMP     SOUTW   ; NO
00432'020177$     LDA    0,RRFLG ; OR RESTART STATUS
00433'006126$     JSR     @ZOR   ; BIT WITH STATUS WORD 1
00434'102460      .ZAC    0,0     ; CLEAR RESTART
00435'040177$     STA    0,RRFLG ; STATUS WORD
00436'020117$     SOUTW: LDA    0,OWRDM ; OUTPUT WORD MASK
00437'107400      AND     0,1     ; KEEP VALID BITS.
00440'020123$     LDA    0,STBTS ; START-STOP BITS
00441'107000      ADD     0,1     ; FORM FULL OUTPUT.
00442'044622      STOUT: STA    1,OUTWD ; STORE NEW OUTPUT WORD.
                                .IFN    DEBUG   ; *****
                                STA    1,STMOT ; UPDATE SIM OUTPUT
                                LDA    2,STMIN ; MOVE SIM INPUT
                                STA    2,TMINW ; TO TM INPUT WORD
                                JMP     TIMRS   ; GO DO TIMERS
                                TMINW: 0       ; TM INPUT WORD
                                STMIN: -1     ; SIM TM INPUT
                                STMOT: 0      ; SIM TM OUTPUT
                                .ENDC      ; *****
;
; TASK TIMER CONTROL
;
00443'030113$     TIMRS: LDA    2,MHTSK ; -HIGHEST TASK #
000000$           TMRLP:        ; TIMERLOOP
                                .IFN    DEBUG   ; *
                                LDA    0,ESTAT,2 ; IS THIS TASK DORMANT
                                .SNZ    0,0     ; STATE=0 MEANS DORMANT.
                                JMP     NEXTS   ; YES, IGNORE THIS TASK.
                                .ENDC      ; *
00444'015034$     DSZ     ETIME,2 ; NO, DECREMENT TASK TIMER.
00445'000404      JMP     NEXTS   ; TIMER STILL COUNTING.
                                .IFN    DEBUG   ; *
                                LDA    0,ESTAT,2 ; STATE OF TASK
                                MOVZR# 0,0,SZR ; TEST > 1
                                HALT    ; YES, HALT IF MISSED INTERRUPT
                                .ENDC      ; *
00446'025025$     LDA    1,ECYCL,2 ; TIMER COUNTED DOWN.
00447'045034$     STA    1,ETIME,2 ; SET TIMER TO INITIAL E
00450'011026$     ISZ     ESTAT,2 ; INCREMENT TASK STATE.
00451'151404      NEXTS: INC    2,2,SZR ; INCREMENT TASK INDEX
00452'000772      JMP     TMRLP   ; LOOP FOR ALL TASKS.
;
;
; ENTER SCHEDULER
;

```



```

00454'044113$ STA 1,EA0SA,2 ; SET EA0SA TO 1.
00455'020143$ LDA 0,UMASK ; INTERRUPT UNMASK
00456'062177 DOBS 0,CPU ; UNMASK ALL, ENABLE RUPTS
;
; TASK SCHEDULER
;
00457'030113$ SCHED: LDA 2,MHTSK ; AC2 = -HIGHEST TASK #.
00460'021026$ SCLOP: LDA 0,ESTAT,2 ; STATE OF TASK.
00461'101234 MOVZR# 0,0,SZR ; TEST >1.
00462'000414 JMP XEQTS ; YES, EXECUTE TASK.
00463'151404 INC 2,2,SZR ; NO, INCR TASK INDEX
00464'000774 JMP SCLOP ; LOOP FOR ALL TASKS.
; NOTICE THAT IF NO TASK IS ACTIVE, THE HIGHEST NUMBERED
; TASK (IDLE JOB) IS EXECUTED OR AN IDLE LOOP IS ENTERED
. IFN DEBUG ; *****
JMP XEQTS ; EXECUTE IDLE JOB *****
. ENDC ; *****
00465'020407 IDLOP: LDA 0,CLKCT ; SET DELAY TIME
00466'040407 STA 0,RTDYC ; TO ALLOW RTC RUPT.
00467'014406 DSZ RTDYC ; ALLOW RTC PLENTY OF
00470'000777 JMP .-1 ; TIME TO INTERRUPT.
00471'020203$ LDA 0,CLOKF ; NO, ENABLE RTC
00472'061115 DOAS 0,ARTC ; AT 1000 HZ
00473'000772 JMP IDLOP ; IDLE.
00474'000674 CLKCT: 444. ; 2 MSEC DELAY.
00475'000000 RTDYC: 0 ; DELAY COUNTER.
;
; EXECUTE TASK WHOSE INDEX IS IN AC2.
;
00476'050114$ XEQTS: STA 2,XEQPR ; SET EXECUTING TASK INDEX.
00477'021032$ LDA 0,EA2SA,2 ; PLACE A2 IN TEMP
00500'040416 STA 0,A2TEM ; LOCN
00501'021030$ LDA 0,EA0SA,2 ; RESTORE REGS FROM
00502'025031$ LDA 1,EA1SA,2 ; TASK CONTROL AREA.
00503'035033$ LDA 3,EA3SA,2 ;
00504'031027$ LDA 2,EPCSA,2 ;
00505'151220 MOVZR 2,2 ; RESTORE CARRY.
00506'050407 STA 2,RETRN ; SAVE PC.
00507'152460 .ZAC 2,2 ; AC2=0, PRESERVE CARRY.
00510'060277 INTDS ; DISABLE INTERRUPT DURING DISM
00511'050115$ STA 2,SCEDF ; CLEAR SCHEDULER FLAG.
00512'030404 LDA 2,A2TEM ; RESTORE AC2.
00513'060177 BEGXQ: INTEN ; ENABLE INTERRUPT AND
00514'002401 JMP @RETRN ; BEGIN TASK EXECUTION.
00515'000000 RETRN: 0 ; INTERRUPT RETURN ADDRESS.
00516'000000 A2TEM: 0 ; A2 SAVE.
; TASK COMPLETION SUBROUTINE -- CALLED BY JSR@ ZWAIT
;
00517'060277 WAIT: INTDS ; DISABLE RUPT
00520'030114$ LDA 2,XEQPR ; AC2 = EXECUTING TASK INDEX.
00521'175100 MOVL 3,3 ; SAVE PC AND CARRY.
00522'055027$ STA 3,EPCSA,2 ;
00523'015026$ DSZ ESTAT,2 ; DECREMENT TASK STATE.
. IFN DEBUG ; *****
LDA 0,ETIME,2 ; MOVE TIMER TO TASK COM
STA 0,ETETI,2 ; TIMER IN DEBUG *****
. ENDC ; *****
00524'000727 JMP GSCED ; ENTER SCHEDULER
;

```

•END ; END INTERRUPT SECTION

A0SAV	000015SX
A1SAV	000016SX
A2RSV	000101'
A2SAV	000017SX
A2TEM	000516'
A3RSV	000102'
A3SAV	000020SX
AIADD	000050'
AINC1	000003SX
AINC2	000004SX
AINC3	000005SX
AINC4	000006SX
AINC5	000007SX
ALMSK	000205SX
AMB	000041
AMTM	177777 X
APDMT	000171SX
ARTC	000015
ASTA	000153SX
ASTLF	000144SX
ASTWR	000176SX
ATMOD	000047'
AXFLB	000207SX
BADWD	000332'
BDLAY	000147'
BEGXQ	000513'
BIT	000052SX
BITN	000151SX
BITRM	000261'
BITTM	000255'
BOFLG	000161SX
C1	000060SX
C16	000054SX
C2	000057SX
C32	000053SX
C4	000056SX
C7	000167SX
C8	000055SX
CDABT	000101SX
CLKCT	000474'
CLOKF	000203SX
CNFLS	000166SX
CPARY	000260'
CROTB	000066SX
CYCLE	000012SX
DATAM	000116SX
DDATA	000213'
DSTAR	000171'
DSTOP	000271'
DVALU	000262'
EA0SA	000030SX
EA1SA	000031SX
EA2SA	000032SX
EA3SA	000033SX
ECOMC	000165SX
ECYCL	000025SX
EMOD	000164SX
EPCSA	000027SX
ESPCE	000024SX
ESTAT	000026SX

ETETI	000035\$X
ETIME	000034\$X
EXRSY	000263'
FDATA	000205'
FLTIM	000145\$X
FORMM	000121\$X
GLMSK	000206\$X
GRN	000042
GSCED	000453'
GSTA	000152\$X
HITSK	000004
IABTB	000064\$X
IADDR	000051'
IAFLB	000071\$X
IBDLA	000005
IBTTM	000007
ICNTR	000267'
IDLE	177777 X
IDLOP	000465'
IFBIT	000063\$X
IFRTM	000124
IGNOR	000257'
ILOOP	000032'
IMASK	000142\$X
INTR	000000'
INTCD	000210\$X
INTLM	000154'
INTRP	000104'
IONLB	000070\$X
IOTPT	000251'
IRSYB	000065\$X
ITNET	000014
JRSTR	000066'
LAKC0	000046\$X
LAKC1	000047\$X
LBYTE	000122\$X
LINPT	000270'
LOOPS	000160\$X
LOW4M	000120\$X
LRJC0	000050\$X
LRJC1	000051\$X
MAMTM	000170\$X
MHTSK	000113\$X
MXSTT	000416'
NBITS	000150'
NBTS0	000254'
NCCOM	000266'
NEXTS	000451'
NOEXR	000355'
NOPRF	000411'
NORSY	000402'
NOSAV	000131'
NSPBT	000016
NXBIT	000241'
OAFLB	000074\$X
OASTB	000076\$X
OBFEM	000417'
OBOFB	000077\$X
OCAKB	000072\$X
OCDAB	000100\$X

OEMRB	000075\$X
ORSYB	000067\$X
ORSYW	000374'
OUTPT	000340'
OUTST	000425'
OUTWD	000264'
OWRDM	000117\$X
PAROK	000236'
PCINT	000001\$X
PCSAV	000014\$X
PDMNT	177777 X
PDMT1	177777 X
PDMT2	177777 X
PDMT4	177777 X
PDMT5	177777 X
PDMT6	177777 X
PEOTB	000062\$X
PFINT	000055'
PRAER	000201\$X
PRGER	000202\$X
PRIWR	000174\$X
PRRER	000200\$X
PRZP1	000156\$X
PRZP2	000157\$X
PSTOP	000245'
PTMR	000011\$X
RCCOM	000334'
RDLOP	000011'
RED	000040'
RETRN	000515'
RLMSK	000204\$X
RRFLG	000177\$X
RROTB	000061\$X
RSFLG	000146\$X
RSTA	000154\$X
RSTIM	000253'
RSWRD	000173\$X
RTDYC	000475'
SCEDF	000115\$X
SCERB	000073\$X
SCHED	000457'
SCLOP	000460'
SDATA	000151'
SGSTA	000162\$X
SINPT	000000U
SMOD1	000336'
SMODE	000337'
SOUTW	000436'
SPCEL	000002\$X
SPTAB	000023\$X
SSDNB	000102\$X
SSTAR	000146'
SSTOP	000153'
STATE	000013\$X
STBTS	000123\$X
STFLG	000265'
STIMO	000360'
STL	000057'
STMIN	000000U
STMOT	000000U

STMR	000010\$X
STOUT	000442'
STPDL	000152'
STPOK	000300'
STWRD	000175\$X
TASK0	000051'X
TASK1	000052'X
TASK2	000053'X
TASK3	000054'X
TETIM	000022\$X
TIBUF	000103\$X
TIFIR	000104\$X
TIIN	000105\$X
TIMER	000021\$X
TIMRS	000443'
TIN	000027
TINPT	000316'
TIOUT	000106\$X
TMERF	000163\$X
TMINT	000150\$X
TMINW	000000U
TMODE	000256'
TMOUT	000147\$X
TMRLP	000444'
TOBUF	000107\$X
TOFIR	000110\$X
TOIN	000111\$X
TOOUT	000112\$X
TOUT	000043
TOUTW	000172\$X
TOW1	000036\$X
TOW13	000043\$X
TOW16	000044\$X
TOW17	000045\$X
TOW2	000037\$X
TOW3	000040\$X
TOW7	000041\$X
TOW9	000042\$X
TPRIO	000402'
TSRSY	000326'
UMASK	000143\$X
UNINT	000067'
VOLUM	000155\$X
VOUT	000040
WAIT	000517'
WSTAR	000162'
XEQPR	000114\$X
XEQTS	000476'
ZABRT	000136\$X
ZALS	000134\$X
ZBITN	000135\$X
ZCLER	000140\$X
ZGET	000125\$X
ZGSCD	000046'
ZIDLE	000000U
ZIJMP	000133\$X
ZIN	000020
ZOR	000126\$X
ZPTSB	000141\$X
ZPUT	000124\$X

ZRSTR 000132SX
ZSMUL 000137SX
ZSTFL 000130SX
ZSTPR 000127SX
ZWAIT 000131SX
.NIOC 000103'

PROGRAM SEGMENT 3

.TITL SEGM3

;
; TASK 0 - PROCESS INPUT Z DETECTORS FOR SPEEDS AND VOLUME
;

.ENT TASK0

000000 TASK0:
000000'034452 SPDET: LDA 3,MNSPB ;INIT SECTION. AC3 =-# OF SPEEDS
000001'126460 .ZAC 1,1 ; ACI = 0
000002'045424\$ T0INL: STA 1,ESPCE,3 ; CLEAR SPEED COUNTERS.
000003'175404 INC 3,3,SZR ; END OF LOOP.
000004'000776 JMP T0INL ; NO, CLEAR ALL CELLS.
000005'044155\$ STA 1,VOLUM ; CLEAR VOLUME CELLS.

; NOTE -- ACI MUST BE 0 AT THIS POINT SINCE IT CONTAINS
; VALUE OF INPUT Z.

000000 SPLOP: ; TASK 0 LOOP

.IFN DEBUG ; *
LDA 0,Z0DBG ; Z+0 DEBUG WORD
JMP .+3 ; SKIP INPUT

Z0DBG: 0 ; Z+0 DEBUG WORD

.ENDC ; *

000006'060420 DIA 0,ZIN ; CURRENT DETECTOR STATUS TO AC0
000007'030441 LDA 2,ZSPMS ; SPEED BITS MASK
000010'143405 AND 2,0,SNR ; MASK SPEED DETECTOR BITS AND
000011'000413 JMP NOT11 ; NONE.

000012'040437 STA 0,PRZIN ; SAVE SPEED BITS.
000013'152000 .GM1 2,2 ; AC2 = INITIAL BIT #
000014'006135\$ T1T01: JSR @ZBITN ; FIND NEXT ONE BIT IN AC0,PUT
000015'000405 JMP END11 ; NO MORE ONE BITS.
000016'011002\$ ISZ SPCEL,2 ; INCR SPEED COUNTER FOR ONE BIT
000017'000775 JMP T1T01 ; OK IF NON-ZERO. LOOP
000020'015002\$ DSZ SPCEL,2 ; DONT ALLOW OVERFLOW
000021'000773 JMP T1T01 ; LOOP FOR ALL ONE BITS.

000022'020427 END11: LDA 0,PRZIN ; RESTORE CURRENT INPUT TO AC0
000023'000402 JMP .+2

000024'040425 NOT11: STA 0,PRZIN ; SAVE SPEED BIS
000025'100000 COM 0,0 ; .NOT. SPEED BITS TO AC0
000026'123405 AND 1,0,SNR ; PLACE 1-0 TRANSITIONS IN AC0
000027'000416 JMP NOT10 ; NO 1-0 TRANSITIONS

000030'034155\$ LDA 3,VOLUM ;AC3 = INPUT Z VOLUME BITS
000031'110000 COM 0,2 ; OR 1-0 TRANSITIONS
000032'157400 AND 2,3 ; WITH EXISTING
000033'156000 ADC 2,3 ; VOLUMES.

000034'054155\$ STA 3,VOLUM ; UPDATE VOLUME BITS
000035'152000 .GM1 2,2 ; SET AC2 TO INITIAL BIT #

000036'006135\$ T1T00: JSR @ZBITN ; FIND NEXT 1-0 BIT IN AC0.
000037'000406 JMP NOT10 ; NONE

000040'035002\$ LDA 3,SPCEL,2 ; MOVE NEW SPEED VALUE
000041'055023\$ STA 3,SPTAB,2 ; INTO SPEED TABLE
000042'176460 .ZAC 3,3 ; CLEAR SPEED CELL ENTRY TO PREB
000043'055002\$ STA 3,SPCEL,2 ; FOR NEXT ACTUATION.

000044'000772 JMP T1T00 ; LOOP FOR ALL 1-0 BITS.
000045'006131\$ NOT10: JSR @ZWAIT ; WAIT FOR NEXT TASK TIME FRAME.
000046'024403 LDA 1,PRZIN ; PREVIOUS INPUT Z TO AC1
000047'000737 JMP SPLOP ; LOOP FOR NEW TIME FRAME

; TASK 0 LOCAL PARAMETERS

00050'177774 ZSPMS: 177774 ; SPEED BITS MASK FOR INPUT Z
00051'000000 PRZIN: 0 ; PREVIOUS INPUT Z
00052'177762 MNSPB: -NSPBT ; -# OF SPEED BITS.

.END

```

---
AGSAV 000015SX
A1SAV 000016SX
A2SAV 000017SX
A3SAV 000020SX
AINC1 000003SX
AINC2 000004SX
AINC3 000005SX
AINC4 000006SX
AINC5 000007SX
AMB 000041
AMTM 177777 X
APDMT 000171SX
ASTA 000153SX
ASTLF 000144SX
ASTWR 000176SX
BIT 000052SX
BITN 000151SX
BOFLG 000161SX
C1 000060SX
C16 000054SX
C2 000057SX
C32 000053SX
C4 000056SX
C7 000167SX
CS 000055SX
CDABT 000101SX
CNFLS 000166SX
CROTD 000066SX
CYCLE 000012SX
DATAM 000116SX
DEBUG 000000
FA0SA 000030SX
EA1SA 000031SX
EA2SA 000032SX
EA3SA 000033SX
ECOMC 000165SX
ECYCL 000025SX
EMMOD 000164SX
END11 000022'
EPCSA 000027SX
ESPCE 000024SX
ESTAT 000026SX
ETETI 000035SX
ETIME 000034SX
FLTIM 000145SX
FORMM 000121SX
GRN 000042
GSTA 000152SX
HITSK 000004
IARTE 000064SX
IAFLB 000071SX
IFBIT 000063SX
IMASK 000142SX
IONLB 000070SX
IRSYB 000065SX
LAKC0 000046SX
LAKC1 000047SX
LBYTE 000122SX
LOOPS 000160SX
LOW4M 000120SX

```

 LRJC0 000050\$X
 LRJC1 000051\$X
 MAMTM 000170\$X
 MHTSK 000113\$X
 MNSPB 000052'
 NOT10 000045'
 NOT11 000024'
 NSPBT 000016
 OAFLE 000074\$X
 OASTB 000076\$X
 OBOFB 000077\$X
 OCAKB 000072\$X
 OCDAB 000100\$X
 OEMRB 000075\$X
 ORSYB 000067\$X
 OWRDM 000117\$X
 PCINT 000001\$X
 PCSAV 000014\$X
 PDMNT 177777 X
 PDMT1 177777 X
 PDMT2 177777 X
 PDMT4 177777 X
 PDMT5 177777 X
 PDMT6 177777 X
 PEOTB 000062\$X
 PRIWR 000174\$X
 PREIN 000051'
 PREP1 000156\$X
 PREP2 000157\$X
 PTMR 000011\$X
 RED 000040
 RRFLG 000177\$X
 RROTB 000061\$X
 RSFLG 000146\$X
 RSTA 000154\$X
 RSPRD 000173\$X
 SCEDF 000115\$X
 SCERB 000073\$X
 SGSTA 000162\$X
 SPCEL 000002\$X
 SPDET 000000'
 SPLOP 000006'
 SPTAB 000023\$X
 SSDNB 000102\$X
 STATE 000013\$X
 STBTS 000123\$X
 STL 000057
 STMR 000010\$X
 STWRD 000175\$X
 T0INL 000002'
 T1T00 000036'
 T1T01 000014'
 TASK0 000000'
 TETIM 000022\$X
 TIBUF 000103\$X
 TIFIR 000104\$X
 TIIN 000105\$X
 TIMER 000021\$X
 TIN 000027
 TIOUT 000106\$X

```

---
TMERF 000163$X
TMINT 000150$X
TMOUT 000147$X
TOBUF 000107$X
TOFIR 000110$X
TOIN 000111$X
TOOUT 000112$X
TOUT 000043
TOUTW 000172$X
TOW1 000036$X
TOW13 000043$X
TOW16 000044$X
TOW17 000045$X
TOW2 000037$X
TOW3 000040$X
TOW7 000041$X
TOW9 000042$X
UMASK 000143$X
VOLUM 000155$X
VOUT 000040
XEQPR 000114$X
XDBG 000000U
XABRT 000136$X
XALS 000134$X
XBITN 000135$X
XCLER 000140$X
XGET 000125$X
XJMP 000133$X
XIN 000020
XOR 000126$X
XPTSB 000141$X
XPUT 000124$X
XRSTR 000132$X
XSMUL 000137$X
XSPMS 000050'
XSTFL 000130$X
XSTPR 000127$X
XWAIT 000131$X

```

PROGRAM SEGMENT 4

.TITL SEGM4

; TASK 1 - DETECTOR PROCESSING AND TM OUTPUT BUFFERING

.ENT TASK1,Z1VB5

000000 TASK1:
000000'102460 DETPR: .ZAC 0,0 ; INIT. CLEAR AC0
000001'040156\$ STA 0,PRZP1 ; CLFAR PREVIOUS INPUT Z+1
000002'040452 STA 0,0OCCUP ; OUTPUT OCCUP,
000003'040447 STA 0,0VOL ; OUTPUT VOL
000004'040447 STA 0,0VOL+1; WORDS,
000005'040157\$ STA 0,PRZP2 ; PREVIOUS INPUT Z+2
000006'024433 LDA 1,MOCUP ; AC1 = ADDR(OCCUP-1)
000007'044003\$ STA 1,AINC1 ; STORE IN AUTO INCR LOC 1
000010'030432 LDA 2,MNOCU ; AC2 = -# OF OCCUP CELLS
000011'034427 LDA 3,OCINI ; AC3 = INITIAL OCCUP VALUE
000012'056003\$ STA 3,@AINC1; INIT OCCUP CELL AND INCR TO N#
000013'151404 INC 2,2,SZR ; INCR OCCUP CELL COUNT
000014'000776 JMP .-2 ; LOOP FOR ALL
000015'034426 LDA 3,ATINI ; INIT VALUE FOR AVER TRAP TIME
000016'054426 STA 3,AVTRP ; INIT ALL AVER TRAP CELLS
000017'054426 STA 3,AVTRP+1;
000020'054426 STA 3,AVTRP+2;
000021'054426 STA 3,AVTRP+3;
000022'000434 JMP DTLOP ; START TASK1

; SOME LOCAL TASK 1 PARAMETERS

000015 OCCUP: .BLK 13. ; OCCUPANCY COUNTERS
000040'177634 OCINI: -100. ; INITIAL OCCUP VALUE (5 SEC)
000041'000022' MOCUP: OCCUP-1 ; ADDR-1 OF OCCUP
000042'177763 MNOCU: -13. ; -# OF OCCUP CELLS
000043'000050 ATINI: 40. ; INIT VALUE FOR AVTRP (45 FPS)
000004 AVTRP: .BLK 4 ; AVER TRAP TIME CELLS
000050'113130 ZP10M: 113130 ; Z+1 OCCUP BIT MASK
000051'000363 AZ10B: Z10BT-OCCUP ; ADDR OF Z10BT
000052'000000 OVOL: 0 ; OUTPUT VOL WORD 1
000053'000000 0 ; OUTPUT VOL WORD 2
000054'000000 OOCUP: 0 ; OUTPUT OCCUP
000055'000264' ASVLB: SVOLB ; ADDR OF SVOLB

; TASK 1 LOOP PROCESS Z+1 VOLUMES

000000 DTLOP:

.IFN DEBUG ; *
LDA 1,Z1DBG ; SET CURRENT INPUT = DEBUG WOR
JMP .+3 ; SKIP INPUT

Z1DBG:

.ENDC ; *

000056'064421 DIA 1,ZIN+1 ; READ INPUT Z+1 INTO AC1
000057'006776 JSR @ASVLB ; GO STORE VOLUME BITS
000060'000370' Z1VBT-1 ; Z+1 VOL BIT TABLE
000061'157170 157170 ; Z+1 VOL MASK
000062'000156\$ PRZP1 ; PREVIOUS Z+1

; Z+1 OCCUPANCY PROCESSING

000063'030765 LDA 2,ZP10M ; OCCUPANCY BITS MASK
000064'020156\$ LDA 0,PRZP1 ; CURRENT Z+1 INPUT
000065'143405 AND 2,0,SNR ; KEEP VALID OCCUPANCY BITS. TEB
000066'000421 JMP DNO11 ; NONE
000067'030752 LDA 2,MOCUP ; ADDR OF OCCUP-1
000070'006135\$ D1TO1: JSR @ZBITN ; FIND NEXT 1 BIT IN ACO, PUT #B
000071'000416 JMP DNO11 ; NO MORE 1 BITS.
000072'011000 ISZ 0,2 ; INCR OCCUP COUNTER FOR EACH 1#
000073'000775 JMP D1TO1 ; NO OFLO

```

---
00074'034755      LDA      3,AZ10B ; OFLO. INDEX Z10RT.
00075'157000      ADD      2,3
00076'035400      LDA      3,0,3 ; GET OCCUP OUTPUT BIT
00077'024755      LDA      1,00CUP ; GETOUTPUT OCCUP WORD
00100'174000      COM      3,3 ; OR NEW OCCUP BIT
00101'167400      AND      3,1 ;
00102'166000      ADC      3,1 ;
00103'044751      STA      1,00CUP ; RESTORE OCCUP OUTPUT WORD
00104'024734      LDA      1,0CINI ; INIT OCCUP COUNTER
00105'045000      STA      1,0,2 ; INIT VALUE TO OCCUP COUNTER
00106'000762      JMP      D1T01 ; LOOP FOR ALL OCCUP BITS
      000000      DNO11: ;
;
; PROCESS DETECTOR INPUT Z+2 VOLUMES
      .IFN      DEBUG ; *
      LDA      1,Z2DBG ; DEBUG INPUT Z+2 TO AC1
      JMP      .+3 ; SKIP INPUT
Z2DBG:      0 ;
      .ENDC ; *
00107'064422      DIA      1,ZIN+2 ; READ CURRENT Z+2 STATUS
00110'006745      JSR      @ASVLB ; GO STORE VOL BITS
00111'000422'      Z2VBT-1 ; Z+2 VOL BIT TABLE
00112'054540      054540 ; Z+2 VOL MASK
00113'000157$      PRZP2 ; PREVIOUS Z+2
      .EOT

```

; PROCESS INPUT Z VOLTS AND SPEEDS

```

;
00114'152460      .ZAC      2,2      ; CLEAR AC2
00115'060277      INTDS          ; DONT ALLOW TASK 0
00116'020155$     LDA      0,VOLUM ; GET INPUT Z VOL BITS
00117'050155$     STA      2,VOLUM ; CLEAR VOL BITS
00120'060177      INTEN          ; ALLOW TASK 0
00121'152000      .GM1      2,2      ; AC2=INITIAL BIT #
00122'006135$     SCLOP: JSR      @ZBITN ;GET # OF NEXT VOL BIT IN AC2
00123'000465      JMP      SNOMO   ; NO MORE BITS SET.
00124'040453      STA      0,TEM11 ; SAVE REMAINING VOL BITS
00125'034454      LDA      3,AZ0VB ; INDEX Z0VBT
00126'157000      ADD      2,3      ;
00127'025400      LDA      1,0,3   ; GET TELEM OUTPUT VOL BIT
00130'020722      LDA      0,OVOL  ; TM OUTPUT VOL WORD
00131'006126$     JSR      @ZOR   ; SET VOL BIT
00132'044720      STA      1,OVOL  ; RESTORE OUTPUT WORD
00133'025023$     LDA      1,SPTAB,2 ; LOAD SPEED BUFFER ENTR
00134'020446      LDA      0,MINTP ; MINIMUM OCCUP TIME
00135'034446      LDA      3,MAXTP ; MAXIMUM OCCUP TIME
00136'106432      .SGT      0,1    ; TEST OCCUP TIME
00137'166032      .SGE      3,1    ; IN LIMITS.
00140'000435      JMP      SKSPD  ; NO, IGNORE
00141'034443      LDA      3,AAVTR ; INDEX AVTRA
00142'157000      ADD      2,3      ;
00143'023400      LDA      0,@0,3  ; CURRENT AVER TRAP TIME
00144'050434      STA      2,TEM12 ; SAVE BIT #
00145'034440      LDA      3,IVART ; AC3-ALLOWABLE VARIATION OVER
00146'117000      ADD      0,3      ; AVTRP+IVART
00147'136032      .SGE      1,3    ; TEST TPTIM IN RANGE OF AVER.
00150'000403      JMP      TPTOK  ; YES. AC1=TPTIM
00151'024435      LDA      1,IVARA ; NO, SET TPTIM TO
00152'107000      ADD      0,1     ; AVTRP+IVARA. PUT IN AC1
00153'030434      TPTOK: LDA      2,MXLAM ; AC2=MAXIMUM LAMBDA
00154'132432      .SGT      1,2    ; TEST TPTIM<MAX LAMBDA
00155'131000      MOV      1,2     ; NO, USE MAX LAMBDA
00156'106400      SUB      0,1     ; TPTIM-AVTRP TO AC1
00157'006137$     JSR      @ZSMUL  ; SET AC0,AC1=AC1.AC2=(TPTIM-AVE
00160'125100      MOVL     1,1     ; PERFORM ARS OF 7 BY SHIFTING
00161'125300      MOVS     1,1     ; SWAPPING HALVES OF L.S. WORD.
00162'101100      MOVL     0,0     ; MOVE CARRY TO LSB OF MOST SIG
00163'030122$     LDA      2,LBYTE ; MASK FOR LOWER BYTE
00164'143700      ANDS     2,0     ; MASK LOWER BYTE AND MOVE TO UB
00165'147400      AND      2,1     ; MASK LOWER BYTE OF L.S. WORD.
00166'107000      ADD      0,1     ; MERGE BOTH BYTES. THE RESULT
; ; A TRUE DOUBLE SHIFT OF 7, OR
00167'030411      LDA      2,TEM12 ; RESTORE BIT# TO AC2
00170'034414      LDA      3,AAVTR ; INDEX AVTRA
00171'157000      ADD      2,3      ;
00172'023400      LDA      0,@0,3  ; AC0=CURRENT AVER TRAP TIME
00173'123000      ADD      1,0     ; AVTRP=AVTRP+(TPTIM-AVTRP)*LAMB
00174'043400      STA      0,@0,3  ; RESTORE UPDATED AVER TRAP TIME
00175'020402      SKSPD: LDA      0,TEM11 ; RESTORE VOL BITS TO AC0.
00176'000724      JMP      SCLOP  ; LOOP FOR ALL VOL BITS
;
; MORE TASK1 LOCAL PARAMETERS
;
00177'000000      TEM11:      0      ; TEMP
00200'000000      TEM12:      0      ; TEMP
00201'000353'     AZ0VB:      Z0VBT ; ADDR OF Z0VBT

```

```
00202'000024 MINTP:      20.      ; MIN TRAP TIME (90 FPS)
00203'003410 MAXTP:      1800.     ; MAX TRAP TIME (1 FPS)
00204'000436' AAVTR:      AVTRA     ; ADDR OF AVTRA
00205'000036 IVART:      30.      ; TRAP TIME ALLOWABLE VARIATION
00206'000017 IVARA:      15.      ; AVER TRAP TIME INCR IF VARIAT
00207'000200 MXLAM:      128.     ; MAX LAMBDA
;
000000 SNOMO:
;
; PLACE DETECTOR DATA IN TM OUTPUT BUFFER
;
00210'004511          JSR      PTMDT   ; PLACE DET VOLTS IN TM OUTPUT RE
00211'000052'          OVOL     ; GROUP WILL BE PUT IN BUFFER IF
00212'000004          4        ; 4 GROUPS
00213'000042$        TOW9     ; OUTPUT FORMAT 9
00214'004505          JSR      PTMDT   ; SAME FOR PED INPUTS. GROUP WIE
00215'000053'          OVOL+1  ; ZEROED IF PUT IN BUFFER.
00216'000002          2        ; 2 GROUPS
00217'000043$        TOW13    ; OUTPUT FORMAT 13
00220'004501          JSR      PTMDT   ; SAME FOR OCCUP BITS. IF BUFFED
00221'000054'          OCCUP    ; GROUP WILL NOT BE ZEROED.
00222'000002          2        ; 2 GROUPS
00223'000041$        TOW7     ; OUTPUT FORMAT 7
00224'020627          LDA      0,OVOL+1; MOVE PED BITS TO
00225'101300          MOVS     0,0    ; UPPER BYTE
00226'040625          STA      0,OVOL+1;
00227'020625          LDA      0,OCCUP ; MOVE OCCUP BITS
00230'101300          MOVS     0,0    ; TO UPPER BYTE
00231'040623          STA      0,OCCUP ;
;
; ENCODE AVERAGE TRAP TIMES TO 4 BIT CODES AND PLACE
; IN TM OUTPUT BUFFER IF CHANGED
;
00232'020056$        LDA      0,C4    ; STORE # OF AVER TRAP
00233'040744          STA      0,TEM11 ; TIME CELLS IN COUNTER
00234'020513          LDA      0,AAVRT ; PUT ADDR OF AVTRP-1
00235'040003$        STA      0,AINC1 ; IN AUTO-INCR LOC 1
00236'020512          LDA      0,ATMSO ; STORE FIRST SPEED CODE
00237'040416          STA      0,SOFMN ; OUTPUT FORMAT ADDR
00240'020511 AVTLP: LDA      0,FOTCD ; PLACE FIRST OUTPUT CODE #
00241'040737          STA      0,TEM12 ; IN COUNTER.
00242'020510          LDA      0,ASPDC ; PUT ADDR OF SPEED CODE TABLE-1
00243'040004$        STA      0,AINC2 ; IN AUTO:INCR LOC 2.
00244'022003$        LDA      0,@AINC1; AVER TP TIME TO AC0. INCR TO
00245'026004$ ASCLP: LDA      1,@AINC2; TRAP TIME THRESHOLD TO AC1. I
00246'122432          .SGT      1,0    ; TEST TRAP TIME > THRESHOLD.
00247'000403          JMP      FOUNS   ; YES, FOUND SPEED CODE
00250'014730          DSZ      TEM12   ; DECR SPEED CODE #. IF=0,USE 0.
00251'000774          JMP      ASCLP   ; IF NOT 0 KEEP SEARCHING
00252'102000 FOUNS:  .GM1      0,0    ; SET COMPARE FLAG
00253'006141$        JSR      @ZPTSB  ; PLACE SPD CODE IN OUTPUT BUFFE
00254'000200'        TEM12     ; DIFF FROM LAST OUTPUT.
00255'000172$ SOFMN: TOUTW    ; OUTPUT FORMAT ADDR
00256'010777          ISZ      SOFMN   ; INCR OUTPUT FORMAT ADDR
00257'014720          DSZ      TEM11   ; DECR AVER SPEED CELL COUNT.
00260'000760          JMP      AVTLP   ; LOOP FOR ALL CELLS.
; END OF FRAME
00261'006131$        JSR      @ZWAIT  ; WAIT FOR NEXT TASK 1 FRAME
00262'002401          JMP      @.+1   ; LOOP FOR NEW FRAME
00263'000056'        DTLOP     ; ADDR OF START OF LOOP
```

;

.EOT

```
;
; SUBROUTINE TO STORE VOLUME BITS INTO OVOL FOR FUTUR
; TM OUTPUT. CALL WITH VOLUME BITS IN AC1.
;
; JSR SVOLB ; CALL SUBR
; BITBL-1 ; ADDRESS OF BIT TABLE-1
; MASK ; MASK TO BE APPLIED TO VOL BITS
; PREV ; PREVIOUS VOL BITS
; --- ---- ; RETURN
; BIT TABLE CONTAINS THE OUTPUT BIT FOR EACH BIT IN B
; IS SET IN ADDITION TO ANOTHER BIT, THE OTHER BIT WE
; INTO OVOL+1. DESTROYS ALL REGISTERS.
;
00264'023402 SVOLB: LDA 0,02,3 ; GET PREVIOUS VOL BITS
00265'031401 LDA 2,1,3 ; VOL MASK
00266'147400 AND 2,1 ; KEEP VOL BITS
00267'047402 STA 1,02,3 ; UPDATE PREVIOUS VALUE
00270'100000 COM 0,0 ; NOT PREVIOUS
00271'123405 AND 1,0,SNR ; FIND 0-1 TRANSITIONS
00272'001403 JMP 3,3 ; NONE, RETURN
00273'054423 STA 3,SSRET ; STORE RETURN ADDR.
00274'031400 LDA 2,0,3 ; ADDR-1 OF BIT TABLE
00275'006135$ SVLLP: JSR @ZBITN ; INCR AC2 BY # OF NEXT BIT SET
00276'000416 JMP SVOUT ; NONE SET, RETURN.
00277'040420 STA 0,SSAV0 ; SAVE REMAINING VOL BITS.
00300'034420 LDA 3,AOVOL ; AC3 = ADDR OF OVOL
00301'021000 LDA 0,0,2 ; AC0 = ENTRY FROM BIT TABLE.
00302'101226 MOVZR 0,0,SEZ ; BIT 15 TO CARRY. TEST BIT 15
00303'175420 INCZ 3,3 ; YES, SET AC3 = OVOL+1. CLEAR
00304'101100 MOVL 0,0 ; REPOSITION BIT TABLE ENTRY.
00305'025400 LDA 1,0,3 ; OVOL
00306'100000 COM 0,0 ; OR BIT TABLE ENTRY
00307'107400 AND 0,1 ;
00310'106000 ADC 0,1 ;
00311'045400 STA 1,0,3 ; RESTORE OVOL
00312'020405 LDA 0,SSAV0 ; RESTORE REMAINING VOL BITS.
00313'000762 JMP SVLLP ; LOOP.
00314'034402 SVOUT: LDA 3,SSRET ; RETURN TO
00315'001403 JMP 3,3 ; CALL+4
00316'000000 SSRET: 0 ; RETURN ADDR
00317'000000 SSAV0: 0 ; TEMP SAVE FOR AC0
00320'000052' AOVOL: OVOL ; ADDR OF OUTPUT VOL WORD.
;
;
; SUBROUTINE TO PLACE DATA BITS IN TM OUTPUT BUFFER.
; JSR PTMDT ; CALL SUBR
; ARG0 ; LOCATION CONTAINING DATA BITS
; ARG1 ; # OF 4 BIT GROUPS OF DATA BITS
; ARG2 ; ADDR OF OUTPUT FORMAT FOR LEFT
; --- ---- ; RETURN
00321'021401 PTMDT: LDA 0,1,3 ; STORE # OF GROUPS.
00322'040423 STA 0,PTMC ;
00323'021402 LDA 0,2,3 ; STORE STARTING OUTPUT FORMAT B
00324'040413 STA 0,PTFMN ;
00325'021400 LDA 0,0,3 ; STORE LOCATION OF DATA BITS
00326'040410 STA 0,PTLCN ;
00327'054417 STA 3,PTDRT ; SAVE RETURN ADDR.
00330'022406 PTMLP: LDA 0,@PTLCN ; AC0 = DATA WORD
00331'030056$ LDA 2,C4 ; AC2 = SHIFT COUNT
00332'006134$ JSR @ZALS ; CIRCULAR LEFT SHIFT DATA WORD
```

```

---
00333'042403      STA      0,@PTLCN; STORE SHIFTED DATA WORD.
00334'102460      .ZAC      0,0      ; SET NO-COMPARE FLAG
00335'006141$     JSR      @ZPTSB. ; GO STORE 4 BITS.
00336'000000      PTLCN:    0      ; DATA WORD LOCN.
00337'000000      PTFMN:    0      ; OUTPUT FORMAT ADDR
00340'010777      ISZ      PTFMN  ; INCR OUTPUT FORMAT ADDR
00341'014404      DSZ      PTMC   ; DECR GROUP COUNT.
00342'000766      JMP      PTMLP  ; LOOP FOR ALL GROUPS.
00343'034403      LDA      3,PTDRT ; RETURN LOC.
00344'001403      JMP      3,3    ; RETURN.
00345'000000      PTMC:    0      ; GROUP COUNTER.
00346'000000      PTDRT:    0      ; RETURN LOC.
;
      .EOT

```

```

;
; MORE TASK 1 PARAMETERS
00347'000043' AAVRT: AVTRP-1 ; ADDR OF AVTRP-1
00350'000040$ ATMSO: TOW3 ; ADDR OF FIRST SPEED TM OUTPUT
00351'000017 FOTCD: 15. ; FIRST SPEED OUTPUT CODE
00352'000454' ASPDC: SPCOD-1 ; ADDR OF SPEED CODE TABLE-1
00353'100000 Z0VBT: 1B0 ; Z+0 VOLUME BIT TABLE BIT 0
00354'000200 1B1 ; 1
00355'040000 1B2 ; 2
00356'020000 1B3 ; 3
00357'000100 1B4 ; 4
00360'010000 1B5 ; 5
00361'004000 1B6 ; 6
00362'000040 1B7 ; 7
00363'002000 1B8 ; 8
00364'001000 1B9 ; 9
00365'000020 1B10 ; 10
00366'000400 1B11 ; 11
00367'000100 1B12 ; 12 HASKELL
00370'000020 1B13 ; 13 HASKELL
00371'000010 Z1VBT: 1B14 ; Z+1 VOLUME BIT TABLE BIT 0
00372'000000 0 ; 1
00373'000000 0 ; 2
00374'000004 1B15 ; 3
00375'000000 0 ; 4
00376'000040 Z1VB5: 1B16 ; 5 WLNT. (MOCK, FORS=1B11) *
00377'000002 1B17 ; 6
00400'000000 0 ; 7
00401'000000 0 ; 8
00402'000001 1B18 ; 9
00403'000000 0 ; 10
00404'100001 1B19+1B15; 11 FORS, WLNT, ROYL, MOCK.
00405'000200 1B20 ; 12 FORS
00406'020000 Z1OBT: 1B21 ; Z+1 OCCUP BIT TABLE BIT 0
00407'000000 0 ; 1
00410'000000 0 ; 2
00411'100000 1B22 ; 3
00412'000000 0 ; 4
00413'000400 1B23 ; 5 FORS, WLNT, MOCK
00414'001000 1B24 ; 6
00415'000000 0 ; 7
00416'000000 0 ; 8
00417'004000 1B25 ; 9
00420'000000 0 ; 10
00421'010000 1B26 ; 11 FORS, WLNT, ROYL, MOCK
00422'040000 1B27 ; 12 FORS
00423'000000 Z2VBT: 0 ; Z+2 VOL BIT TABLE BIT 0
00424'001001 1B28+1B15; 1
00425'000000 0 ; 2
00426'002001 1B29+1B15; 3
00427'000401 1B30+1B15; 4
00430'000000 0 ; 5
00431'000000 0 ; 6
00432'010001 1B31+1B15; 7
00433'000000 0 ; 8
00434'020001 1B32+1B15; 9
00435'004001 1B33+1B15; 10
00436'000045' AVTRA: AVTRP+1 ; AVER TRAP POINTER BIT 0
00437'000045' AVTRP+1 ; 1
00440'000045' AVTRP+1 ; 2

```

```

00441'000044'
00442'000044'
00443'000044'
00444'000047'
00445'000047'
00446'000047'
00447'000046'
00450'000046'
00451'000046'
00452'0000454'
00453'0000454'
00454'000000
00455'0000631
00456'0000315
00457'0000210
00460'0000146
00461'0000122
00462'0000104
00463'0000072
00464'0000063
00465'0000055
00466'0000051
00467'0000045
00470'0000042
00471'0000037
00472'0000035
00473'0000033

```

```

DUMMY:
SPCOD:

```

```

;
;
;
;

```

```

AVTRP ; 3
AVTRP ; 4
AVTRP ; 5
AVTRP+3 ; 6
AVTRP+3 ; 7
AVTRP+3 ; 8
AVTRP+2 ; 9
AVTRP+2 ; 10
AVTRP+2 ; 11
DUMMY ; 12
DUMMY ; 13
0 ; DUMMY AVER SPEED
409. ; SPEED OUTPUT CODE 15 SPEED<3.8
205. ; 14, <6.0
136. ; 13, <9.0
102. ; 12, <12.0
82. ; 11, <15.0
68. ; 10, <18.0
58. ; 9, <21.1
51. ; 8, <24.0
45. ; 7, <27.2
41. ; 6, <29.9
37. ; 5, <33.1
34. ; 4, <36.1
31. ; 3, <39.5
29. ; 2, <42.2
27. ; 1, <45.3
; 0, >45.3

```

```

.END ; END TASK 1

```

A0SAV	000015\$X
A1SAV	000016\$X
A2SAV	000017\$X
A3SAV	000020\$X
AAVRT	000347'
AAVTR	000204'
AINC1	000003\$X
AINC2	000004\$X
AINC3	000005\$X
AINC4	000006\$X
AINC5	000007\$X
AMB	000041
AMTM	177777 X
AOVOL	000320'
APDMT	000171\$X
ASCLP	000245'
ASPCD	000352'
ASTA	000153\$X
ASTLF	000144\$X
ASTWR	000176\$X
ASVLP	000055'
ATINI	000043'
ATMSO	000350'
AVTLP	000240'
AVTRA	000436'
AVTRP	000044'
AZ0VB	000201'
AZ10B	000051'
RIT	000052\$X
RITN	000151\$X
ROFLG	000161\$X
C1	000060\$X
C16	000054\$X
C2	000057\$X
C32	000053\$X
C4	000056\$X
C7	000167\$X
C8	000055\$X
CDABT	000101\$X
CNFLS	000166\$X
CROTB	000066\$X
CYCLE	000012\$X
D1T01	000070'
DATAM	000116\$X
DEBUG	000000
DETPR	000000'
DNO11	000107'
DTLOP	000056'
DUMMY	000454'
EA0SA	000030\$X
EA1SA	000031\$X
EA2SA	000032\$X
EA3SA	000033\$X
ECOMC	000165\$X
ECYCL	000025\$X
EMMOD	000164\$X
EPCSA	000027\$X
ESPCE	000024\$X
ESTAT	000026\$X
ETETI	000035\$X

```

---
ETIME 000034$X
FLTIM 000145$X
FORMM 000121$X
FOTCD 000351'
FOUNS 000252'
GRN 000042
GSTA 000152$X
HITSK 000004
IABTB 000064$X
IAFLB 000071$X
IFBIT 000063$X
IMASK 000142$X
IONLB 000070$X
IRSYR 000065$X
IVARA 000206'
IVART 000205'
LAKC0 000046$X
LAKC1 000047$X
LBYTE 000122$X
LOOPS 000160$X
LOW4M 000120$X
LRJC0 000050$X
LRJC1 000051$X
MAMTM 000170$X
MAXTP 000203'
MHTSK 000113$X
MINTP 000202'
MNOCU 000042'
MOCUP 000041'
MXLAM 000207'
NSPBT 000016
OAFLE 000074$X
OASTB 000076$X
ORQFB 000077$X
OCAKB 000072$X
OCCUP 000023'
OCDAB 000100$X
QCINI 000040'
OEMRR 000075$X
OOCUP 000054'
ORSYR 000067$X
OVOL 000052'
OVRDM 000117$X
PCINT 000001$X
PCSAV 000014$X
PDMNT 177777 X
PDMT1 177777 X
PDMT2 177777 X
PDMT4 177777 X
PDMT5 177777 X
PDMT6 177777 X
PEOTB 000062$X
PRIER 000174$X
PREP1 000156$X
PREP2 000157$X
PTDRT 000346'
PTFMN 000337'
PTLCN 000336'
PTMC 000345'
PTMDT 000321'

```

 PTMLP 000330'
 PTMR 000011\$X
 RED 000040
 RRFLG 000177\$X
 RROTE 000061\$X
 RSFLG 000146\$X
 RSTA 000154\$X
 RSWRD 000173\$X
 SCEDF 000115\$X
 SCERB 000073\$X
 SCLOP 000122'
 SGSTA 000162\$X
 SKSPD 000175'
 SNOMO 000210'
 SOFMN 000255'
 SPCEL 000002\$X
 SPCOD 000455'
 SPTAR 000023\$X
 SSAV0 000317'
 SSDNB 000102\$X
 SSRET 000316'
 STATE 000013\$X
 STRTS 000123\$X
 STL 000057
 STMR 000010\$X
 STWRD 000175\$X
 SVLLP 000275'
 SVOLB 000264'
 SVOUT 000314'
 TASK1 000000'
 TEM11 000177'
 TEM12 000200'
 TETIM 000022\$X
 TIBUF 000103\$X
 TIFIR 000104\$X
 TIIN 000105\$X
 TIMER 000021\$X
 TIN 000027
 TIOUT 000106\$X
 TMERF 000163\$X
 TMINT 000150\$X
 TMOUT 000147\$X
 TOBUF 000107\$X
 TOFIR 000110\$X
 TOIN 000111\$X
 TOOUT 000112\$X
 TOUT 000043
 TOUTW 000172\$X
 TOW1 000036\$X
 TOW13 000043\$X
 TOW16 000044\$X
 TOW17 000045\$X
 TOW2 000037\$X
 TOW3 000040\$X
 TOW7 000041\$X
 TOW9 000042\$X
 TPTOX 000153'
 UMASK 000143\$X
 VOLUM 000155\$X
 VOUT 000040

XEOPR 000114SX
EQVBT 000353'
E1DBG 000000U
E1OBT 000406'
E1VB5 000376'
E1VBT 000371'
E2DBG 000000U
E2VBT 000423'
EABRT 000136SX
EALS 000134SX
EBITN 000135SX
ECLER 000140SX
EGET 000125SX
EIJMP 000133SX
EIN 000020
EOR 000126SX
EP10M 000050'
EPTSB 000141SX
EPUT 000124SX
ERSTR 000132SX
ESMUL 000137SX
ESTFL 000130SX
ESTPR 000127SX
ZWAIT 000131SX

PROGRAM SEGMENT 5

```

---
      .TITL   SEG5
;
; TASK 2 - PROCESS SIGNAL LIGHTS
;
      .ENT    TASK2, ATABL, ATBL5
000000 TASK2:
000000' 020410 LITES: LDA    0, RFLTM ; RESTART FLASH TIME
000001' 006130$ JSR    @ZSTFL ; GO SET-UP FLASH MODE
000002' 006140$ JSR    @ZCLER ; CLEAR TASK 2 VARIABLES
000003' 000012' T2CLS ; CLEAR LIST
000004' 126520      .GPI  1, 1
000005' 046471 STA    1, @AAOUT; SET FLASH OUTPUTS
000006' 046403 STA    1, @AERTI ; SET TLM INPUT ERR TIMER
;
000007' 000426      JMP    LITLP ; BEGIN TASK 2
000100' 000036 RFLTM:  30. ; RESTART FLASH TIME (.5 SEC) *
000111' 000327' AERTI:  ERTIM ;
; LIST OF VARIABLES TO BE CLEARED
000120' 000160$ T2CLS:  LOOPS ; LOOP INPUTS FOR EMER CTL
000130' 000535'      GOUTP ; GRN OUTPUT
000140' 000534'      ROUTP ; RED OUTPUT
000150' 000046$      LAKC0 ; LAST INPUT CODE 0
000160' 000047$      LAKC1 ; LAST INPUT CODE 1
000170' 000164$      EMMOD ; EMER MODE
000200' 000050$      LRJC0 ; LAST CODE 0 REJECT
000210' 000325'      COMCD ; COMMAND WORD
000220' 000051$      LRJC1 ; LAST CODE 1 REJECT
000230' 000667'      CNFER ; SIG CNF ERR
000240' 000670'      LPRIO ; LAST OUTPUT PRIORITY WORD
000250' 000166$      CNFLS ; CONFIRM ERROR FLASH FLAG
000260' 000200$      PRER  ; RED ERROR BITS
000270' 000201$      PRAER ; AMBER ERROR BITS
000300' 000202$      PRGER ; GREEN ERROR BITS
000310' 000072'      LSRER ; LAST RED ERRORS
000320' 000101'      LSAER ; LAST AMBER ERRORS
000330' 000063'      LSGER ; LAST GREEN ERRORS
;
000340' 000000      0 ; END OF LIST
;
; TASK 2 - PROCESS SIGNAL LIGHTS
;
;
;
; PACK LOOP DETECTOR INPUT BITS INTO PROPER POSITIONS
; FOR EMERGENCY CONTROL PROCESSING
000350' 024156$ LITLP: LDA    1, PRZP1 ; PREVIOUS Z+1 INPUT
000360' 020160$ LDA    0, LOOPS ; UPDATE EMER CTL LOOP
000370' 006126$ JSR    @ZOR  ; INPUT WORD BY ORING
000400' 044160$ STA    1, LOOPS ; IN THE NEW BITS.
;
;
; PROCESS SIGNAL CONFIRMS
      .IFN    DEBUG ;
LDA    1, @AGOUT; IN DEBUG MODE PERFORM
LDA    0, DEGAN ; AND, OR WITH DEBUG WORDS
AND    0, 1 ;
LDA    0, DBGOR ;
JSR    @ZOR  ;
MOVS   1, 1 ; SWAPPED DEBUG CONFIRM
JMP    .+4 ; SKIP PAST INPUT

```

```

---
DBGAN:          177777 ; DEBUG GREEN AND WORD
DBGOR:          0      ; DEBUG GREEN OR WORD
                .ENDC
00041'064426   DIA    1,ZIN+6 ; READ GREEN CONFIRMS
00042'044505   STA    1,GCIN ;
                .IFN   DEBUG ;
LDA            1,@AROUT; PREVIOUS RED OUTPUT
LDA            0,DBRAN ; IN DEBUG MODE PERFORM
AND            0,1    ; AND,OR WITH DEBUG WORDS
LDA            0,DBROR ;
JSR            @ZOR   ;
MOVS           1,0    ; SWAPPED DEBUG CONFIRM
JMP            .+4   ; SKIP INPUT.
DBRAN:         177777 ; DEBUG RED AND WORD
DBROR:         0      ; DEBUG RED OR WORD
                .ENDC
00043'060424   DIA    0,ZIN+4 ; READ RED CONFIRMS
00044'101300   MOVS   0,0    ; SWAPPED RED CONFIRMS
00045'040504   STA    0,RCIN ; STORE RED CONFIRMS
                .IFN   DEBUG ;
LDA            1,@AAOUT; IN DEBUG MODE PERFORM
LDA            0,DBAAN ; AND,OR WITH DEBUG WORDS
AND            0,1    ;
LDA            0,DBAOR ;
JSR            @ZOR   ;
MOVS           1,1    ; SWAPPED DEBUG CONFIRM
JMP            .+4   ; SKIP PAST INPUT
DBAAN:         177777 ; DEBUG AMBER AND WORD
DBAOR:         0      ; DEBUG AMBER OR WORD
                .ENDC
00046'064425   DIA    1,ZIN+5 ; READ AMBER CONFIRMS
00047'131300   MOVS   1,2    ;
00050'050500   STA    2,ACIN ;
; PROCESS INPUT Z+3
                .IFN   DEBUG ; *****
LDA            1,Z3DBG ; IN DEBUG MODE, USE
JMP            .+3   ; DEBUG Z+3 INPUT
Z3DBG:         0      ;
                .ENDC
00051'064423   DIA    1,ZIN+3 ; INPUT Z+3 TO ACI
00052'044162$  STA    1,SGSTA ; STORE IN SIG STATUS WORD
00053'030431   LDA    2,SDSFB ; SIG SHUTDOWN + FLASH BITS
00054'133404   AND    1,2,SZR ; TEST EITHER CONDITION
00055'000451   JMP    STUPF  ; YES, PROCESS FLASH
; TEST FOR CONFIRM ERRORS
00056'024471   LDA    1,GCIN ; GRN CNF
00057'004426   JSR    CNFCK  ; PERFORM CHECK
00060'000535' AGOUT:  GOUTP  ; PREV GRN OUTPUT
00061'000206$  GLMSK  ; GRN MASK
00062'000202$  PRGER  ; GRN ERRORS
00063'000000   LSGER:  0      ; LAST GRN ERRORS
00064'026417   LDA    1,@AROUT; PREV RED OUTPUT
00065'125300   MOVS   1,1    ; POSITION
00066'004417   JSR    CNFCK  ; PERFORM CHECK
00067'000151'   RCIN   ; RED CNF WORD
00070'000204$  RLMSK  ; RED MASK
00071'000200$  PRRER  ; RED ERRORS
00072'000000   LSRER:  0      ; LAST RED ERRORS
00073'024455   LDA    1,ACIN ; AMBER CNF
00074'125300   MOVS   1,1    ; POSITION

```

```

---
00075'004410 JSR CNFCK ; PERFORM CHECK
00076'000533' AAOUT: AOUTP ; PREVIOUS AMBER OUTPUT
00077'000205$ ALMSK ; AMB MASK
00100'000201$ PRAER ; AMB ERRORS
00101'000000 LSAER: 0 ; LAST AMB ERRORS
00102'000450 JMP DLTIM ; GO TO LITE PROCESSING
00103'000534' AROUT: ROUTP ;
00104'030000 SDSFB: 1B2+1B3 ; SIG SHUTDWN + SIG FLASH
;
;
;
; SUBR TO HANDLE SIGNAL LIGHT CONFIRM ERROR CHECKING
; JSR CNFCK ; CALL SUBR
; ARG0 ; FIRST COMPARE VALUE ADDR. OTHER
; ARG1 ; ADDR OF MASK OF VALID BITS.
; ARG2 ; ADDR TO STORE SERIOUS ERRS
; ARG3 ; LAST ERROR BITS
; --- ---- ; NORMAL RETURN. IF FATAL ERROR
; THE HALVES OF AC1 WILL BE SWAPPED AND BITS
; WILL BE COMPARED. ANY BITS ON IN AC1 THAT ARE 0
; IN ARG0 CAUSE FATAL ERROR. ANY BIT OFF IN AC1 TH
; ON IN ARG0 CAUSE CONFIRM ERROR FLAG TO BE SET
; ONLY BITS SET IN ARG1 WILL BE CHECKED.
;
00105'023400 CNFCK: LDA 0,00,3 ; FIRST COMPARE VALUE = X
00106'033401 LDA 2,01,3 ; MASK TO AC2
00107'124300 COMS 1,1 ; POSITION AND COMP. SECOND VAL
00110'143400 AND 2,0 ; KEEP VALID BITS
00111'123414 AND# 1,0,SZR ; X.AND.(.NOT.Y)
00112'012417 ISZ @ACNFR ; INCR ERROR FLAG.
00113'100000 COM 0,0 ; .NOT. X
00114'124000 COM 1,1 ; Y
00115'143400 AND 2,0 ; KEEP VALID BITS
00116'123400 AND 1,0 ; .NOT.X .AND.Y
00117'025403 LDA 1,3,3 ; PREV ERR BITS
00120'041403 STA 0,3,3 ; UPDATE PREV.
00121'123405 AND 1,0,SNR ; TEST CONSEC ERRS
00122'001404 JMP 4,3 ; NO, RETURN.
00123'043402 STA 0,02,3 ; YES, SAVE SERIOUS ERRS.
00124'012405 ISZ @ACNFR ; INCR ERR FLAG
00125'010166$ ISZ CNFLS ; SET CONFIRM FLASH FLAG
00126'102460 STUPF: .ZAC 0,0 ; AC0 = MINIMUM FLASH TIME
00127'006130$ JSR @ZSTFL ; GO SET-UP FLASH MODE
00130'000455 JMP EDECR ; SKIP PAST DECR LOGIC
00131'000667' ACNFR: CNFER ; CONFIRM ERROR FLAG ADDR
; SUBR TO SET A BIT IN ONE WORD AND CLEAR SAME BIT IN AN
; JSR STRSB ; CALL WITH BIT # IN AC2
; ADDR1 ; ADDR OF SET WORD
; ADDR2 ; ADDR OF CLEAR WORD
; COMPLEMENT OF BIT WILL BE RETURNED IN AC1.
00132'050414 STRSB: STA 2,SVA2 ; SAVE BIT #
00133'025052$ LDA 1,BIT,2 ; BIT TO AC1
00134'124000 COM 1,1 ; .NOT. BIT
00135'033401 LDA 2,01,3 ; CLEAR WORD.
00136'133400 AND 1,2 ; CLEAR BIT.
00137'053401 STA 2,01,3 ; RESTORE CLEAR WORD.
00140'033400 LDA 2,00,3 ; SET WORD.
00141'133400 AND 1,2 ; OR BIT
00142'132000 ADC 1,2 ;
00143'053400 STA 2,00,3 ; RESTORE SET WORD

```

```

---
00144'030402      LDA      2,SVA2  ;
00145'001402      JMP      2,3    ;
00146'000000      SVA2:    0      ;
                   ; SOME MISC. VARIABLES
00147'000000      GCIN:    0      ; GRN CNF INPUT
00150'000000      ACIN:    0      ; SWAPPED AMB CNF INPUT
00151'000000      RCIN:    0      ; SWAPPED RED CNF INPUT
                   ;

```

```

; DECREMENT LITE TIMERS AND PROCESS AMBER TIME-OUT
;
00152'020153$ DLTIM: LDA 0,ASTA ; AMBER OUTPUT WORD
00153'101202 MOVR 0,0,SZC ; TEST BIT 15 (FLASH)
00154'000431 JMP EDECR ; YES, SKIP DECR LOGIC
00155'020152$ LDA 0,GSTA ; GREEN LIGHT STATUS
00156'152000 .GMI 2,2 ; INIT BIT #
00157'006135$ GTLOP: JSR @ZBITN ; GET BIT # OF BIT SET IN GSTA
00160'000413 JMP TAMBR ; NO MORE BITS SET.
00161'015010$ DSZ STMR,2 ; DECR STMR(N)
00162'000775 JMP GTLOP ; LOOP FOR ALL GSTA BITS.
00163'011010$ ISZ STMR,2 ; DONT ALLOW NEGATIVE STMR
00164'034055$ LDA 3,C8 ; IF STMR(N) = 0, TEST FOR PED
00165'156432 .SGT 2,3 ; LIGHT.
00166'000771 JMP GTLOP ; NO.
00167'004743 JSR STRSB ; YES. SET RED BIT, CLEAR GREEN
00170'000154$ RSTA ; BIT FOR PED LIGHTS.
00171'000152$ GSTA ;
00172'000765 JMP GTLOP ; LOOP FOR ALL GSTA BITS
00173'020153$ TAMBR: LDA 0,ASTA ; AMBER LIGHT STATUS
00174'152000 .GMI 2,2 ; INIT BIT #
00175'006135$ ATLOP: JSR @ZBITN ; GET NEXT BIT # SET IN ASTA
00176'000407 JMP EDECR ; NO MORE.
00177'015010$ DSZ STMR,2 ; DECR STMR(N)
00200'000775 JMP ATLOP ; NOT EXPIRED
00201'004731 JSR STRSB ; EXPIRED.
00202'000154$ RSTA ; SET RED BIT.
00203'000153$ ASTA ; CLEAR AMBER BIT.
00204'000771 JMP ATLOP ; LOOP FOR ALL AMBER BITS.
000000 EDECR: ; END DECR SECTION.
00205'014522 DSZ ERTIM ; DECR TM INPUT ERROR TIMER
00206'000404 JMP ETMOK ; OK IF NOT ZERO
00207'176520 .GPI 3,3 ; IF 0, SET
00210'054517 STA 3,ERTIM ; TIMER TO EXPIRE NEXT FRAME
00211'054163$ STA 3,TMERF ; SET TM INPUT ERROR FLAG
000000 ETMOK:
00212'004520 JSR ENCOD ; ENCODE B-SIDE OF
00213'001030' BMASK ; GREEN CONFIRM
00214'044506 STA 1,BECOD ; SAVE B-CODE
00215'004515 JSR ENCOD ; ENCODE A SIDE OF
00216'001015' AMASK ; GREEN CONFIRM
00217'030167$ LDA 2,C7 ;
00220'020502 LDA 0,BECOD ; B-CODE TO AC0
00221'132432 .SGT 1,2 ; TEST A-CODE=8
00222'000403 JMP .+3 ; NO
00223'050477 STA 2,BECOD ; YES, SET B-CODE=7
00224'105000 MOV 0,1 ; SET A-CODE=B-CODE
00225'044474 STA 1,AECOD ; STORE A-CODE
; READ AND PROCESS TM INPUT
00226'102460 .ZAC 0,0 ; CLEAR
00227'040501 STA 0,INPFL ; TM INPUT FLAG
00230'040501 STA 0,CMRJF ; AND COM REJ FLAG
00231'006125$ JSR @ZGET ; GET WORD FROM
00232'000104$ TIFIR ; TM INPUT BUFFER
00233'000534 JMP NOINP ; BUFFER EMPTY
00234'010474 TSINP: ISZ INPFL ; SET INPUT FLAG
00235'040531 STA 0,SAVC1 ; SAVE INPUT CODE
00236'034063$ LDA 3,IFBIT ; TM INPUT FORMAT BIT
00237'117404 AND 0,3,SZR ; TEST FORMAT 1

```

```

00240'000417      JMP      CODE1      ; YES
00241'024071$    LDA      1,IAFLB  ; ACT FLS INPUT BIT
00242'107415     AND#    0,1,SNR   ; TEST FLASH REQUEST
00243'000403     JMP      TSONL    ; NO
00244'102460     .ZAC    0,0      ; YES, SET MIN FLASH TIME
00245'006130$    JSR      @ZSTFL   ; SET FLASH MODE
00246'024070$    TSONL:  LDA      1,IONLB  ; SEE IF
00247'020517     LDA      0,SAVC1  ; ONLINE
00250'107415     AND#    0,1,SNR   ; REQUEST
00251'000445     JMP      NOTOL    ; NO.
00252'034063$    LDA      3,IFBIT  ; YES. TEST LAST
00253'024445     LDA      1,LCOMC  ; COMMAND TYPE.
00254'167414     AND#    3,1,SZR   ; IF TYPE 1 RESTE
00255'000436     JMP      TCCOM    ; ERR TIMER.
00256'000440     JMP      NOTOL    ; IF NOT, LEAVE ALONE.
00257'024153$    CODE1:  LDA      1,ASTA  ; TEST FLASH
00260'125202     MOVR    1,1,SZC   ; MODE
00261'010450     ISZ    CMRJF    ; REJECT
00262'024164$    LDA      1,EMMOD  ; TEST EMER CTL
00263'125004     .SZR   1,1      ; MODE IN PROGRESS.
00264'010445     ISZ    CMRJF    ; REJECT
; CONVERT CODE TO INTERNAL REPRESENTATION
00265'024436     LDA      1,CODMS  ; CODE MASK
00266'131120     MOVZL  1,2      ; SHIFT MASK LEFT 1 IN AC2
00267'107400     AND    0,1      ; B CODE TO AC1
00270'101200     MOVR    0,0      ; SHIFT A CODE RIGHT
00271'101200     MOVR    0,0      ; 2 PLACES
00272'143620     ANDZR  2,0      ; MASK AND SHIFT A CODE RIGHT 1.
00273'030167$    LDA      2,C7    ; TEST B CODE =7
00274'132414     .SEQ   1,2      ;
00275'000403     JMP      CODOK    ; NO.
00276'105000     MOV     0,1      ; YES, MOVE A CODE TO B
00277'141400     INC    2,0      ; AND SET A CODE =8
00300'004455     CODOK:  JSR      DECOD  ; DECODE B SIDE
00301'001031'    BTABL  ; B DECODE TABLE
00302'044422     STA      1,TECOD  ; SAVE B COMMAND
00303'105000     MOV     0,1      ; A CODE TO AC1
00304'004451     JSR      DECOD  ; DECODE A SIDE
00305'001016'    ATABL  ; A DECODE TABLE
00306'020416     LDA      0,TECOD  ; B COMMAND
00307'123000     ADD    1,0      ; ADD A COMMAND
00310'040415     STA      0,COMCD  ; STORE FULL COMMAND
00311'020455     LDA      0,SAVC1  ; INPUT WORD
00312'024406     LDA      1,LCOMC  ; RESET INPUT ERROR
00313'030413     TCCOM:  LDA      2,ICERT  ; TIMER IF NEW SIGNAL
00314'106414     .SEQ   0,1      ; COMD OR ONLINE
00315'050412     STA      2,ERTIM  ; COMD RECEIVED IN
00316'040402     NOTOL:  STA      0,LCOMC  ; TIMEOUT RANGE(ICERT).
00317'000450     JMP      NOINP   ; CONTINUE
;
;
00320'000000     LCOMC:  0        ; LAST COMMAND RECEIVED
00321'000000     AECOD:  0        ; A-CODE
00322'000000     BECOD:  0        ; B-CODE
00323'000007     CODMS:  7        ; CODE MASK
00324'000000     TECOD:  0        ; TEMP CODE SAVE
00325'000000     COMCD:  0        ; COMMAND CODE
00326'001130     ICERT:  600.    ; INPUT COMD ERR TIME (.1 SEC) *
00327'000000     ERTIM:  0        ; INPUT COMMAND ERR TIMER
00330'000000     INPFL:  0        ; TM INPUT FLAG

```

```

---
00331'000000 CMRJF:      0      ; COM REJ FLAG
;
; ENCOD GREEN OUTPUT STATUS FOR OUTPUT TO TELEMETRY
; JSR ENCOD ; CALL SUBR. GREEN CNF IN GOUTP
; XMASK ; ADDR OF MASK PRECEEDING CODE E
; RETURNS CODE IN AC1
00332'031400 ENCOD: LDA 2,0,3 ; ADDR OF MASK TO AC2
00333'021001 LDA 0,1,2 ; TABLE COUNT TO AC0
00334'040420 STA 0,ECNT ; SAVE # OF ENTRIES
00335'021000 LDA 0,0,2 ; MASK TO AC0
00336'026415 LDA 1,BGOUT ; GRN OUTPUT
00337'123400 AND 1,0 ; MASK BITS FOR EACH SIDE
00340'151400 INC 2,2 ; AC2 = ADDR-1 OF TABLE
00341'050005s STA 2,AINC3 ; STORE IN AUTO INCR LOCN
00342'126460 .ZAC 1,1 ; CLEAR CODE WORD
00343'032005s ENCLP: LDA 2,@AINC3; GET VALID COMMAND BITS
00344'112415 .SNE 0,2 ; TEST VALID GRN CNF
00345'001401 JMP 1,3 ; YES, RETURN WITH CODE IN AC1
00346'125400 INC 1,1 ; NO, INCR CODE
00347'014405 DSZ ECNT ; DECR ENTRY COUNT
00350'000773 JMP ENCLP ; LOOP.
00351'126460 .ZAC 1,1 ; IF NOT FOUND,
00352'001401 JMP 1,3 ; RETURN CODE 0
00353'000535' BGOUT: GOUTP ; ADDR OF GRN OUTPUT
00354'000000 ECNT: 0 ; ENCOD TEMP
;
; SUBR TO DECODE INTERSECTION TM SIGNAL COMMANDS
; JSR DECOD ; CALL SUBR WITH CODE IN AC1.
; ARG0 ; ADDR OF DECOD TABLE
; RETURNS DECODED VALUE IN AC1. IF INVALID INPUT, SETS E
; REJECT BIT AND TREATS AS NO INPUT.
00355'033400 DECOD: LDA 2,0,3 ; # OF VALID ENTRIES
00356'146432 .SGT 2,1 ; TEST FOR VALID CODE
00357'000405 JMP BADCM ; NO.
00360'031400 LDA 2,0,3 ; YES, AC2=TABLE ADDR
00361'133000 ADD 1,2 ; INDEX DOWN IN TABLE
00362'025001 LDA 1,1,2 ; LOAD DECODED VALUE
00363'001401 JMP 1,3 ; RETURN
00364'010745 BADCM: ISZ CMRJF ; SET COMMAND REJECT FLAG
00365'000402 JMP NOINP ; TREAT AS NO INPUT
00366'000000 SAVC1: 0

```

+++

```
00367'024153$ NOINP: LDA 1,ASTA ; TEST FLASH
00370'125202 MOVR 1,1,SZC ; MODE
00371'000463 JMP ENDIN ; YES
00372'024164$ LDA 1,EMMOD ; TEST EMER CTL
00373'125005 .SNZ 1,1 ; MODE
00374'000403 JMP NOEMM ; NO
00375'020165$ LDA 0,ECOMC ; MOVE EMER CTL COMMAND
00376'040727 STA 0,COMCD ; TO COMMAND WORD
00377'024153$ NOEMM: LDA 1,ASTA ; AMBER BITS
00400'020725 LDA 0,COMCD ; COMMAND CODE(DESIRED GREENS)
00401'107404 AND 0,1,SZR ; TEST DUPLICATE AMBERS AND
00402'000451 JMP CIGNR ; DESIRED GREENS. IGNORE IF SO
00403'024152$ LDA 1,GSTA ; CURRENT GREEN STATUS
00404'030524 LDA 2,LMASK ; SIGNAL LIGHT MASK
00405'147400 AND 2,1 ; KEEP SIGNAL BITS
00406'100000 COM 0,0 ; NOT(DESIRED GREENS)
00407'123405 AND 1,0,SNR ; GET ENDING GREENS
00410'000431 JMP GNGRN ; NONE.
00411'040520 STA 0,NAMS ; STORE AS NEW AMBERS
00412'152000 .GM1 2,2 ; INIT BIT #
00413'006135$ GELOP: JSR @ZBITN ; GET NEW AMBER BIT #
00414'000405 JMP ENAM1 ; NO MORE
00415'025010$ LDA 1,STMR,2; TEST MINIMUM TIME EXPIRED.
00416'125224 MOVZR 1,1,SZR ; =1 MEANS EXPIRED
00417'000434 JMP CIGNR ; NOT EXPIRED, IGNORE COMMAND.
00420'000773 JMP GELOP ; LOOP
00421'020510 ENAM1: LDA 0,NAMS ; NEW AMBERS
00422'152000 .GM1 2,2 ; INIT BIT #
00423'006135$ SATLP: JSR @ZBITN ; BIT # OF NEW AMBER
00424'000406 JMP ENAM2 ; NO MORE
00425'034170$ LDA 3,MAMTM ; ADDR-1 OF AMTM
00426'157000 ADD 2,3 ; AMTM(N)
00427'025401 LDA 1,1,3 ; GET AMBER TIME
00430'045010$ STA 1,STMR,2; STORE AMBER TIME IN STRM(N)
00431'000772 JMP SATLP ; LOOP
00432'020477 ENAM2: LDA 0,NAMS ; NEW AMBERS
00433'024153$ LDA 1,ASTA ; OR NEW WITH
00434'006126$ JSR @ZOR ; OLD AMBERS.
00435'044153$ STA 1,ASTA ; UPDATE AMBER STATUS
00436'024152$ LDA 1,GSTA ; REMOVE NEW AMBERS
00437'107400 AND 0,1 ; FROM GREEN STATUS
00440'044152$ STA 1,GSTA ; UPDATE GREEN STATUS
00441'020664 GNGRN: LDA 0,COMCD ; COMMAND WORD
00442'124000 COM 1,1 ; .NOT. GREEN STATUS
00443'107405 AND 0,1,SNR ; FORM NEW DESIRED GREENS
00444'000410 JMP ENDIN ; NONE
00445'044465 STA 1,NGRN ; NEW GREENS
00446'004472 JSR TVALD ; TEST A SIDE VALID,SET AND/OR B
00447'001015' AMASK ; PROPER STATUS BITS.
00450'004470 JSR TVALD ; SAME FOR B SIDE.
00451'001030' BMASK ;
00452'000402 JMP ENDIN ; CONTINUE
00453'010656 CIGNR: ISZ CMRJF ; REJECT COMMAND
00454'020154$ ENDIN: LDA 0,RSTA ; NOW MOVE RED,
00455'040457 STA 0,ROUTP ;
00456'020153$ LDA 0,ASTA ; AMBER,
00457'024162$ LDA 1,SGSTA ; TEST SIGNAL SHUTDOWN SWITCH
00460'030102$ LDA 2,SSDNB ; SET. IF SO, FLASH MODE HAS
00461'133404 AND 1,2,SZR ; BEEN SET UP(GSTA=RSTA=0). SO
```

```

---
00462' 102460      .ZAC      0,0      ; SET AOUTP=0.
00463' 040450      STA      0,AOUTP ;
00464' 020152s    LDA      0,GSTA  ; AND GREEN STATUS
00465' 040450      STA      0,GOUTP ; TO ACTUAL OUTPUT WORDS.
; PROCESS COMMAND ACK, COMMAND REJ
00466' 034642      LDA      3,INPFL ; TM INPUT FLAG
00467' 175005      .SNZ     3,3      ; TEST TM INPUT
00470' 000427      JMP      NXINP   ; NO, SKIP
00471' 034063s    LDA      3,IFBIT ; YES, GET TYPE
00472' 024674      LDA      1,SAVC1 ; OF INPUT
00473' 137404      AND      1,3,SZR ; IN AC3
00474' 176520      .GPI1   3,3      ; AC3=0 FOR CODE 0, 1 FOR CODE 1
00475' 020057s    LDA      0,C2    ; NOW DETERMINE IF COMMAND
00476' 030633      LDA      2,CMRJF ; WAS REJECTED. IF SO
00477' 151004      .SZR     2,2      ; ADD 2 TO CODE IN
00500' 117000      ADD      0,3     ; AC3
00501' 021446s    LDA      0,LAKC0,3 ; GET LAST ACK OR REJ
00502' 106415      .SNE     0,1     ; WORD AND COMPARE TO THIS
00503' 000414      JMP      NXINP   ; SAME, IGNORE
00504' 045446s    STA      1,LAKC0,3 ; UPDATE LAST VALUE
00505' 151004      .SZR     2,2      ; TEST FOR REJECT OF ACK
00506' 000405      JMP      COMRJ   ; REJECT
00507' 024072s    LDA      1,0CAKB ; PLACE COM ACK
00510' 004426      JSR      PTUN2   ; BIT IN TM OUTPUT
00511' 000043s    TOW13   ; BUFFER, FORMAT 13
00512' 000405      JMP      NXINP   ; CONTINUE
00513' 020066s    COMRJ:  LDA      0,CROTB ; SET COMMAND
00514' 060277      INTDS    ; REJECT BIT
00515' 006127s    JSR      0ZSTPR ; IN TELEMETRY PRIORITY
00516' 060177      INTEN    ; OUTPUT WORD
00517' 102460      NXINP:  .ZAC      0,0      ; CLEAR COM REJ
00520' 040611      STA      0,CMRJF ; FLAG
; CHECK FOR MORE TM INPUT
00521' 006125s    JSR      0ZGET   ; GET WORD FROM TM INPUT BUFFER
00522' 000104s    TIFIR    ; BUFFER POINTER
00523' 000531      JMP      SGOUT   ; BUFFER EMPTY
00524' 002401      JMP      0.+1   ; GOT WORD. PROCESS INPUT
00525' 000234s    TSINP    ;
; SOME TASK 2 PARAMETERS
000017 NVALG    =      15.      ; # OF VALID GREEN CONFIGS
00526' 177761      MNVLG:  -NVALG   ; - # OF VALID GREEN CONFIGS
00527' 001040s    AVALG:  VALGN-1 ; ADDR-1 OF VALID GREEN TABLE
00530' 177600      LMASK:  177600   ; SIGNAL LIGHT MASK
00531' 000000      NAMS:   0        ; NEW AMBERS
00532' 000000      NGRN:  0        ; NEW GREENS
00533' 000000      AOUTP:  0        ; SIGNAL OUTPUTS
00534' 000000      ROUTP:  0        ;
00535' 000000      GOUTP:  0        ;
;
; SUBR TO PLACE WORD UNCONDITIONALLY IN TM OUTPUT BUFFER
; JSR      PTUN2   ; CALL SUBR WITH OUTPUT BITS IN#
; ARG0     ; ADDR OF TM OUTPUT FORMAT WORD
; BUFFER OFLO BIT WILL BE SET IF BUFFER IS FULL
00536' 102460      PTUN2:  .ZAC      0,0      ; SET NO-COMPARE FLAG
00537' 000500      JMP      PTCM2+1 ; GO TO COMMON SUBR
;
; SUBR TO TEST EACH DESIRED NEW GREEN BIT AND INITIATE IT
; GREEN IF VALID. UPDATES GSTA AND RSTA. ALSO INITIATESB
; IF NECESSARY.
; JSR      TVALD   ; CALL SUBR. NEW GREENS IN NGRN

```

```

;
;
00540' 054467 TVALD: STA 3,TVRET ; SAVE RETURN
00541' 031400 LDA 2,0,3 ; ADDR OF MASK
00542' 020770 LDA 0,NGRN ; NEW GREENS
00543' 025000 LDA 1,0,2 ; MASK
00544' 123405 AND 1,0,SNR ; KEEP NEW GREENS FOR THIS SIDE.
00545' 001401 JMP 1,3 ; EXIT IF NONE.
00546' 044462 STA 1,TMASK ; SAVE MASK
00547' 151400 INC 2,2 ; ADDR OF TABLE-1
00550' 050461 STA 2,TBLAD ; SAVE TABLE START
00551' 025000 LDA 1,0,2 ; # OF TABLE ENTRIES
00552' 044460 STA 1,TBLNM ; SAVE FOR LATER.
00553' 152000 .GM1 2,2 ; STARTING BIT #
00554' 006135$ NVLBT: JSR 0ZBITN ; GET BIT # OF NEXT NEW GREEN.
00555' 000450 JMP EXTVL ; NO MORE.
00556' 024453 LDA 1,TBLAD ; MOVE TABLE ADDR-1
00557' 044005$ STA 1,AINC3 ; TO AUTO INCR LOC
00560' 024452 LDA 1,TBLNM ; MOVE # OF TABLE ENTRIES
00561' 044452 STA 1,DIND ; TO COUNTER
00562' 040452 STA 0,TVSV0 ; SAVE REMAINING NGRN BITS.
00563' 020153$ LDA 0,ASTA ; FORM OR(GSTA,ASTA,BIT(N))=
00564' 024152$ LDA 1,GSTA ; CURRENT GREENS+CURRENT AMBERS+
00565' 006126$ JSR 0ZOR ; NEW GREEN BIT.
00566' 021052$ LDA 0,BIT,2 ; GET BIT(N)
00567' 006126$ JSR 0ZOR ; RESULT IN AC1. .NOT. BIT IN AQ
00570' 034440 LDA 3,TMASK ; KEEP ONLY BITS
00571' 107400 AND 3,1 ; FOR THIS SIDE.
00572' 036035$ MOTAB: LDA 3,@AINC3; GET VALID TABLE ENTRY AND INCR
00573' 166415 .SNE 3,1 ; TEST VALID CONFIG
00574' 000404 JMP GOTVL ; YES.
00575' 014436 DSZ DIND ; DECR TABLE COUNT.
00576' 000774 JMP MOTAB ; LOOP FOR WHOLE TBL
00577' 000424 JMP TRNXB ; NOT VALID. TRY NEXT BIT.
00600' 114300 GOTVL: COMS 0,3 ; SWAP HALVES OF GREEN BIT COMPL
00601' 175220 MOVZR 3,3 ; POSITION GREEN BIT TO CORRESPD
;
; TO PED BIT AND COMPL.
00602' 024433 LDA 1,PEDBT ; PED BITS.
00603' 137400 AND 1,3 ; KEEP PED BIT IF CALL FOR
00604' 174000 COM 3,3 ; .NOT. PED BIT.
00605' 163400 AND 3,0 ; .NOT. PED BIT + GREEN BIT
00606' 024152$ LDA 1,GSTA ; CURRENT GREEN STATUS
00607' 107400 AND 0,1 ; OR NEW GREEN AND
00610' 106000 ADC 0,1 ; NEW PED BIT WITH GSTA
00611' 044152$ STA 1,GSTA ; UPDATE GSTA
00612' 024154$ LDA 1,RSTA ; RED STATUS
00613' 107400 AND 0,1 ; CLEAR NEW GREEN +
00614' 044154$ STA 1,RSTA ; NEW PED BITS FROM RSTA.
00615' 034171$ LDA 3,APDMT ; ADDR OF PED TIMES
00616' 157000 ADD 2,3 ;
00617' 021400 LDA 0,0,3 ; PDMN(N)
00620' 041011$ STA 0,PTMR,2; STORE PDMN(N) IN PED TIMER
00621' 101400 INC 0,0 ; STORE PED TIME + 1 AS
00622' 041010$ STA 0,STMR,2; MINIMUM GREEN TIME.
00623' 020411 TRNXB: LDA 0,TVSV0 ; RESTORE NEW GREEN BITS
00624' 000730 JMP NVLBT ; CHECK NEXT BIT.
00625' 034402 EXTVL: LDA 3,TVRET ; RETURN ADDR.
00626' 001401 JMP 1,3 ; RETURN TO CALL+2.
; TVALD PARAMETERS
00627' 000000 TVRET: 0 ; RETN ADDR

```

00630'000000	TMASK:	0	; VALID CONFIG MASK
00631'000000	TBLAD:	0	; VALID TABLE ADDR
00632'000000	TBLNM:	0	; # OF TABLE ENTRIES
00633'000000	DIND:	0	; TABLE COUNTER
00634'000000	TVSV0:	0	; SAVE AC0
00635'000167	PEDBT:	167	; PED BITS

ttt

```
;  
; SUBR TO PLACE WORD IN TM OUTPUT BUFFER IF CHANGED  
; JSR PTCM2 ; CALL SUBR WITH OUTPUT BITS IN  
; ARG0 ; ADDR OF TM OUTPUT FORMAT WORD  
00636'102000 PTCM2: .GM1 0,0 ; AC0=COMPARE FLAG  
00637'044414 STA 1,PT20T ; STORE OUTPUT BITS  
00640'025400 LDA 1,0,3 ; GET FORMAT ADDR  
00641'044406 STA 1,PT2AD ; AND STORE  
00642'175400 INC 3,3 ; SET UP RETURN  
00643'054407 STA 3,PT2RT ;  
00644'060277 INTD5 ; DISALLOW RUPT  
00645'0061415 JSR @ZPTSB ; CALL TM PUT SUBR  
00646'000653' PT20T ; DATA WORD  
00647'000000 PT2AD: 0 ; FORMAT ADDR  
00650'060177 INTEN ; ENABLE RUPT  
00651'002401 JMP @PT2RT ; RETURN  
00652'000000 PT2RT: 0 ; RETURN ADDR  
00653'000000 PT20T: 0 ; DATA BITS  
;  
; CHECK SIGNAL OUTPUTS FOR VALIDITY  
00654'020653 SGOUT: LDA 0,AVALG ; ADDR-1 OF VALID GREEN TABLE  
00655'040005$ STA 0,AINC3 ; STORE IN AUTO-INCR LOC  
00656'034650 LDA 3,MNVLG ; -# OF VALID GRNS  
00657'020656 LDA 0,GOUTP ; PROPOSED GRN OUTPUT  
00660'026005$ CKGOT: LDA 1,@AINC3; VALID GRN CONFIG  
00661'124000 COM 1,1 ; MAKE VALID BITS ZERO  
00662'107405 AND 0,1,SNR ; TEST FOR VALID GREENS  
00663'000406 JMP GROOK ; ALL GREENS OK  
00664'175404 INC 3,3,SZR ; COUNT # OF VALID GRNS  
00665'000773 JMP CKGOT ; LOOP FOR ALL TABLE ENTRIES  
00666'006136$ JSR @ZABRT ; ABORT IF NOT VALID GRNS  
00667'000000 CNFER: 0 ; CONFIRM ERROR FLAG  
00670'000000 LPRIO: 0 ; LAST PRIO OUTPUT  
00671'024153$ GROOK: LDA 1,ASTA ; AC1=PROPOSED AMBER OUTPUT  
00672'125212 MOVR# 1,1,SZC ; TEST FLASH BIT  
00673'000414 JMP DOOUT ; YES  
00674'030640 LDA 2,ROUTP ; AC2=PROPOSED RED OUTPUT  
00675'107415 AND# 0,1,SNR ; TEST FOR DUPL GRNS AND AMBS  
00676'113414 AND# 0,2,SZR ; TEST DUPL GRNS AND REDS  
00677'006136$ JSR @ZABRT ; YES,ABORT  
00700'133414 AND# 1,2,SZR ; TEST DUPL AMBERS AND REDS  
00701'006136$ JSR @ZABRT ; YES, ABORT  
00702'006126$ JSR @ZOR ; OR(GREEN,AMBER,RED)  
00703'141000 MOV 2,0 ;  
00704'006126$ JSR @ZOR ;  
00705'124014 .SM1 1,1 ; RESULT SHOULD BE ALL BITS  
00706'006136$ JSR @ZABRT ; ABORT IF NOT.  
00707'020625 DOOUT: LDA 0,ROUTP ; LOAD OUTPUTS  
00710'024623 LDA 1,AOUTP ;  
00711'030624 LDA 2,GOUTP ;  
000000 .IFE DEBUG ; *  
00712'061040 DOA 0,RED ; OUTPUT RED  
00713'065041 DOA 1,AMB ; OUTPUT AMB  
00714'071042 DOA 2,GRN ; OUTPUT GRN *  
 .ENDC ;  
;  
; FORMAT AND SEND STATUS BITS TO TM OUTPUT  
00715'026477 LDA 1,@AAECD; A-CODE  
00716'152460 .ZAC 2,2 ; AC2 = 0
```

```

---
00717'020750 LDA 0,CNFER ; CONFIRM ERROR FLAG
00720'050747 STA 2,CNFER ; CLEAR IT
00721'101004 .SZR 0,0 ; TEST NOT CNF ERR
00722'030073S LDA 2,SCERB ; SET CNF ERR BIT
00723'147000 ADD 2,1 ; FORM TM OUTPUT BITS
00724'004712 JSR PTCM2 ; GO STORE TM OUTPUT
00725'000036S TOW1 ; FORMAT 1.
00726'026465 LDA 1,@ABECD ; GET B-CODE
00727'020162S LDA 0,SGSTA ; SIGNAL STATUS WORD
00730'030101S LDA 2,CDABT ; CAB DOOR BIT
00731'113404 AND 0,2,SZR ; TEST CAB DOOR AJAR
00732'030100S LDA 2,OCDAB ; YES, GET OUTPUT BIT
00733'147000 ADD 2,1 ; FORM B-CODE + CAB DOOR STATUS
00734'004702 JSR PTCM2 ; STORE OUTPUT FORMAT 2
00735'000037S TOW2 ;
00736'126460 .ZAC 1,1 ; CLEAR AC1
00737'020164S LDA 0,EMMOD ; TEST EMER MODE.
00740'101004 .SZR 0,0 ;
00741'024075S LDA 1,0EMRB ; YES, SET EMER CTL OUTPUT BIT
00742'020162S LDA 0,SGSTA ; SIG STATUS(Z+3)
00743'030207S LDA 2,AXFLB ; AUX FLASH BIT.
00744'113404 AND 0,2,SZR ; TEST AUX FLASH.
00745'030074S LDA 2,OAFLB ; YES, GET OUTPUT BIT
00746'147000 ADD 2,1 ; YES, SET OUTPUT BIT
00747'004667 JSR PTCM2 ; GO STORE TM OUTPUT
00750'000172S TOUTW ; FORMAT 0
00751'024162S LDA 1,SGSTA ; GET SIG STATUS (Z+3)
00752'125200 MOVR 1,1 ; POSITION TEMP SW
00753'125200 MOVR 1,1 ; SIG SHUT DOWN SW
00754'125200 MOVR 1,1 ; FLASH SW
00755'125300 MOVS 1,1 ; AND MANUAL SW
00756'004660 JSR PTCM2 ; PUT OUTPUT FORMAT 16 WORD
00757'000044S TOW16 ; IN TM OUT BUFR
00760'024162S LDA 1,SGSTA ; RESTORE INPUT Z+3
00761'127120 ADDZL 1,1 ; SHIFT LEFT 3
00762'125100 MOVL 1,1 ; POSITION INTERCONNECT BITS
00763'125300 MOVS 1,1 ; MOVE 4 BITS TO LOWER BYTE
00764'004652 JSR PTCM2 ; PUT OUTPUT FORMAT 17 WORD
00765'000045S TOW17 ; IN TM OUTPUT BUFR
00766'102460 .ZAC 0,0 ; AC0 = 0
00767'060277 INTDS ; DISABLE RUPT
00770'034161S LDA 3,BOFLG ; GET BUFF OFLO FLAG
00771'040161S STA 0,BOFLG ; CLEAR IT
00772'060177 INTEN ; ALLOW RUPT
00773'175004 .SZR 3,3 ; TEST BUFF OFLO
00774'020077S LDA 0,0BOFB ; YES, SET BIT
00775'030076S LDA 2,OASTB ;
00776'024162S LDA 1,SGSTA ; RELOAD (Z+3)
00777'125112 MOVL# 1,1,SZC ; TEST ABORT/STALL
01000'143000 ADD 2,0 ; YES, SET BIT
01001'030667 LDA 2,LPRIO ; LAST PRIORITY OUTPUT
01002'112415 .SNE 0,2 ; COMPARE TO CURRENT
01003'000405 JMP SKPRO ; SAME, DONT SEND
01004'040664 STA 0,LPRIO ; UPDATE LAST
01005'060277 INTDS ; DISABLE RUPT
01006'006127S JSR @ZSTPR ; STORE PRIORITY OUTPUT
01007'060177 INTEN ; ENABLE RUPT
01010'006131S SKPRO: JSR @ZWAIT ; WAIT FOR NEXT TASK2 FRAME
01011'002401 JMP 0,+1 ; LOOP FOR NEXT FRAME
01012'000035' LITLP ; TASK 2 LOOP ADDR

```



```

---
01013'000322' ABECD:          BECOD   ; B-CODE
01014'000321' AAECD:          AECOD   ; A-CODE
;
;
;
;
; INTERSECTION COMMAND DECODE TABLES
01015'170200 AMASK:          17B3+1B8
01016'000007 ATABL:          7          ; # OF ENTRIES (9. FOR FOREST) *
01017'000000          0          ; CODE 0
01020'100000          1B0        ; 1
01021'040000          1B1        ; 2
01022'020000          1B2        ; 3
01023'010000          1B3        ; 4
01024'060000          1B1+1B2    ; 5
01025'030000          1B2+1B3    ; 6
01026'000200          1B8        ; 7
01027'040200          1B1+1B8    ; 8
          001024' ATBL5      =    ATABL+6 ; CODE 6
01030'007400 BMASK:          17B7
01031'000007 BTABL:          7          ; # OF ENTRIES
01032'000000          0          ; CODE 0
01033'004000          1B4        ; 1
01034'002000          1B5        ; 2
01035'001000          1B6        ; 3
01036'000400          1B7        ; 4
01037'003000          1B5+1B6    ; 5
01040'001400          1B6+1B7    ; 6
;
          000002          .RDX      2          ; USE BINARY FOR CLARITY
01041'104104 VALGN:          1000100001000100; VALID SIMULTANEOUS GREENS ENTE
01042'103103          1000011001000011; 1
01043'101501          1000001101000001; 2
01044'064064          0110100000110100; 3
01045'063063          0110011000110011; 4
01046'061461          0110001100110001; 5
01047'034024          0011100000010100; 6
01050'033023          0011011000010011; 7
01051'031421          0011001100010001; 8 END OF NORMAL VALID GREENS
01052'044244          0100100010100100; 9 REMAINDER ARE FOR FOREST LAN
01053'043243          0100011010100011; 10
01054'041641          0100001110100001; 11
01055'014204          0001100010000100; 12
01056'013203          0001011010000011; 13
01057'011601          0001001110000001; 14
          000010          .RDX      8          ; BACK TO OCTAL
;
          .END          ; END SEG 5

```

 A0SAV 000015SX
 A1SAV 000016SX
 A2SAV 000017SX
 A3SAV 000020SX
 AAECD 001014'
 AAOUT 000076'
 ABECD 001013'
 ACIN 000150'
 ACNFR 000131'
 AECOD 000321'
 AERTI 000011'
 AGOUT 000060'
 AINC1 000003SX
 AINC2 000004SX
 AINC3 000005SX
 AINC4 000006SX
 AINC5 000007SX
 ALMSK 000205SX
 AMASK 001015'
 AMB 000041'
 AMTM 177777 X
 AOUTP 000533'
 APDMT 000171SX
 AROUT 000103'
 ASTA 000153SX
 ASTLF 000144SX
 ASTWR 000176SX
 ATABL 001016'
 ATBL5 001024'
 ATLOP 000175'
 AVALG 000527'
 ANFLB 000207SX
 BADCM 000364'
 BECOD 000322'
 EGOUT 000353'
 BIT 000052SX
 BITN 000151SX
 BMASK 001030'
 BOFLG 000161SX
 BTABL 001031'
 C1 000060SX
 C16 000054SX
 C2 000057SX
 C32 000053SX
 C4 000056SX
 C7 000167SX
 C8 000055SX
 CDABT 000101SX
 CIGNR 000453'
 CKGOT 000660'
 CLOKF 000203SX
 CMRJF 000331'
 CNFCK 000105'
 CNFER 000667'
 CNFLS 000166SX
 CODE1 000257'
 CODMS 000323'
 CODOK 000300'
 COMCD 000325'
 COMRJ 000513'

CROTB 000066SX
CYCLE 000012SX
DATAM 000116SX
DBAAN 000000U
DBAOR 000000U
DEGAN 000000U
DEGOR 000000U
DBRAN 000000U
DBROR 000000U
DECOD 000355'
DIND 000633'
DLTIM 000152'
DOOUT 000707'
EA0SA 000030SX
EA1SA 000031SX
EA2SA 000032SX
EA3SA 000033SX
ECNT 000354'
ECONC 000165SX
ECYCL 000025SX
EDECR 000205'
EMMOD 000164SX
ENAM1 000421'
ENAM2 000432'
ENCLP 000343'
ENCOD 000332'
ENDIN 000454'
EPCSA 000027SX
ERTIM 000327'
ESPCE 000024SX
ESTAT 000026SX
ETETI 000035SX
ETIME 000034SX
ETMOK 000212'
EXTVL 000625'
FLTIM 000145SX
FORMM 000121SX
GCIN 000147'
GELOP 000413'
GLMSK 000206SX
GNGRN 000441'
GOTVL 000600'
GOUTP 000535'
GRN 000042'
GROOK 000671'
GSTA 000152SX
GTLOP 000157'
HITSK 000004'
IAETB 000064SX
IAFLB 000071SX
ICERT 000326'
IFBIT 000063SX
IMASK 000142SX
INPFL 000330'
INTCD 000210SX
IONLB 000070SX
IRSYB 000065SX
LAKC0 000046SX
LAKC1 000047SX
LBYTE 000122SX

LCOMC 000320'
LITES 000000'
LITLP 000035'
LMASK 000530'
LOOPS 000160\$X
LOW4M 000120\$X
LPRIO 000670'
LRJC0 000050\$X
LRJC1 000051\$X
LSAER 000101'
LSGER 000063'
LSRER 000072'
MAMTM 000170\$X
MHTSK 000113\$X
MVVLG 000526'
MOTAB 000572'
NAM5 000531'
NGRN 000532'
NOEMM 000377'
NOINP 000367'
NOTOL 000316'
NSPBT 000016'
NVALG 000017'
NVLBT 000554'
NXINP 000517'
OAFLB 000074\$X
OASTE 000076\$X
OBOFB 000077\$X
OCAKB 000072\$X
OCDAB 000100\$X
OEMRB 000075\$X
ORSYB 000067\$X
OWRDM 000117\$X
PCINT 000001\$X
PCSAV 000014\$X
PDMNT 177777 X
PDMT1 177777 X
PDMT2 177777 X
PDMT4 177777 X
PDMT5 177777 X
PDMT6 177777 X
PEDBT 000635'
PEOTB 000062\$X
PRAER 000201\$X
PRGER 000202\$X
PRIWR 000174\$X
PRRER 000200\$X
PRZP1 000156\$X
PRZP2 000157\$X
PT2AD 000647'
PT20T 000653'
PT2RT 000652'
PTCM2 000636'
PTMR 000011\$X
PTUN2 000536'
RCIN 000151'
RED 000040'
RFLTM 000010'
RLMSK 000204\$X
RQUTD 000500'

 RRFLG 000177\$X
 RROTB 000061\$X
 RSFLG 000146\$X
 RSTA 000154\$X
 RSWRD 000173\$X
 SATLP 000423'
 SAVC1 000366'
 SCEDF 000115\$X
 SCERB 000073\$X
 SDSFB 000104'
 SGOUT 000654'
 SGSTA 000162\$X
 SKPRO 001010'
 SPCEL 000002\$X
 SPTAB 000023\$X
 SSDNB 000102\$X
 STATE 000013\$X
 STETS 000123\$X
 STL 000057
 STMR 000010\$X
 STRSB 000132'
 STUPF 000126'
 STWRD 000175\$X
 SVA2 000146'
 T2CLS 000012'
 TAMBR 000173'
 TASK2 000000'
 TELAD 000631'
 TELNM 000632'
 TCCOM 000313'
 TECOD 000324'
 TETIM 000022\$X
 TIBUF 000103\$X
 TIFIR 000104\$X
 TIIN 000105\$X
 TIMER 000021\$X
 TIN 000027
 TIOUT 000106\$X
 TMASK 000630'
 TMERF 000163\$X
 TMINT 000150\$X
 TMOUT 000147\$X
 TOBUF 000107\$X
 TOFIR 000110\$X
 TOIN 000111\$X
 TOOUT 000112\$X
 TOUT 000043
 TOUTW 000172\$X
 TOW1 000036\$X
 TOW13 000043\$X
 TOW16 000044\$X
 TOW17 000045\$X
 TOW2 000037\$X
 TOW3 000040\$X
 TOW7 000041\$X
 TOW9 000042\$X
 TRNKB 000623'
 TSINP 000234'
 TSONL 000246'
 TVALD 000540'

TVRET 000627'
TVSVØ 000634'
UMASK 000143\$X
VALGN 001041'
VOLUM 000155\$X
VOUT 000040
XEQPR 000114\$X
Z3DEG 000000U
ZABRT 000136\$X
ZALS 000134\$X
ZDITN 000135\$X
ZCLER 000140\$X
ZGET 000125\$X
ZJMP 000133\$X
ZIN 000020
ZOR 000126\$X
ZPTSB 000141\$X
ZPUT 000124\$X
ZRSTR 000132\$X
ZSMUL 000137\$X
ZSTFL 000130\$X
ZSTPR 000127\$X
ZWAIT 000131\$X

PROGRAM SEGMENT 6

.TITL SEGM6

; TASK 3-EMERGENCY CONTROL
; INITIALIZATION

.FXTN ATABL,31VB5
.ENT TASK3

000000 TASK3:
00000'102460 EMERC: .ZAC 0,0 ; CLEAR
00001'040444 STA 0,PERFL ; PERM FLASH FLAG,
00002'024444 LDA 1,CNFTM ; INIT CONFIRM ERROR
00003'044441 STA 1,CNERT ; TIMER
00004'000471 JMP EMERL ; GO TO EMER LOOP

; SUBR TO PICK OFF LEFTMOST BITS OF AC0, PLACE THEM IN
; AC1 AND ADD A BIAS THEN DOUBLE THE QUAN.
; JSR PICKB ; CALL SUBR WITH NUM IN AC0
; -N ; -# OF BITS TO PICK
; BIAS ; BIAS VALUE

00005'050413 PICKB: STA 2,PSVA2 ; SAVE AC2
00006'031400 LDA 2,0,3 ; -N TO AC2
00007'126460 .ZAC 1,1 ; CLEAR AC1
00010'101120 MOVZL 0,0 ; LEFT BIT TO CRY
00011'125100 MOVL 1,1 ; CRY TO AC1,B15
00012'151404 INC 2,2,SZR ; COUNT BITS
00013'000775 JMP -3 ; LOOP
00014'031401 LDA 2,1,3 ; BIAS
00015'147120 ADDZL 2,1 ; ADD AND DOUBLE
00016'030402 LDA 2,PSVA2 ; RESTORE AC2
00017'001402 JMP 2,3 ; RETURN
00020'000000 PSVA2: 0 ; AC2 SAVE

; SUBR TO UNPACK PIN MATRIX WORDS. ENTER WITH WORD IN
; AC0. RETURNS WITH FINAL VALUE IN AC1

; JSR UNPKM ; CALL SUBR
; ADD1 ; ADDR TO STORE FIELD 1
; ADD2 ; ADDR TO STORE FIELD 2
; ADD3 ; ADDR TO STORE FIELD 3
; ADD4 ; ADDR TO STORE FIELD 4

00021'171000 UNPKM: MOV 3,2 ; CALL+1 TO AC2
00022'004763 JSR PICKB ; PICK OFF FIRST
00023'177772 -6 ; 6 BITS
00024'000001 1 ; 1 SEC BIAS
00025'047000 STA 1,00,2 ; STORE IN ADD1
00026'004757 JSR PICKB ; PICK OFF NEXT
00027'177774 -4 ; 4 BITS
00030'000001 1 ; 1 SEC BIAS
00031'047001 STA 1,01,2 ; STORE IN ADD2
00032'004753 JSR PICKB ; PICK OFF NEXT
00033'177775 -3 ; 3 BITS
00034'000001 1 ; 1 SEC BIAS
00035'047002 STA 1,02,2 ; STORE IN ADD3
00036'004747 JSR PICKB ; PICK OFF LAST
00037'177775 -3 ; 3 BITS
00040'000001 1 ; 1 SEC BIAS
00041'047003 STA 1,03,2 ; STORE IN ADD4
00042'001004 JMP 4,2 ; RETURN


```

---
00043'177770 M8: -8. ; CONSTANT -8
00044'000000 CNERT: 0 ; CNF ERR TIMER
00045'000000 PERFL: 0 ; PERM FLASH FLAG
00046'000170 CNFTM: 120. ; CNF ERR TIME PERIOD (.5 SEC)
00047'000006 MAXCF: 6 ; MAX # OF CNF ERRS ALLOWED

; PHASE TIMES
00050'000024 MAXT: 20. ; MAX PHASE 1
00051'000024 20. ; 2
00052'000024 20. ; 3
00053'000024 20. ; 4
00054'000024 20. ; 5
00055'000024 20. ; 6
00056'000036 30. ; 7
00057'000024 MINT: 20. ; MIN PHASE 1
00060'000024 20. ; 2
00061'000024 20. ; 3
00062'000024 20. ; 4
00063'000024 20. ; 5
00064'000024 20. ; 6
00065'000006 6 ; 7
00066'000001 EXTT: 1 ; EXT PHASE 1
00067'000001 1 ; 2
00070'000001 1 ; 3
00071'000001 1 ; 4
00072'000001 1 ; 5
00073'000001 1 ; 6
00074'000004 4 ; 7

```

```

;
; START TASK 3 LOOP
; PROCESS PIN MATRIX
000000 EMERL: ; EMER CTL LOOP
      .IFN DEBUG ; *****
      LDA 0,Z8DBG ; Z+8 DEBUG
      JMP .+3 ;
Z8DBG: 0 ;
      .ENDC ; *****
00075'060430 DIA 0,ZIN+8. ; READ FIRST PIN WORD
00076'101213 MOVR# 0,0,SNC ; TEST B15
00077'000447 JMP SKPMT ; 0=IGNORE MATRIX
00100'004721 JSR UNPKM ; UNPACK WORD 1
00101'000050' MAXT ; PHASE 1 TIMES
00102'000057' MINT ;
00103'000066' EXTT ;
00104'000052' MAXT+2 ; DUMMY
      .IFN DEBUG ; *****
      LDA 0,Z9DBG ; Z+9 DEBUG
      JMP .+3 ;
Z9DBG: 0 ;
      .ENDC ; *****
00105'060431 DIA 0,ZIN+9. ; READ PIN 2
00106'004713 JSR UNPKM ; UNPACK
00107'000052' MAXT+2 ; PHASE 3 TIMES
00110'000061' MINT+2 ;
00111'000070' EXTT+2 ;
00112'000051' MAXT+1 ; PHASE 2 TIMES
00113'044745 STA 1,MINT+1 ; SET MIN=MAX FOR PHASE 2
      .IFN DEBUG ; *****
      LDA 0,Z10DB ; Z+10. DEBUG
      JMP .+3 ;
Z10DB: 0 ;

```

```

      .ENDC ; *****
00114'060432 DIA 0,ZIN+10.; READ PIN 3
00115'004704 JSR UNPKM ; UNPACK
00116'000053' MAXT+3 ; PHASE 4 TIMES
00117'000062' MINT+3 ;
00120'000071' EXTT+3 ;
00121'000055' MAXT+5 ; DUMMY

      .IFN DEBUG ; *****
LDA 0,Z11DB ; Z+11. DEBUG
JMP .+3
Z11DB: 0 ;

      .ENDC ; *****
00122'060433 DIA 0,ZIN+11.; READ PIN 4
00123'004676 JSR UNPKM ; UNPACK
00124'000055' MAXT+5 ; PHASE 6 TIMES
00125'000064' MINT+5 ;
00126'000073' EXTT+5 ;
00127'000054' MAXT+4 ; PHASE 5.
00130'044733 STA 1,MINT+4; SET MIN=MAX FOR P5
00131'060477 READS 0 ; READ AMBERS
00132'024170$ LDA 1,MAMTM ; ADDR-1 OF AMBER TIMES
00133'044007$ STA 1,AINCS ; TO AUTO-INCR CELL.
00134'030707 LDA 2,M8 ; :8 = AMBER COUNT
00135'004650 AMLOP: JSR PICKB ; GET 2 AMBER
00136'177776 -2 ; BITS.
00137'000006 6 ; BIAS OF 3 SEC (IN .5 SEC UNITS)
00140'135220 MOVZR 1,3 ; .5 SEC UNITS TO AC3
00141'125120 MOVZL 1,1 ; MULTIPLY .5 UNITS BY 4.
00142'167000 ADD 3,1 ; ADD.5 UNITS=.1 SEC UNITS.
00143'046007$ STA 1,@AINCS; STORE AMBER TIME
00144'151404 INC 2,2,SZR ; COUNT AMBERS
00145'000770 JMP AMLOP ; LOOP

;
; SET VARIABLE PARAMETERS FOR MCCM,MKBD,FORS BY
; PROCESSING BITS 13,14 OF INTCD.
      SKPMT:
000000 LDA 0,INTCD ; INTERSECTION CODE WORD
00146'020210$ LDA 1,C16 ; AC1 = 1B11
00147'024054$ LDA 2,AATAB ; AC2 = ADDR OF ATABL
00150'030543 LDA 3,C6 ; AC3 = 6
00151'034540 ANDZR 3,0,SNR ; TEST BITS = 00 (NORMAL INTR)
00152'163625 JMP NOTSP ; YES, NOT SPECIAL
00153'000411 MOVZR# 0,0,SBN ; NO,TEST BITS = 11(MCCM)
00154'101237 JMP NOTSP+1 ; NO,FORS OR MKBD
00155'000410 LDA 0,MCSPPH ; AT MCCOMMAS CHANGE
00156'020556 STA 0,PHAST+4 ; PHASES 5&6 TO ALL RED
00157'040552 LDA 0,MCSPPH+1 ; ON "A" SIDE, THEN
00160'020555 STA 0,PHAST+5 ; EAST-BOUND ONLY
00161'040551 .ZAC 0,0 ; CLEAR INDICATORS
00162'102460 STA 0,6,2; DISALLOW CODE 5 FROM TLM
000000 NOTSP:
00164'125120 MOVZL 1,1 ; YES, AC1 = 1B10
00165'030055$ LDA 2,C8 ; AC2= 8.
00166'101222 MOVZR 0,0,SZC ; TEST B14 = 1 (FOREST)
00167'175401 INC 3,3,SKP ; YES, AC3 = 7,AC2 = 9
00170'171000 MOV 3,2 ; NO, AC2 = 7
00171'151400 INC 2,2 ;
00172'046520 STA 1,@AZ1V5; STORE Z+1 VOL BIT 5
00173'052520 STA 2,@AATAB; STORE # OF A-CODES
00174'054524 STA 3,NPHAS ; STORE # OF EMER PHASES.

```

;

t t t

; FLASH PROCESSING

;

00175'152460 .ZAC 2,2 ; CLEAR AC2
00176'020166\$ LDA 0,CNFLS ; GET # OF CONFIRM
00177'024650 LDA 1,MAXCF ; FLASH ERRORS IN LAST PERIOD
00200'122032 .SGE 1,0 ; AND COMPARE TO MAX ALLOWED
00201'040644 STA 0,PERFL ; GREATER, SET PERMANENT FLASH
00202'014642 DSZ CNERT ; DECR CONFIRM ERR TIMER
00203'000404 JMP CNFOK ; STILL OK
00204'050166\$ STA 2,CNFLS ; CLEAR CONFIRM ERROR COUNT
00205'020641 LDA 0,CNFTM ; RESTART THE
00206'040636 STA 0,CNERT ; CONFIRM ERROR TIMER
00207'020636 CNFOK: LDA 0,PERFL ; TEST PERMANENT
00210'101005 .SNZ 0,0 ; FLASH FLAG SET. SKIP IF SO
00211'014145 DSZ FLTIM ; NO, HAS FLASH TIME EXPIRED
00212'000456 JMP CKAFL ; NO, CONTINUE FLASH
00213'010145\$ ISZ FLTIM ; YES, RESET TIMER
00214'020153\$ LDA 0,ASTA ; TEST FLASH
00215'101203 MOVR 0,0,SNC ; BIT SET
00216'000415 JMP NOFLS ; NO
00217'020071\$ LDA 0,IAFLB ; TM INPUT FLASH BIT
00220'060277 INTDS ; DONT ALLOW TASK 2
00221'024046\$ LDA 1,LAKC0 ; LAST CODE 0 TM INPUT
00222'107404 AND 0,1,SZR ; TEST FLASH REQUEST FROM TM
00223'000450 JMP TMAFL ; YES, LEAVE FLASH ALONE
00224'050153\$ STA 2,ASTA ; CLEAR AMBER STATUS (FLASH)
00225'102000 .GM1 0,0 ; ALL ONES
00226'040154\$ STA 0,RSTA ; SET ALL RED BITS
00227'050165\$ STA 2,ECOMC ; CLEAR EMER COMMAND = ALL RED
00230'040164\$ STA 0,EMMOD ; SET EMER MODE = -1
00231'060177 INTEN ; ENABLE RUPTS
00232'000555 JMP STPS1 ; GO START NEW PHASE

; NOT IN FLASH MODE

00233'020163\$ NOFLS: LDA 0,TMERF ; TM INPUT ERROR FLAG
00234'101004 .SZR 0,0 ; IF SET,
00235'000405 JMP SEMRC ; START EMER CTL
00236'020046\$ LDA 0,LAKC0 ; TEST EMER CTL
00237'024070\$ LDA 1,IONLB ; REQUEST FROM TM
00240'107404 AND 0,1,SZR ; INPUT
00241'000434 JMP CEMRM ; NO
00242'020164\$ SEMRC: LDA 0,EMMOD ; TEST EMER MODE
00243'101004 .SZR 0,0 ;
00244'000434 JMP CEMRC ; IN PROGRESS, CONTINUE
00245'060277 INTDS ; DISABLE RUPT
00246'010164\$ ISZ EMMOD ; SET EMER MODE FLAG
00247'020152\$ LDA 0,GSTA ; CURRENT GREEN STATUS
00250'034451 LDA 3,GRBTS ; SIGNAL GREEN BITS
00251'163400 AND 3,0 ; KEEP SIGNAL BITS
00252'040165\$ STA 0,ECOMC ; TO EMER COMMAND
00253'060177 INTEN ; ALLOW RUPT
00254'024443 LDA 1,MAPHS ; ADDR-1 OF PHASE TABLE
00255'044007\$ STA 1,AINC5 ; TO AUTO INCR LOC
00256'152520 .GP1 2,2 ; PHASE COUNTER
00257'034441 LDA 3,NPHAS ; # OF PHASES TO AC3
00260'174400 NEG 3,3 ; MAKE NEG
00261'026007\$ CPSLP: LDA 1,@AINC5 ; GREEN CONFIG FROM PHASE TABLE
00262'124000 COM 1,1 ; MAKE ZEROES ONES
00263'107415 AND# 0,1,SNR ; TEST FOR SAME GREENS
00264'000524 JMP SNWPS ; YES

```

---
00265'151400      INC      2,2      ; INCR PHASE COUNT
00266'175404      INC      3,3,SZR  ; NO, INCR PHASE #
00267'000772      JMP      CPSLP   ; TEST ALL PHASES
00270'102460      CK AFL:  .ZAC     0,0      ; NONE FOUND
00271'060277      INTDS      ; DISABLE RUPTS
00272'006130S     JSR      @ZSTFL  ; SET UP MINIMUM FLASH MODE
00273'060177      TMAFL:  INTEN    ; ENABLE RUPTS
00274'000527      JMP      ENDT3   ; CONTINUE
;
00275'020164S     CEMRM:  LDA      0,EMMOD ; TEST EMER
00276'101005      .SNZ     0,0      ; MODE IN PROGRESS
00277'000524      JMP      ENDT3   ; NO, CONTINUE
00300'020154S     CEMRC:  LDA      0,RSTA  ; CURRENT RED STATUS
00301'024165S     LDA      1,ECOMC  ; EMER COMMAND
00302'034417      LDA      3,GRBTS  ; REMOVE PED
00303'163400      AND      3,0      ; BITS
00304'124000      COM      1,1      ; NOT DESIRED GREENS
00305'167400      AND      3,1      ; REMOVE PED BITS
00306'106414      .SEQ    0,1      ; TEST CURRENT OUTPUT
00307'000514      JMP      ENDT3   ; NOT SAME
00310'000426      JMP      INEMR   ; SAME AS COMMANDED
;
00311'000006      C6:      6          ; CONST 6
00312'177777      @Z1V5:  Z1V5    ; ADDR OF Z1VBT+5 (TASK 1)
00313'177777      AATAB:  ATABL   ; ADDR OF ATABL (TASK 2)
00314'000000      GRTO:   0          ; GREEN TIMER
00315'000000      EPHSN:  0          ; PHASE #
00316'000000      PTIM:   0          ; PHASE TOTAL TIME
00317'000324      MAPHS:  PHAST-1 ; ADDR-1 OF PHASE TABLE
00320'000006      NPHAS:  6          ; # OF PHASES          *****
00321'177600      GRBTS:  177600 ; GREEN BIT MASK
00322'000047      MAMAX:  MAXT-1 ; ADDR-1 OF MAX TIMES
00323'000065      MAEXT:  EXTT-1 ; ADDR-1 OF EXT TIMES
00324'000056      MAMIN:  MINT-1 ; ADDR-1 OF MIN TIMES
;
; PHASE TABLE - GREEN BITS FOR EACH PHASE
      000002      .RDX     2          ; BINARY FOR CLARITY
00325'101400      PHAST:  100000110B8 ; PHASE 1
00326'103000      100001100B8 ; 2
00327'033000      001101100B8 ; 3
00330'034000      001110000B8 ; 4
00331'064000      011010000B8 ; 5
00332'061400      011000110B8 ; 6
00333'041600      010000111B8 ; 7 (FOREST ONLY)
; MCCOMMAS SPECIAL PHASES
00334'004000      MCSPH:  000010000B8 ; PHASE 5
00335'041400      010000110B8 ; PHASE 6
      000010      .RDX     8
;
; ACTUAL EMERGENCY CONTROL SECTION
00336'010760      INEMR:  ISZ     PTIM    ; INCR TOTAL PHASE TIME
00337'030756      LDA      2,EPHSN ; PHASE # TO AC2
00340'034764      LDA      3,MAMIN ; GET MINIMUM INTERVAL
00341'157000      ADD      2,3      ; FOR THIS
00342'025400      LDA      1,0,3 ; PHASE
00343'020753      LDA      0,PTIM  ; TOTAL ELAPSED PHASE TIME
00344'106432      .SGT    0,1      ; TEST FOR MINIMUM INTERVAL
00345'000456      JMP      ENDT3   ; DURING MINIMUM
00346'034754      LDA      3,MAMAX ; AFTER MINIMUM
00347'157000      ADD      2,3      ; GET MAXIMUM PHASE

```

```

---
00350'025400 LDA 1,0,3 ; TIME
00351'122432 .SGT 1,0 ; TEST FOR MAXIMUM EXPIRED
00352'000416 JMP TSEND ; YES
00353'034462 LDA 3,MLPBT ; GET LOOP
00354'157000 ADD 2,3 ; BITS FOR
00355'025400 LDA 1,0,3 ; THIS PHASE
00356'120000 COM 1,0 ; COMPLEMENT LOOP BITS
00357'060277 INTDS ; DONT ALLOW LOOP UPDATE
00360'034160$ LDA 3,LOOPS ; LOOP DETECTOR STATUS
00361'163400 AND 3,0 ; CLEAR CALLS FOR
00362'040160$ STA 0,LOOPS ; THIS PHASE AND RESTORE
00363'060177 INTEN ; ALLOW RUPTS
00364'167404 AND 3,1,SZR ; TEST FOR ANY CALLS
00365'000432 JMP STEXT ; YES, EXTEND THIS PHASE
00366'014726 DSZ GRTO ; NO, DECR GREEN INTERVAL
00367'000434 JMP ENDT3 ; DURING EXTEND INTERVAL
00370'020163$ TSEND: LDA 0,TMERF ; TEST TM EMER
00371'101004 .SZR 0,0 ; CTL FLAG
00372'000410 JMP MADVP ; SET, GO ADVANCE PHASE
00373'020046$ LDA 0,LAKC0 ; TEST EMER CTL
00374'024070$ LDA 1,IONLB ; REQUEST FROM
00375'123405 AND 1,0,SNR ; TM INPUT
00376'000406 JMP ADVNP ; SET, GO ADVANCE PHASE
00377'102460 .ZAC 0,0 ; NO EMER CTL REQUEST PRESENT
00400'040164$ STA 0,EMMOD ; CLEAR EMER MODE
00401'000422 JMP ENDT3 ; CONTINUE
00402'102460 MADVP: .ZAC 0,0 ; CLEAR THE
00403'040163$ STA 0,TMERF ; TM ERR FLAG
00404'151400 ADVNP: INC 2,2 ; INCR PHASE #
00405'020713 LDA 0,NPHAS ; # OF PHASES
00406'112032 .SGE 0,2 ; TEST LAST PHASE
00407'152520 STPS1: .GP1 2,2 ; YES, RESET TO 1
00410'050705 SNWPS: STA 2,EPHSN ; STORE PHASE #
00411'102460 NEWPS: .ZAC 0,0 ; CLEAR TOTAL
00412'040704 STA 0,PTIM ; PHASE TIME
00413'034704 LDA 3,MAPHS ; COMPUTE ADDR IN
00414'157000 ADD 2,3 ; PHASE TABLE
00415'021400 LDA 0,0,3 ; GREEN CONFIG FOR THIS PHASE
00416'040165$ STA 0,ECOMC ; STORE GREEN COMMAND
00417'034704 STEXT: LDA 3,MAEXT ; NOW STORE
00420'157000 ADD 2,3 ; EXTENSION
00421'021400 LDA 0,0,3 ; INTERVAL TIME
00422'040672 STA 0,GRTO ; IN GREEN TIMER
00423'006131$ ENDT3: JSR @ZWAIT ; WAIT FOR NEXT TIME FRAME
00424'002401 JMP @.+1 ; LOOP
00425'000075' EMERL ; ADDR OF EMER CTL LOOP.
;
; LOOP BITS FOR EACH PHASE
00426'100000 LPBTS: 1B0 ; PHASE 1
00427'000000 0 ; 2
00430'000100 1B9 ; 3
00431'001000 1B6 ; 4
00432'000000 0 ; 5
00433'010000 1B3 ; 6
00434'000010 1B12 ; 7
00435'000425' MLPBT: LPBTS-1 ; TABLE ADDR
;
.END ; END SEG 6

```

 A0SAV 000015\$X
 A1SAV 000016\$X
 A2SAV 000017\$X
 A3SAV 000020\$X
 AATAB 000313'
 ADVNP 000404'
 AINC1 000003\$X
 AINC2 000004\$X
 AINC3 000005\$X
 AINC4 000006\$X
 AINC5 000007\$X
 ALMSK 000205\$X
 AMB 000041
 AMLOP 000135'
 AMIM 177777 X
 APDMT 000171\$X
 ASTA 000153\$X
 ASTLF 000144\$X
 ASTWR 000176\$X
 ATABL 000313'X
 AXFLB 000207\$X
 AE1V5 000312'
 BIT 000052\$X
 BITV 000151\$X
 BOFLG 000161\$X
 C1 000060\$X
 C16 000054\$X
 C2 000057\$X
 C32 000053\$X
 C4 000056\$X
 C6 000311'
 C7 000167\$X
 C8 000055\$X
 CDABT 000101\$X
 CEMRC 000300'
 CFMRM 000275'
 CKAFL 000270'
 CLOKF 000203\$X
 CNERT 000044'
 CNFLS 000166\$X
 CNFOK 000207'
 CNFTM 000046'
 CPSLP 000261'
 CROT8 000066\$X
 CYCLE 000012\$X
 DATAM 000116\$X
 EA0SA 000030\$X
 EA1SA 000031\$X
 EA2SA 000032\$X
 EA3SA 000033\$X
 ECOMC 000165\$X
 ECYCL 000025\$X
 EMERC 000000'
 EMERL 000075'
 EMMOD 000164\$X
 ENDT3 000423'
 EPCSA 000027\$X
 EPHSN 000315'
 ESPCE 000024\$X
 ESTAT 000026\$X

```

---
ETETI 0000350X
ETIME 0000345X
EXTT 000066'
FLTIM 000145SX
FORMM 000121SX
GLMSK 000206SX
GRBTS 000321'
GRN 000042
GRTO 000314'
GSTA 000152SX
HITSK 000004
IARTB 000064SX
IAFLB 000071SX
IFBIT 000063SX
IMASK 000142SX
INEMR 000336'
INTCD 000210SX
IONLB 000070SX
IRSYB 000065SX
LAKC0 000046SX
LAKC1 000047SX
LBYTE 000122SX
LOOPS 000160SX
LOW4M 000120SX
LPBTS 000426'
LRJC0 000050SX
LRJC1 000051SX
MB 000043'
MADVP 000402'
MAEXT 000323'
MAMAX 000322'
MAMIN 000324'
MAMTM 000170SX
MAPHS 000317'
MAXCF 000047'
MAXT 000050'
MCSPH 000334'
MHTSK 000113SX
MINT 000057'
MLPBT 000435'
NEWPS 000411'
NOFLS 000233'
NOTSP 000164'
NPHAS 000320'
NSPBT 000016
OAFLB 000074SX
OASTB 000076SX
OBOFB 000077SX
OCAKB 000072SX
OCDAB 000100SX
OEMRB 000075SX
ORSYB 000067SX
OWRDM 000117SX
PCINT 000001SX
PCSAV 000014SX
PDMNT 177777 X
PDMT1 177777 X
PDMT2 177777 X
PDMT4 177777 X
PDMT5 177777 X

```

 PDMT6 177777 X
 PEOTR 000062SX
 PERFL 000045'
 PHAST 000325'
 PICKR 000005'
 PKAER 000201SX
 PAGER 000202SX
 PRIWR 000174SX
 PRKER 000200SX
 PR3P1 000156SX
 PR3P2 000157SX
 PSVA2 000020'
 PTIM 000316'
 PTMR 000011SX
 RED 000040
 RLMSX 000204SX
 RKFLG 000177SX
 RKOTB 000061SX
 RSFLG 000146SX
 RSTA 000154SX
 RSWRD 000173SX
 SCEDF 000115SX
 SCERR 000073SX
 SEMRC 000242'
 SGSTA 000162SX
 SKPMT 000146'
 SNWPS 000410'
 SPCEL 000002SX
 SPTAB 000023SX
 SSDNB 000102SX
 STATE 000013SX
 STBTS 000123SX
 STEXT 000417'
 STL 000057
 STMR 000010SX
 STPS1 000407'
 STWRD 000175SX
 TASK3 000000'
 TETIM 000022SX
 TIBUF 000103SX
 TIFIR 000104SX
 TIIN 000105SX
 TIMER 000021SX
 TIN 000027
 TIOUT 000106SX
 TMAFL 000273'
 TMERF 000163SX
 TMINT 000150SX
 TMOUT 000147SX
 TOBUF 000107SX
 TOFIR 000110SX
 TOIN 000111SX
 TOOUT 000112SX
 TOUT 000043
 TOUTW 000172SX
 TOW1 000036SX
 TOW13 000043SX
 TOW16 000044SX
 TOW17 000045SX
 TOW2 000037SX

TOW3	000040SX
TOW7	000041SX
TOW9	000042SX
TSEND	000370'
UMASK	000143SX
UNPKM	000021'
VOLUM	000155SX
VOUT	000040
XEQPR	000114SX
Z10DB	000000U
Z11DB	000000U
Z1VBS	000312'X
Z8DBG	000000U
Z9DBG	000000U
ZABRT	000136SX
ZALS	000134SX
ZBITN	000135SX
ZCLER	000140SX
ZGET	000125SX
ZIJMP	000133SX
ZIN	000020
ZOR	000126SX
ZPTSB	000141SX
ZPUT	000124SX
ZRSTR	000132SX
ZSMUL	000137SX
ZSTFL	000130SX
ZSTPR	000127SX
ZWAIT	000131SX

PROGRAM SEGMENT 7

.TITL SEGM7

; ;
; IDLE JOB FOR DEBUG PURPOSES. THIS JOB RUNS IN B;
; IT CONTROLS THE TTY FOR MONITOR AND UPDATE FUNC;
; AAAAAACR = MONITOR. AAAAA VVVVVVCR = UPDATE.
; . SUBSTITUTED FOR AAAAA INDICATES
; PREVIOUS AAAAA+1.
; ILLEGAL CHAR = IGNORE.

.ENT IDLE

000000 IDLE:
000001'102460 REINI: .ZAC 0,0 ; OUTPUT A
000002'004562 JSR PUTC ; CR,LF
000003'020537 LDA 0,ASTER ; OUTPUT AN
000004'004560 JSR PUTC ; *
000005'060210 NIOC TTI ; CLEAR TTY INPUT
000006'063610 CHNCH: SKPDN TTI ; ANY TTY INPUT.
000007'000442 JMP MONTR ; NO, MONITOR.
000008'004473 JSR TINIT ; YES. INITIALIZE TO KILL MONITQ
000009'004503 NXCHR: JSR GETC ; READ TTY CHAR. INTO AC0
000010'014532 DSZ FIRST ; CHECK FIRST CHAR
000011'000412 JMP NOTFR ; NO
000012'024527 LDA 1,PEROD ; YES
000013'106414 .SEQ 0,1 ; CHECK FIRST CHAR=.
000014'000407 JMP NOTFR ; NO
000015'024522 LDA 1,MADDR ; YES, GET LAST ADDR.
000016'125400 INC 1,1 ; ADDR+1
000017'044514 STA 1,INPT0 ; STORE NEW ADDR
000018'126520 .GP1 1,1 ; SET REMAINING
000019'044502 STA 1,CHCNT ; CHARS=1
000020'000454 JMP WNCHR ; GET NEXT CHAR
000021'000000 NOTFR:
000022'014500 DSZ CHCNT ; DECREMENT CHAR. COUNTER.
000023'000436 JMP CHOCT ; CONVERT OCT TO BIN.
000024'014477 DSZ IMODE ; LAST CHAR. DECR MODE COUNTER.
000025'000427 JMP ENVAL ; END OF UPDATE VALUE.
; END OF ADDR INPUT. CHECK MONITOR OR UPDATE.
000026'024504 LDA 1,INPT0 ; MOVE INPUT VALUE
000027'044502 STA 1,IADDR ; TO ADDR LOCN.
000028'101005 .SNZ 0,0 ; TEST CR
000029'000411 JMP MONIT ; YES, MONITOR.
000030'024473 LDA 1,SPCH ; NO CHECK LAST CHAR = SPACE.
000031'106414 .SEQ 0,1 ;
000032'000742 JMP REINI ; NO, ILLEGAL.
; PREPARE FOR UPDATE
000033'020471 LDA 0,NVLCH ; # OF UPDATE VALUE CHARACTERS
000034'040464 STA 0,CHCNT ; TO CHAR COUNTER.
000035'102460 .ZAC 0,0 ;
000036'040472 STA 0,INPT0 ; CLEAR INPUT LOCN
000037'000434 JMP WNCHR ; GO GET NEXT CHAR.
; BEGIN MONITOR
000038'024467 MONIT: LDA 1,IADDR ; MOVE ADDR TO
000039'044473 STA 1,MADDR ; MONITOR ADDR
000040'024777 LDA 1,-1 ; SET LAST VALUE OF VARIABLE BEB
000041'044463 STA 1,LASTV ; MONITORED TO SOMETHING RIDICUO
000042'026470 MONTR: LDA 1,@MADDR; CURRENT VALUE OF MONITORED WOB
000043'020461 LDA 0,LASTV ; PREVIOUS VALUE
000044'044460 STA 1,LASTV ; UPDATE PREVIOUS
000045'106414 .SEQ 0,1 ; TEST FOR CHANGE
000046'004470 JSR BNOCT ; YES, CONVERT TO OCT AND OUTPUT
000047'000730 JMP CHNCH ; CHECK FOR TTY INPUT

```

; END OF UPDATE VALUE INPUT
00056*101004 ENVAL: .SER 0,0 ; TEST CR
00057*000721 JMP REINI ; NO, ILLEGAL
00060*020454 LDA 0,INPT0 ; UPDATE VALUE
00061*042452 STA 0,@IADDR ; STORE IN UPDATE ADDR
00062*000762 JMP MONIT ; MONITOR

; CHECK OCTAL NUMBER
00063*024452 CHOCT: LDA 1,OCT0 ; IS INPUT CHAR >= OCTAL 0 CHAR
00064*106032 .SGE 0,1 ;
00065*000713 JMP REINI ; NO, ILLEGAL
00066*122400 SUB 1,0 ; YES, CONVERT TO BINARY
00067*024055S LDA 1,C8 ;
00070*122432 .SGT 1,0 ; IS INPUT <= 7
00071*000707 JMP REINI ; NO, ILLEGAL
00072*024442 LDA 1,INPT0 ; CURRENT INPUT WORD
00073*127120 ADDEL 1,1 ; SHIFT LEFT 3, LOGICAL
00074*125120 MOVEL 1,1 ;
00075*107000 ADD 0,1 ; ADD IN NEW OCTAL BITS
00076*044436 STA 1,INPT0 ; UPDATE INPUT WORD
00077*063610 WNCHR: SKPDN TTI ; WAIT FOR
00100*000777 JMP -.1 ; NEXT CHAR
00101*000707 JMP NXCHR ; GO GET IT

; INITIALIZE
00102*054434 TINIT: STA 3,DSRET ; SAVE RETURN
00103*102460 .EAC 0,0 ;
00104*040430 STA 0,INPT0 ; INPUT LOCN
00105*020424 LDA 0,NADCH ; STORE # OF ADDR CHARS
00106*040416 STA 0,CHCNT ; IN CHARACTER COUNTER
00107*102520 .GP1 0,0 ;
00110*040415 STA 0,IMODE ; SET MODE=1
00111*040432 STA 0,FIRST ; SET FIRST CHAR FLAG.
00112*002424 JMP 0DSRET ; RETURN

; GET TTY CHAR AND ECHO
00113*054423 GETC: STA 3,DSRET ; SAVE RETN
00114*060610 DIAC 0,TTI ; BRING IN CHAR FROM TTY BUFFER
00115*024422 LDA 1,LOW7 ; LOWER 7 BITS MASK
00116*123400 AND 1,0 ; KEEP LOWER 7 BITS OF CHAR
00117*024407 LDA 1,CRCH ; CR
00120*106415 .SNE 0,1 ; TEST CR
00121*102460 .EAC 0,0 ; YES, OUTPUT CR,LF
00122*004441 JSR PUTC ; NO, ECHO INPUT ON TTY
00123*002413 JMP 0DSRET ; RETURN

; PARAMETERS
00124*000000 CHCNT: 0 ; CHAR COUNTER
00125*000000 IMODE: 0 ; INPUT MODE
00126*000015 CRCH: 15 ; CR
00127*000040 SPCH: 40 ; SPACE
00130*000007 NVLCH: 7 ; # OF VALUE CHARS
00131*000006 NADCH: 6 ; # OF ADDR CHARS
00132*000000 LASTV: 0 ; PREVIOUS MONITOR VALUE
00133*000000 IADDR: 0 ; INPUT ADDR
00134*000030 INPT0: 0 ; OCTAL INPUT
00135*000060 OCT0: 60 ; OCT 0 CHAR
00136*000000 DSRET: 0 ; RETURN FROM SEVERAL SUBRS
00137*000177 LOW7: 177 ; LOWER 7 BITS MASK
00140*000132 MADDR: LASTV ; MONITOR ADDR
00141*000052 ASTER: 52 ; *
00142*000056 PEROD: 56 ; .
00143*000000 FIRST: 0 ; FIRST CHAR FLAG
;

```

```

--
; CONVERT 16 BIT WORD TO OCTAL AND OUTPUT ON TTY
00144'054772 BNOCT: STA 3,DSRET ; SAVE RETN
00145'152620 SUBZR 2,2 ; AC2=100000
00146'020767 BNOLP: LDA 0,OCT0 ; AC0=ASCII ZERO
00147'146443 SUBO 2,1,SNC ; STILL + IF NO CRY
00150'101401 INC 0,0,SKP ; INC ASCII OUTPUT CHAR
00151'147001 ADD 2,1,SKP ; TOO MUCH- ADD BACK
00152'000775 JMP .-3 ;
00153'004410 JSR PUTC ; OUTPUT OCTAL CHAR
00154'151220 MOVZR 2,2 ;
00155'151220 MOVZR 2,2 ; SHIFT RIGHT 3
00156'151224 MOVZR 2,2,SZR ; LAST DIGIT
00157'000767 JMP BNOLP ; NO, LOOP
00160'102460 .ZAC 0,0 ; OUTPUT A
00161'004402 JSR PUTC ; CR,LF
00162'002754 JMP @DSRET ; YES, RETURN

; OUTPUT CHAR ON TTY. NULL PRODUCES CR, LF
00163'063511 PUTC: SKPBZ TTO ; WAIT UNTIL NOT BUSY
00164'000777 JMP .-1 ;
00165'061111 DOAS 0,TTO ; OUTPUT CHAR
00166'101004 .SZR 0,0 ; SKIP IF NULL CHAR
00167'001400 JMP 0,3 ; RETURN
00170'054407 STA 3,SPUT ; SAVE RETN
00171'020735 LDA 0,CRCH ; CR
00172'004771 JSR PUTC ; OUTPUT CR
00173'020405 LDA 0,LFCH ;LF
00174'004767 JSR PUTC ; OUTPUT LF
00175'102460 .ZAC 0,0 ; RESTORE NULL CHAR
00176'002401 JMP @SPUT ; RETURN
00177'000000 SPUT: 0 ; PUTC RETN
00200'000012 LFCH: 12 ; LF
;
.END

```

```

---
A0SAV 000015SX
A1SAV 000016SX
A2SAV 000017SX
A3SAV 000020SX
AINC1 000003SX
AINC2 000004SX
AINC3 000005SX
AINC4 000006SX
AINC5 000007SX
AMB 000041
AMTM 177777 X
APDMT 000171SX
ASTA 000153SX
ASTER 000141'
ASTLF 000144SX
ASTWR 000176SX
BIT 000052SX
BITN 000151SX
BNOCT 000144'
BNOLP 000146'
BOFLG 000161SX
C1 000060SX
C16 000054SX
C2 000057SX
C32 000053SX
C4 000056SX
C7 000167SX
C8 000055SX
CBABT 000101SX
CHCNT 000124'
CHNCH 000005'
CHOCT 000063'
CNFLS 000166SX
CRCH 000126'
COTE 000066SX
CYCLE 000012SX
DATAM 000116SX
DEBUG 000000
DSRET 000136'
EA0SA 000030SX
EA1SA 000031SX
EA2SA 000032SX
EA3SA 000033SX
ECOMC 000165SX
ECYCL 000025SX
EMMOD 000164SX
ENVAL 000056'
EPCSA 000027SX
ESPCE 000024SX
ESTAT 000026SX
ETETI 000035SX
ETIME 000034SX
FIRST 000143'
FLTIM 000145SX
FORMM 000121SX
GETC 000113'
GRN 000042
GSTA 000152SX
HITSK 000004
IABTB 000064SX

```

```

---
IADDR 000133'
IAFLB 000071$X
IDLE 000000'
IFBIT 000063$X
IMASK 000142$X
IMODE 000125'
INPT0 000134'
IONLB 000070$X
IRSYB 000065$X
LAKC0 000046$X
LAKC1 000047$X
LASTV 000132'
LBYTE 000122$X
LFCH 000200'
LOOPS 000160$X
LOW4M 000120$X
LOW7 000137'
LRJC0 000050$X
LRJC1 000051$X
MADDR 000140'
MAMTM 000170$X
MHTSK 000113$X
MONIT 000044'
MONTR 000050'
NADCH 000131'
NOTFR 000024'
NSPRT 000016
NVLCH 000130'
NXCHR 000010'
OAFLB 000074$X
OASTB 000076$X
OBOFB 000077$X
OCAKB 000072$X
OCDAB 000100$X
OCT0 000135'
OEMRB 000075$X
ORSYB 000067$X
OWRDM 000117$X
PCINT 000001$X
PCSAV 000014$X
PDMNT 177777 X
PDMT1 177777 X
PDMT2 177777 X
PDMT4 177777 X
PDMT5 177777 X
PDMT6 177777 X
PEOTB 000062$X
PEROD 000142'
PRIWR 000174$X
PREP1 000156$X
PREP2 000157$X
PTMR 000011$X
PUTC 000163'
RED 000040
REINI 000000'
RRFLG 000177$X
RROTb 000061$X
RSFLG 000146$X
RSTA 000154$X
RSWRD 000173$X

```



```

---
SCEDF 000115SX
SCERB 000073SX
SGSTA 000162SX
SPCEL 000002SX
SPCH 000127'
SPTAB 000023SX
SPUT 000177'
SSDNB 000102SX
STATE 000013SX
STRTS 000123SX
STL 000057
STMR 000010SX
STWRD 000175SX
TETIM 000022SX
TIDUF 000103SX
TIFIR 000104SX
TIIN 000105SX
TIMER 000021SX
TIN 000027
TINIT 000102'
TIOUT 000106SX
TMERF 000163SX
TMINT 000150SX
TMOUT 000147SX
TORUF 000107SX
TOFIR 000110SX
TOIN 000111SX
TOOUT 000112SX
TOUT 000043
TOUTW 000172SX
TOW1 000036SX
TOW13 000043SX
TOW16 000044SX
TOW17 000045SX
TOW2 000037SX
TOW3 000040SX
TOW7 000041SX
TOW9 000042SX
UMASK 000143SX
VOLUM 000155SX
VOUT 000040
VNCHE 000077'
XEQPR 000114SX
XPRBT 000136SX
XALS 000134SX
XBITN 000135SX
XCLFR 000140SX
XGET 000125SX
XJUMP 000133SX
XIN 000020
XOR 000126SX
XPTSB 000141SX
XPUT 000124SX
XRSTR 000132SX
XSMUL 000137SX
XSTFL 000130SX
XSTPR 000127SX
XWAIT 000131SX

```

PROGRAM SEGMENT 8

```

; SYSTEM EQUAS FOR ACTUATOR PROGRAM SEGMENTS. MUST PREFIX
; EACH PROGRAM SEGMENT.
;

```

```

; EXTERNALS

```

```

.EXTD PCINT, SPCEL, AINC1, AINC2, AINC3, AINC4
.EXTD AINC5, STMR, PTMR, CYCLE, STATE, PCSAV, A0SAV
.EXTD A1SAV, A2SAV, A3SAV, TIMER, TETIM, SPTAB
.EXTD ESPCE, ECYCL, ESTAT, EPCSA, EA0SA, EA1SA
.EXTD EA2SA, EA3SA, ETIME, ETETI
.EXTD TOW1, TOW2, TOW3, TOW7, TOW9, TOW13, TOW16
.EXTD TOW17, LAKC0, LAKC1, LRJC0, LRJC1
.EXTD BIT, C32, C16, C8, C4, C2, C1
.EXTD RROTB, PEOTB, IFBIT, IASTB, IRSYB, CROTB
.EXTD ORSYB, IONLB, IAFLB, OCAKB, SCERB, OAFLB
.EXTD OEMRB, OASTB, OBOFB, OCDAB, CDADT, SSDNB
.EXTD TIBUF, TIFIR, TIIN, TIOUT, TOBUF, TOFIR, TOIN
.EXTD TOOUT, MHTSK, XEQPR, SCEDF, DATAM, OWRDM
.EXTD LOW4M, FORMM, LBYTE, STETS
.EXTD ZPUT, ZGET, ZOR, ZSTPR, ZSTFL, ZWAIT, ZRSTR
.EXTD ZIJMP, ZALS, ZBITN, ZABRT, ZSMUL, ZCLER, ZPTSE
.EXTD IMASK, UMASK, ASTLF, FLTIM, RSFLG, TMOUT
.EXTD TMINT, BITN, GSTA, ASTA, RSTA
.EXTD VOLUM, PRZP1, PRZP2, LOOPS, BOFLG, SGSTA
.EXTD TMERF, EMMOD, ECOMC, CNFLS, C7, MAMTM
.EXTD APDMT, TOUTW, RSWRD, PRIWR, STWRD, ASTWR
.EXTD RRFLG, PRER, PRAER, PRGER, CLOKF
.EXTD RLMSK, ALMSK, GLMSK, AXFLB, INTCD
.EXTN PDMNT, AMTM
.EXTN PDMT1, PDMT2, PDMT4, PDMT5, PDMT6

```

```

; EQUIVALENCE DEFINED CONFIGURATION PARAMETERS
;

```

```

; TASK CONTROL

```

```

000004 HITSK = 4 ; NUMBER OF TASKS
;
000020 ZIN = 20 ; INPUT Z ADDR
000040 VOUT = 40 ; OUTPUT V ADDR
000042 GRN = VOUT+2 ; GREEN OUTPUT ADDR
000041 AMB = VOUT+1 ; AMBER OUTPUT ADDR
000040 RED = VOUT ; RED OUTPUT ADDR
000027 TIN = ZIN+7 ; TELEMETRY INPUT ADDR
000043 TOUT = VOUT+3 ; TELEMETRY OUTPUT ADDR
000057 STL = 57 ; STALL OUTPUT ADDR
000016 NSPBT = 14. ; # OF SPEED BITS
.NREL
.EOT ; END SYSTEM EQUAS

```

*

LOADER MAP

*6

NMAX 003560
ZMAX 000050

ACSAV 000057
AISA 000063
ASAV 000067
ASSAV 000073
ABORT 000441
AINC1 000020
AINC2 000021
AINC3 000022
AINC4 000023
AINC5 000024
ALMSK 000241
ALS 000425
ANTM 000331
APDMT 000273
ASTA 000255
ASTLF 000246
ASTNR 000316
ATABL 003060
ATELS 003066
AKFLB 000144
BIT 000137
BITN 000253
BITNM 000433
BOFLG 000263
C1 000156
C16 000152
C2 000155
C32 000151
C4 000154
C7 000271
C8 000153
CDABT 000145
CLEAR 000470
CLOKF 000002
CNFLS 000270
CNOTE 000147
CYCLE 000004
DATAM 000214
EASSA 000063
EAI SA 000067
EASSA 000073
EASSA 000077
ECONC 000267
LCYCL 000010
ENIOD 000266
EPCSA 000057
ESPCE 000121
ESTAT 000053
ETETI 000107
ETIME 000103
FLTIN 000247
FORMM 000217
GET 000370
GLMSK 000242
GSTA 000254

U IASTB 000155
IAFLB 000152
IDLE 177777
IFBIT 000150
IJUMP 000423
IMASK 000244
INITR 000546
INTCD 000243
INTRP 000652
IONLB 000151
IRSYB 000154
LAKC0 000013
LAKC1 000014
LBYTE 000220
LOOPS 000262
LOW4M 000216
LRJC0 000015
LRJC1 000016
MAMTM 000272
MHTSK 000211
OAFLE 000155
OASTE 000145
ODOFB 000146
OCAKB 000154
OCDAB 000153
OEMRB 000153
OR 000342
ORSYB 000150
OVRDM 000215
PCINT 000000
PCSAV 000053
PDMNT 000320
PDMT1 000321
PDMT2 000322
PDMT4 000324
PDMT5 000325
PDMT6 000326
PEOTB 000150
PRAER 000011
PRGER 000012
PRIWR 000313
PRRER 000010
PRZP1 000260
PREP2 000261
PTMR 000036
PTSUB 000500
PUT 000354
RLMSK 000240
RRFLG 000317
RROTB 000146
RSFLG 000250
RSTA 000256
NSWRD 000274
SCEDF 000213
SCERB 000153
SGSTA 000264
SMUL 000443
SPCEL 000103
SPTAB 000121
SSDNE 000141

STATE 000047
STBTS 000221
STFLS 000403
STMR 000025
STPRI 000346
STWRD 000274
TASK0 001273
TASK1 001346
TASK2 002042
TASK3 003122
TLTIM 000103
TIBUF 000157
TIFIR 000162
TIIN 000163
TIMER 000077
TIOUT 000164
TIERF 000265
TINT 000252
TMOUT 000251
TOBUF 000165
TOFIR 000206
TOIN 000207
TOOUT 000210
TOUTW 000274
TOW1 000275
TOW13 000311
TOW16 000314
TOW17 000315
TOW2 000276
TOW3 000277
TOW7 000303
TOW9 000305
UMASK 000245
VOLUM 000257
WAIT 001265
NEQPR 000212
EIVB5 001744
EABRT 000234
EALS 000232
ELITN 000233
ECLER 000236
EGET 000223
EIJMP 000231
EOR 000224
EPTSB 000237
EFUT 000222
ERSTR 000230
ESMUL 000235
ESTFL 000226
ESTPR 000225
EWAIT 000227

*R

APPENDIX G

SYSTEM CONFIGURATION PARAMETERS

<u>PARAMETER NAME</u>	<u>DESCRIPTION</u>	<u>UNITS</u>	<u>CURRENT VALUE</u>	<u>(SEGMENT, TAPE) LOCATION</u>
AMASK	Mask for selecting the movement bits for the "A" intersection.	-	170200 ₈	(5, 4)
AMB	Amber light output device code.	-	41 ₈	(8, 1)
ATABL	A table for decoding signal commands for the "A" intersection as received from telemetry.	-	See Listing	(5, 4)
AVTRA	A table of addresses that defines the average trap cell that each speed detector bit affects.	-	See Listing	(4, 4)
BMASK	Mask for selecting the movement bits for the "B" intersection.	-	007400 ₈	(5, 4)
BTABL	A table for decoding signal commands for the "B" intersection as received from telemetry.	-	See Listing	(5, 4)
CDABT	Cabinet door ajar input bit location in input word Z+3.	-	Bit+6	(0, 1)
CLOKF	Real-time clock frequency selection bits.	-	3	(0, 1)
CNFTM	Time period for counting signal confirm errors. If the number of confirm errors in this time period exceeds MAXCF, go to permanent flash.	.5 seconds	120.	(6, 1)
CROTB	Command reject bit in telemetry output word.	-	Bit+8	(0, 1)
CYCLE+0	Frame time for TASK 0.	msec.	10.	(0, 1)
CYCLE+1	Frame time for TASK 1.	msec.	50.	(0, 1)
CYCLE+2	Frame time for TASK 2.	msec.	100.	(0, 1)
CYCLE+3	Frame time for TASK 3.	msec.	500.	(0, 1)
DATAM	Telemetry output data bits mask.	-	001700 ₈	(0, 2)

<u>PARAMETER NAME</u>	<u>DESCRIPTION</u>	<u>UNITS</u>	<u>CURRENT VALUE</u>	<u>(SEGMENT, TAPE) LOCATION</u>
DEBUG	0 = assemble operational code. 1 = assemble checkout code.	-	0	(0, 1) (8, 1)
FLMIN	Minimum intersection flash time.	.5 seconds	20.	(1, 1)
FORMM	Telemetry output format bits mask.	-	076000 ₈	(0, 2)
GRBTS	Mask to select the green light movement bits from the output word.	-	177600 ₈	(6, 2)
GRN	Green light output device code.	-	42 ₈	(8, 1)
HITSK	Number of tasks.	-	4	(0, 1) (8, 1)
IABTB	Telemetry input abort bit.	-	Bit+14	(0, 1)
IADDR+0	Starting address for TASK 0.	-	TASK 0	(2, 1)
IADDR+1	Starting address for TASK 1.	-	TASK 1	(2, 1)
IADDR+2	Starting address for TASK 2.	-	TASK 2	(2, 1)
IADDR+3	Starting address for TASK 3.	-	TASK 3	(2, 1)
IAFLB	Telemetry input actuator flash bit.	-	Bit+11	(0, 1)
IBDLA	Delay between the last sample of a telemetry input data bit and the first sample of the next data bit.	msec.	5	(2, 1)
IBFSZ	Telemetry input buffer size. Capacity of buffer is IBFSZ-1.	words	3	(0, 1)
IBTTM	Telemetry bit time, i.e., the length of each serial bit.	msec.	7	(2, 1)

<u>PARAMETER NAME</u>	<u>DESCRIPTION</u>	<u>UNITS</u>	<u>CURRENT VALUE</u>	<u>(SEGMENT, TAPE) LOCATION</u>
ICERT	Telemetry input command error time. If a valid signal command is not received in this length of time the emergency controller will be activated.	.1 seconds	600	(5, 2)
IFBIT	Telemetry input format bit mask. This bit determines whether the input is type 0 or 1.	-	Bit+9	(0, 1)
IONLB	Telemetry input "on-line" bit mask.	-	Bit+10	(0, 1)
IRSYB	Telemetry input "request resync" bit mask.	-	Bit+13	(0, 1)
ITNBT	Total number of bits in a telemetry frame (message).	-	12.	(2, 1)
IVARA	Increment to average trap time if measured trap time - average trap time > IVART.	.01 seconds	15.	(4, 2)
IVART	Allowable variation of measured trap time over average trap time.	.01 seconds	30.	(4, 2)
LMASK	Mask to select green signal light movement bits from green output word.	-	177600 ₈	(5, 3)
LPBTS	A table containing masks to select the proper loop detector bits for each emergency phase.	-	See Listing	(6, 2)
MAXCF	The maximum number of signal confirm errors allowed in time period CNFTM before permanent flash is invoked.	-	6	(6, 1)
MAXTP	The maximum measured trap time that will contribute to average speed calculations.	.01 seconds	1800. (1 FPS)	(4, 2)
MINTP	The minimum measured trap time that will contribute to average speed calculations.	.01 seconds	20. (90. FPS)	(4, 2)
MNOCU	- number of occupancy cells. This must be the negative of the highest numbered bit +1 in input Z+1 that is used for occupancy measurements.	-	-13.	(4, 1)

<u>PARAMETER NAME</u>	<u>DESCRIPTION</u>	<u>UNITS</u>	<u>CURRENT VALUE</u>	<u>(SEGMENT, TAPE) LOCATION</u>
MXLAM	The maximum "lambda" value permissible in the average speed calculation.	.01 seconds	128.	(4, 2)
NBITS	Number of telemetry data bits. It is the total number of bits - the number of start, stop bits.	-	9.	(2, 2)
NPHAS	Number of emergency control phases.	-	6 (7 for Forest)	(6, 2)
NSPBT	The number of the highest bit +1 in input Z that contains a magnetometer input.	-	14.	(0, 1) (8, 1)
NVALG	The number of entries in the valid green output table (VALGN).	-	15.	(5, 3)
OAFLB	Telemetry output actuator flash bit shifted right 6.	-	Bit+14	(0, 1)
OASTB	Telemetry output abort/stall bit.	-	Bit+6	(0, 1)
OBFSZ	Telemetry output buffer size. Capacity of buffer is OBFSZ-1.	words	17.	(0, 1)
OBOFB	Telemetry output buffer overflow bit.	-	Bit+7	(0, 1)
OCAKB	Telemetry output command acknowledge bit, shifted right 6.	-	Bit+13	(0, 1)
OCDAB	Telemetry output cabinet door ajar bit, shifted right 6.	-	Bit+12	(0, 1)
OCINI	- occupancy counter overflow time. Total occupancy time over a loop detector of this amount will cause a telemetry output occupancy bit to be sent.	.05 seconds	-100.	(4, 1)

<u>PARAMETER NAME</u>	<u>DESCRIPTION</u>	<u>UNITS</u>	<u>CURRENT VALUE</u>	<u>(SEGMENT, TAPE) LOCATION</u>
OEMRB	Telemetry output emergency control bit, shifted right 6.	-	Bit+12	(0, 1)
ORSYB	Telemetry output "resync request" bit.	-	Bit+9	(0, 1)
OWRDM	Mask for telemetry output code and output data bits.	-	077700 ₈	(0, 2)
PEOTB	Telemetry output "parity error" bit.	-	Bit+9	(0, 1)
PHAST	Emergency control phase table. Contains the green movement output bits for each phase.	-	See Listing	(6, 2)
RED	Red light output device code.	-	40 ₈	(8, 1)
RFLTM	Restart flash time. The time period that the intersection will flash after a restart.	.5 seconds	30.	(5, 1)
RROTB	Telemetry output "restart performed" bit.	-	Bit+7	(0, 1)
RSTIM	The time period that the actuator will remain in resync mode once it is entered.	msec.	252. (3 frames)	(2, 2)
SCERB	Telemetry output "signal confirm error" bit, shifted right 6.	-	Bit+12	(0, 1)
SDATA	Number of samples of each telemetry input bit to be compared to the first sample.	-	2	(2, 2)
SDSFB	Mask to select signal shutdown and signal flash input bits from input Z+3.	-	030000 ₈	(5, 1)
SSDNB	Signal shutdown bit in input Z+3.	-	Bit+2	(0, 1)
SSTAR	Number of start bit samples to check after the start bit is detected in telemetry input.	-	5	(2, 2)

<u>PARAMETER NAME</u>	<u>DESCRIPTION</u>	<u>UNITS</u>	<u>CURRENT VALUE</u>	<u>(SEGMENT, TAPE) LOCATION</u>
SSTOP	Number of stop bit samples to check after the last data bit in telemetry input.	-	11.	(2, 2)
STBTS	Start and stop bits to be logically "OR'ed" with telemetry output data.	-	000070 ₈	(0, 2)
STL	Stall alarm output device code.	-	57 ₈	(8, 1)
STPDL	Delay time between last sample of last data bit and first sample of stop bit in telemetry input.	msec.	3	(2, 2)
TIN	Telemetry input (receiver) device code.	-	27 ₈	(8, 1)
TOUT	Telemetry output (transmitter) device code.	-	43 ₈	(8, 1)
VALGN	A table of valid green movement output bits. Only those green bits given in the table can be on simultaneously.	-	See Listing	(5, 4)
VOUT	Device code for output V.	-	40 ₈	(8, 1)
ZIN	Device code for input Z.	-	20 ₈	(8, 1)
ZP1 OM	Bit mask to select the occupancy detectors (loops) from input Z+1.	-	113130 ₈	(4, 1)
ZSPMS	Bit mask to select the magnetometer detectors from input Z.	-	177774 ₈	(3, 1)
ZOVBT	A table of bits that defines the telemetry output volume bits for input Z.	-	See Listing	(4, 4)
Z1OBT	A Table of bits that defines the telemetry output occupancy bits for input Z+1.	-	See Listing	(4, 4)

<u>PARAMETER NAME</u>	<u>DESCRIPTION</u>	<u>UNITS</u>	<u>CURRENT VALUE</u>	<u>(SEGMENT, TAPE) LOCATION</u>
Z1VBT	A table of bits that defines the telemetry output volume bits for input Z+1.	-	See Listing	(4, 4)
Z2VBT	A table of bits that defines the telemetry output volume bits for input Z+2.	-	See Listing	(4, 4)