618-235, Volume I

VOYAGER
COMPUTER COMMAND SUBSYSTEM
FLIGHT SOFTWARE
DESIGN DESCRIPTION

REVISION G

8 September 1980

Approved:

S. Lingon
CCS Cognizant Engineer
Voyager

C. P. Jones, Chief
Spacecraft Team
Voyager

R. F. Stott
Division 36 Representative
Voyager

E. L. McKinley, Manager
Flight Engineering Office
Voyager

D. M. Durham
Spacecraft System
Software Engineer
Voyager

Change	Date
7	6 July 1983

1 of 205

DOCUMENT CHANGE NOTICE

Authorized:

618-235 Rev: G Change No.: 7
Vol. I

Date Issued: 07/06/83

Title: Voyager Computer Command Subsystem
Flight Software Design Description
Volume I

Other Documents Affected:

REMARKS: The changes specified herein are incorporated in accordance with the requirements specified by the ECRs listed on the Document Change Log.

APPROVED:

R. S. Lemon
CCS Cognizant Engineer
Voyager

H. P. Marderness
H. P. Marderness, Chief
Spacecraft Team
Voyager

R. Stott
R. F. Stott
Division 36 Representative
Voyager

S. S. Dallas
S. S. Dallas, Manager
Flight Engineering Office
Voyager

Mike Urban
M. G. Urban
Spacecraft System
Software Engineer
Voyager

Document Change #7

Change CCS Flight Software Design Description, Volume I, as follows:

1. Corrections to page v are not related to Rev. G, Change 7 ECRs.
2. Replace pages 3-34 and 3-36 per ECR 37452A
3. Replace page 3-37 per ECR 37447.
4. Replace page 3-113 per ECR 37457.

Changes are designated by vertical margin bars.

19

DOCUMENT CHANGE NOTICE

618-235 Rev: G Change No.: 6

Date Issued: 7/26/82

Title: Computer Command Subsystem

Flight Software Design Description

Volume I

Distribution list attached

REMARKS: The changes specified herein are incorporated in accordance with the requirements specified by the ECRs listed on the Document Change Log.

APPROVED:

D. J. Eisenman 7/1/82
D. J. Eisenman
CCS Cognizant Engineer
Voyager

H. P. Marderness
H. P. Marderness, Chief
Spacecraft Team
Voyager

R. F. Stott 7/7/82
R. F. Stott
Division 36 Representative
Voyager

E. L. McKinley 7/23/82
E. L. McKinley, Manager
Flight Engineering Office
Voyager

J. A. Milavec 7/8/82
J. A. Milavec
Spacecraft System
Software Engineer
Voyager

Document Change #6

Change CCS Flight Software Design Description, Volume I, as follows:

1. Corrections to pages 3-25, 3-29, 3-82, 3-83, 3-98, A-24, and A-25 are not related to Rev. G, Change #6 ECRs.
2. Replace page 3-130 per ECRs 37411 and 37424.
3. Replace page 3-132 per ECR 37411.

Changes are designated by vertical margin bars.

1b

DOCUMENT CHANGE NOTICE

618-235 Rev: G Change No.: 5

Date Issued: 25 Feb 1982

Distribution list attached

Title: Computer Command Subsystem

Flight Software Design Description, Volume I

REMARKS: The changes specified herein are incorporated in accordance with the requirements specified by the ECRs listed on the Document Change Log.

Document Change #5

Change CCS Flight Software Design Description, Volume I, as Follows:
pages 3-8, 3-9, 3-13, 3-16, 3-17, 3-18, 3-25, 3-26, 3-28, 3-34, 3-36, 3-98, 3-125, 3-126, 3-130, 3-132, changes are designated by vertical margin bars.

APPROVED:

D. J. Eisenman 2/10/82
D. J. Eisenman
CCS Cognizant Engineer
Voyager

H P Marderness 2/18/82
H. Marderness, Chief
Spacecraft Team
Voyager

R. F. Stott 2/11/82
R. F. Stott
Division 36 Representative
Voyager

E. L. McKinley 2/19/82
E. L. McKinley, Manager
Flight Engineering Office
Voyager

J. O. Milavec 2/18/82
J. Milavec
Spacecraft System
Software Engineer
Voyager

1-C

DOCUMENT CHANGE NOTICE

618-235 Vol. I Rev: G Change No.: 4

Date Issued: 16 July 1981

Title: Computer Command Subsystem, Flight Software Design Description, Volume I

Distribution list attached

REMARKS: These changes specified herein are incorporated in accordance with the requirements specified by the ECRs listed in the Document Change Log.

Document Change #4

Remove the old pages and replace with the attached pages.

Approved:

U. S. Lingon
CCS Cognizant Engineer
Voyager

T. Risa, Chief
Spacecraft Team
Voyager

R. F. Stott
Division 36 Representative
Voyager

E. L. McKinley, Manager
Flight Engineering Office
Voyager

D. M. Durham
Spacecraft System
Software Engineer
Voyager

2

DOCUMENT CHANGE NOTICE

618-235 Rev: G Change No.: 3
Volume I

Date Issued: 8 Sept 1980

Distribution list attached

Title: Computer Command Subsytem, Flight Software Design Description

Volume I

REMARK: These changes specified herein are incorporated in accordance with the ECRs listed in the Document Change Log.

Document Change #3:
Remove the old pages and replace with the attached pages.

Approved:

U. S. Lingon
CCS Cognizant Engineer
Voyager

C. P. Jones, Chief
Spacecraft Team
Voyager

R. F. Stott
Division 36 Representative
Voyager

E. L. McKinley, Manager
Flight Engineering Office
Voyager

D. M. Durham
Spacecraft System
Software Engineer
Voyager

DOCUMENT CHANGE NOTICE

618-235 Rev: G Change No.: 2
Volume I

Date Issued: 22 Jan 1980

DISTRIBUTION: List Attached

Title: Computer Command Subsystem, Flight Software Design Description

Volume I

REMARKS: These changes specified herein are incorporated in accordance with the ECR requirements delineated in Document Change Log.

Document Change No. 2

Remove the old pages and replace with the attached pages.

Approved:

U. S. Lingon
CCS Cognizant Engineer
Voyager

C. P. Jones
Spacecraft Team Chief
Voyager

R. F. Stott 9/7/79
R. F. Stott
Division 36 Representative
Voyager

R. P. Laeser 1/21/80
R. P. Laeser
Mission Director
Voyager

D. M. Durham 1/15/80
D. M. Durham
Spacecraft System Software
Engineer
Voyager

R. L. Heacock 1/21/80
R. L. Heacock
Project Manager
Voyager

4

DOCUMENT CHANGE NOTICE

618-235 Rev: G Change No.: 1
Volume I

Date Issued: 11 June 1979

DISTRIBUTION: List Attached

Title: Computer Command Subsystem, Flight Software Design Description Volume I

REMARKS: These changes specified herein are incorporated in accordance with the ECR requirements delineated in Document Change Log.

Document Change No. 1

Remove the old pages and replace with the attached pages.

Approved:

U. S. Lingon
CCS Cognizant Engineer
Voyager

G. Cunningham
Spacecraft Team Chief
Voyager

R. F. Stott
Division 36 Representative
Voyager

R. P. Laeser
Mission Director
Voyager

C. P. Jones
Spacecraft System Software Engineer
Voyager

R. L. Heacock
Deputy Project Manager
Voyager

DISTRIBUTION

J. D. Acord	125-147
W. Cook	264-519
J. Harris	264-535
T. Hogle	264-519
R. Lemon (2)	264-535
H. Marderness	264-519
J. Mehta	264-519
J. Milavec	264-519
S. R. Hofmann (5)	264-519
R. Otamura	264-519
A. Sohus	111-100
B. Steinberg	125-241
M. Urban	264-519
L. Zottarelli	264-535
VGR Program File	264-443

DOCUMENT LOG

Date	Routine Mnemonic	Status
21 July 1977	All	Rev. F 36404, 36452, 36457 36483, 36495, 36502 36504, 36505, 36522 36593, 36609, 36665 36701, 36703, 36735 36741, 36771, 36800
12 Apr 1979	AACSIN, DTRPRC, TRNSUP, CMDLOS, PWRCHK, RFLOSS, DSSCAN	Rev. G 36974, 36984, 36986 36996, 37010, 37042 37047, 37051, 37052 37055, 37063, 37066 37082, 37086, 37099 37100, 37103, 37105 37106, 37112, 37139
11 June 1979	AACSIN, PWRCHK, RFLOSS	Rev G Change 1 37140, 37146, 37157 37091, 37160, 37167 37169, 37119
22 Jan 1980	AACSIN, PWRCHK	Rev G Change 2 37200, 37207
08 Sept 1980	AACSIN, DTRPRC, PWRCHK, RFLOSS	Rev G, Change 3 37081A 37224, 37245, 37253, 37257C 37268, 37269 37270A, 37271, 37276, 37282B 37283A, 37308, 37309

DOCUMENT LOG

Date	Routine Mnemonic	Status
16 July 1981	AACSIN, PWRCHK, A-8	Rev. G, Change #4 37315, 37364, 37370, 37371
25 Feb 1982	AACSIN, PWRCHK, RFLOSS, TRNSUP	Rev. G, Change #5 37378, 37392, 37400, 37405, 37407
26 July 1982	RFLOSS	Rev. G, Change #6 37411, 37424
6 July 1983	AACSIN, CMDLOS	Rev. G, Change #7 37457, 37447 37452 A

ROUTINE MNEMONIC	Flight Software Revision Letter (618-235 Vol. II Rev.)							
	A	B	C	D	E	F	G	CHG
TRAPS	A	A	A	A	A	B	B	
COINTS	A	B	C	D	E	F	F	
AACSIN	A	B	C	D	E	F	G	1
TLMDRV	A	A	A	B	B	B	B	
CHKSUM	A	A	A	B	B	B	B	
OUTDRV	A	A	A	A	A	A	B	
MEMLOD	A	A	A	A	A	B	B	
GLBCNT	A	B	B	B	B	B	B	
CMDPRC	A	A	B	C	D	D	D	
ERROR	A	A	A	A	A	B	B	
CMDLOS	A	A	A	A	B	C	D	
IRSPWR	A	B	B	B	B	B	B	
PWRCHK	A	B	C	C	D	E	E	1
RFLOSS	A	B	C	C	D	D	E	1
TARMEX	A	A	A	A	A	B	B	
ANTUPD	A	A	A	A	A	A	A	
DTRPRC	A	B	C	C	C	C	D	
SCNPLT	A	B	B	B	C	D	E	
TRNSUP	-	-	A	B	C	C	D	1
DMLOAD	-	-	-	-	-	A	B	
LHRST	A	B	C	D	E	F	F	
VARABL	A	B	C	D	E	F	G	
PARM2A	A	A	B	B	C	C	C	
PARM2B	A	A	B	B	C	C	C	
PARM3A	A	A	B	B	C	C	C	
PARM3B	A	A	B	B	C	C	C	
DSSCAN	-	-	-	-	-	-	A	

The program listings of Volume II contain the revisions of the routines shown above. The narrative descriptions are contained in Volume I.

ROUTINE REVISION STATUS

Change #1
6/11/79

Figures

10

Figures

Tables

GLOSSARY

The following list gives the acronyms and abbreviations used in this document. Instruction mnemonics and programming symbols are defined in the instruction set or in the descriptions of each routine.

ACC	Accumulator register (hardware) within the CCS processor.
AACS	Attitude and Articulation Control Subsystem
BOT	Beginning Of Tape, the earliest position on the tape recorder at which data may be recorded.
CC	Coded Command, a serial data bit-stream to a user.
CCS	Computer Command Subsystem.
DC	Discrete Command, a switch closure to a user.
DSN	Deep Space Network
DSS	Data Storage Subsystem
DTR	Digital Tape Recorder, part of the Data Storage Subsystem.
ECR	Engineering Change Request
EOT	End of Tape, location on the tape recorder beyond which no recording is allowed.
FDS	Flight Data Subsystem
ID	Identification Code
IMR	Interrupt Mask Register
LSB	Least Significant Bit
LSH	Least Significant Half
MACRO	A software program that expands names and parameters into spacecraft commands or timed sequences of events.
MDS	Modulation-Demodulation Subsystem
MEM	MEMory
MSB	Most Significant Bit
MSH	Most Significant Half

12

OP ID	OPeration IDentification
OU	Output Unit
P/B	PlayBack
PC	Processor Command, a ground command which affects only the CCS.
PC	Program Counter
PPS	Pulse Per Second
TBD	To Be Determined
TIC	Tape Increment Count
TLM	Telemetry
TWTA	Traveling Wave Tube Amplifier
XX_8	Octal number, the digits denoted by X's may take on values of 0 through 7
CDU	Command Detector Unit
CMD	Command
I/O	Input/Output
MIRIS	Modified Infrared Interferometer and Radiometer
ISS	Imaging Science Subsystem
Kbps	Kilo-bits per second
NA	Narrow Angle (camera)
POR	Power On Reset
S/C	Spacecraft
WA	Wide Angle (camera)
VGR	Voyager

12

1.0 INTRODUCTION

1.1 Purpose

The main purpose of this document is to establish a design control document for the software routines of the VGR CCS Flight Software. These software routines, along with the VGR sequence macros given in 618-551, comprise the complete CCS flight software that will be used in VGR spacecraft system testing and flight operations. This document will also serve as a tutorial guide and users' manual for CCS programmers.

1.2 Scope

The design documentation for each software routine consists of a flow diagram and the assembly language code (Volume II) with comments. The narrative descriptions of the operations performed by each routine and how it communicates with other routines are in section 3. A program structure overview preceding the design description of the routines presents a general description of the functions performed by the routines and their interrelationships.

2.0 APPLICABLE DOCUMENTS

The following documents form a part of this design description:

REQUIREMENTS

Jet Propulsion Laboratory

MJS77-4-2005-1	Functional Requirement, Voyager 1977 Flight Equipment, Computer Command Subsystem Hardware
MJS77-4-2005-2	Functional Requirement, Voyager 1977 Flight Equipment, Computer Command Subsystem Software

DOCUMENTS

Jet Propulsion Laboratory

618-58	Voyager 1977 Software Management Plan
618-212	Voyager 1977 Spacecraft Block Dictionary
618-551	Voyager 1977 Sequence Translator Macro Library
618-552	Voyager 1977 Sequence Generation Algorithm

3.0 DESIGN DESCRIPTION

3.1 General Program Structure

The CCS flight software may be divided into the three main categories common to most programming structures associated with real time data processing, namely INPUT, INTERMEDIATE PROCESSING, and OUTPUT. Figure 3-1 illustrates this structure. This form, in itself, contains very little information as all real things operate in a similar fashion: from washing machines (put clothes in for INPUT, wash clothes for INTERMEDIATE PROCESSING, and take out clothes for OUTPUT), to pocket calculators (key in number to add for INPUT, key add for INTERMEDIATE PROCESSING, it displays number for OUTPUT), to the human being (sight, hearing, touch for INPUT, reactions and thinking for INTERMEDIATE PROCESSING, and speech, mechanical movements for OUTPUT). This last example illustrates that the three classifications may be highly interdependent, no one truly separate from the other, in fact, deleting one group, disables or eliminates the entire system.

These three functions, thus are not rigorous guidelines for the flight software. A given unit of code called a ROUTINE may, in a nebulous manner, belong to one of these groups, but at the same time perform functions very similar to the other two. For example, the routine AACSIN (Attitude and Articulation Control Subsystem power code INput processing routine) a member of the INPUT category, will respond to an interrupt from AACS and read in 6 bits of data from the level input matrix (an INPUT function), decode this information as to its meaning (an INTERMEDIATE PROCESSING function), and generate an echo response to AACS to acknowledge the receipt of the data (an OUTPUT function). From this example, it may appear that the routines are autonomous, but this is not true either. In order for it to function, the AACSIN routine requires the presence of the TRAPS routine for

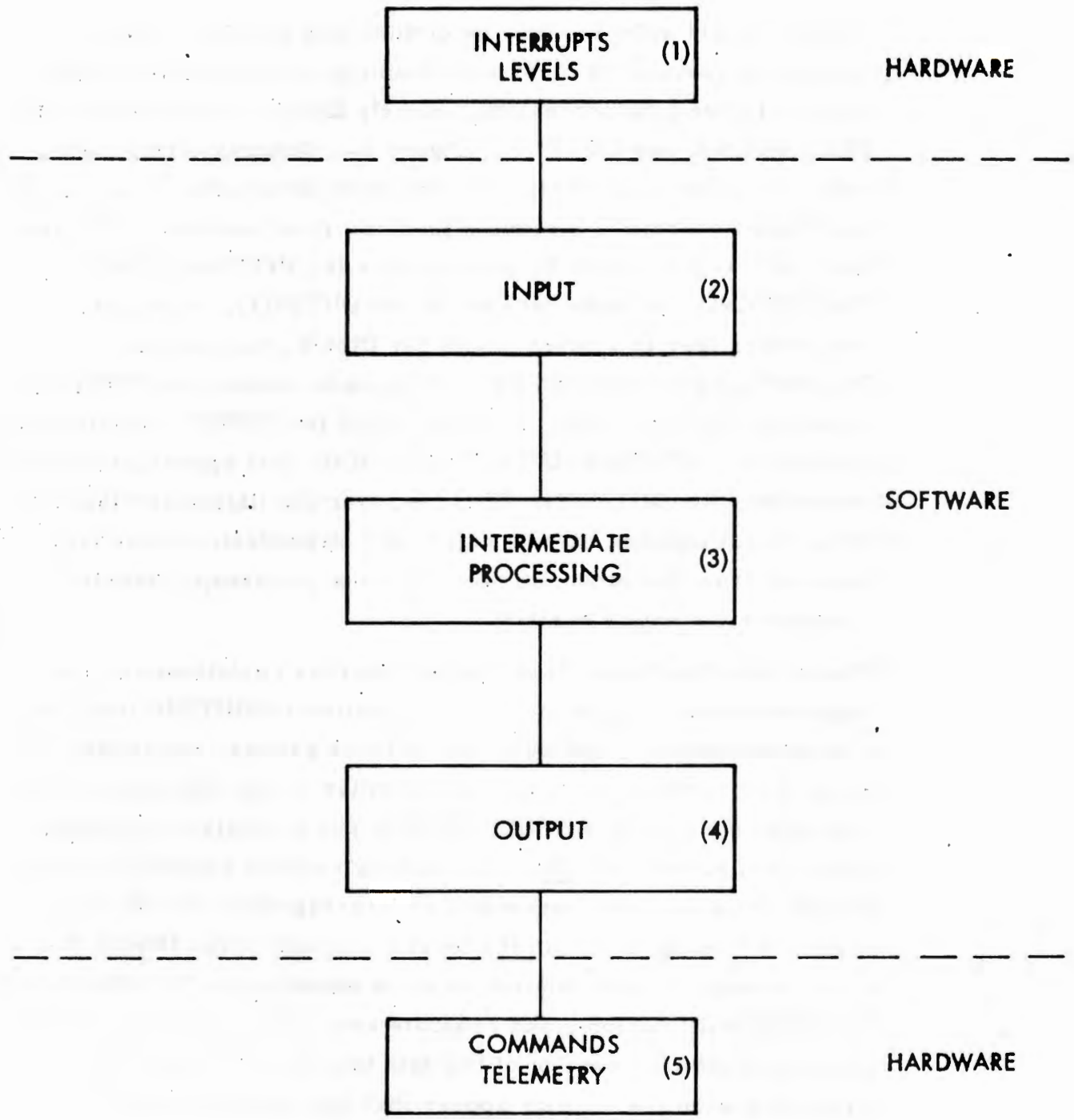


Fig. 3-1. General Program Structure

interrupt input, the ERROR routine if the input data indicates an error condition, the OUTDRV routine for buffering and actually sending the response back to AACS, and the TARMEX routine for timing information to periodically check the AACS.

The general program structure of Figure 3-1 is expanded into the routine structure of Figure 3-2. The five sections of the routine structure, denoted by the dotted lines, have the following function: 1) hardware receives levels and interrupts from other subsystems on the spacecraft (including the CCS itself), 2) software preprocesses this data as INPUT, 3) software performs INTERMEDIATE PROCESSING, 4) software generates commands and telemetry as OUTPUT, and 5) hardware generates switch closures or data patterns to other subsystems on the spacecraft (including the CCS itself). In the paragraphs and pages to follow, the software routines will be categorized into one of the sections (2), (3) or (4); but keep in mind paragraph 3.1, that even though a routine may belong to a given section, it may also perform the functions of the other two sections.

3.2 Routine Description

3.2.1 INPUT Processing

This is section (2) of Figure 3-2. All CCS processing is initiated and terminated in this section, namely in the TRAPS routine. There is no software 'executive' routine, in the strict definition. Instead, there is a hardware 'executive,' namely the hardware interrupt processor which continually polls the 32 interrupts and informs the software when one has occurred. When the software has finished performing any given function, it returns to the WAIT instruction within the TRAPS routine which halts further instruction execution (stops the software) until the hardware interrupt processor (the 'executive') finds another interrupt for the software to process. At that time the hardware again allows instructions

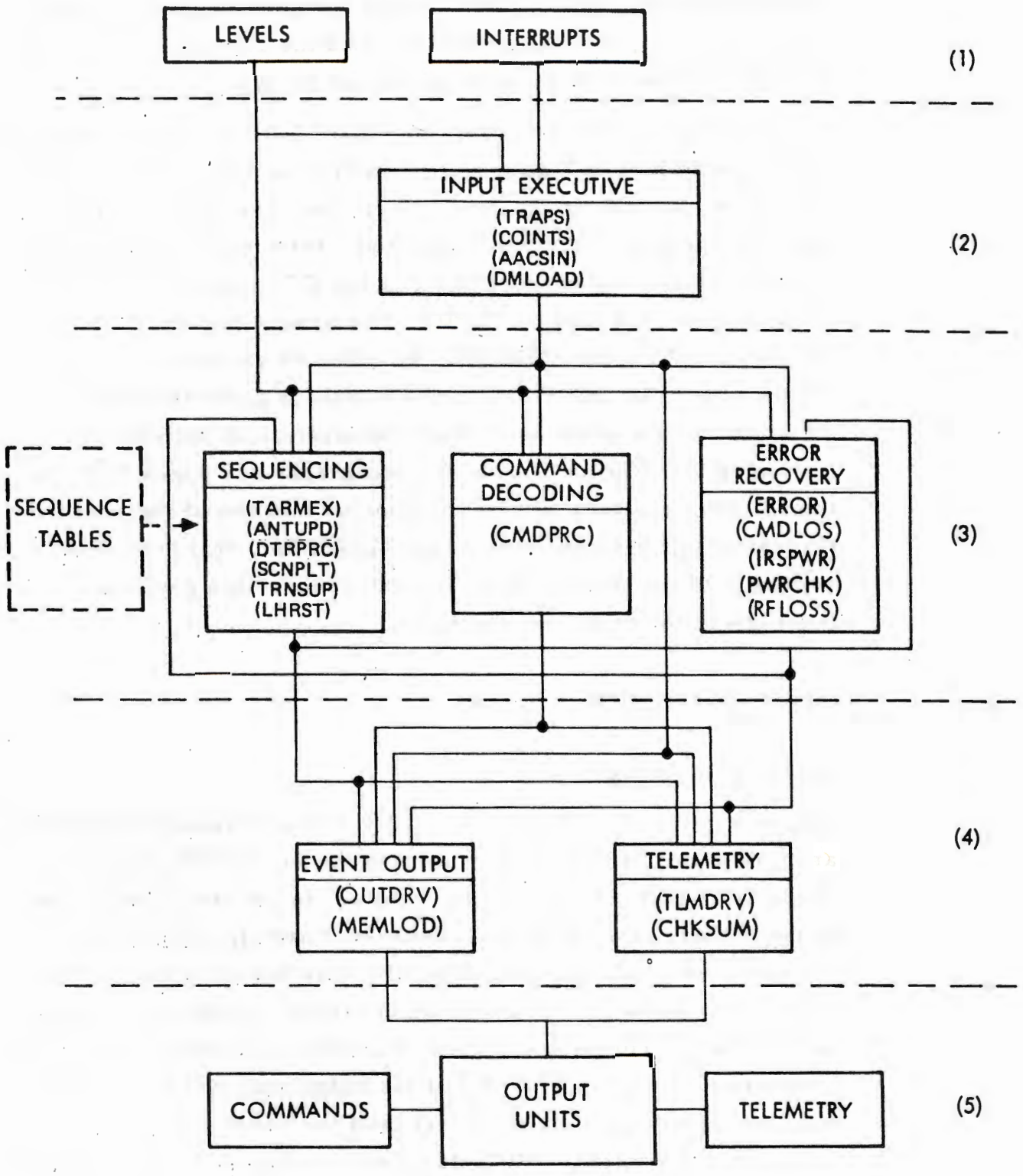


Fig. 3-2. Routine Structure

to be executed (starts the software). All routines which require interrupt information as a source of input or source of control have their origins in the TRAPS routine. On close examination of Figure 3-2 it may be seen that TRAPS is an input for every major function of the flight software. The other three routines of this section, COINTS, AACSIN, and DMLOAD perform some pre-processing of the interrupts and levels before the main processing routines are called. AACSIN responds to information from the AACS, COINTS counts the interrupts associated with timing information, and informs the main sequencing routine TARMEX when a certain number of interrupts has occurred, and DMLOAD is for emergency reloading of the CCS via an output unit in case of a memory bit error in the command decoding routine (CMDPRC).

3.2.1.1 Interrupt Assignments (TRAPS). This routine is a member of the INPUT section (see Figure 3-2). All CCS processing is initiated and terminated in this routine, namely from the location WAIT (address location 40_8). When the CCS software is not active, the PC (program counter) is at 40_8 waiting for the hardware interrupt processor to find the next interrupt. When an interrupt is found, the hardware sets the software active and tells it what interrupt has occurred. After the software is finished processing that interrupt, it goes back to WAIT and becomes inactive again. In some cases, the software will require a large amount of time to process an interrupt — too long a time to ignore possible interrupts which may occur during this interval. When this occurs, the software will have a 'breakpoint' in its execution to allow the processing of other interrupts. When this second (third, fourth, etc.) level of processing is complete, the software which was momentarily stopped will resume, and when finished, end up in the inactive (WAIT) state. For example, assume routine X is activated by an interrupt. While routine X is processing that interrupt, interrupt Y occurs. Routine X has been coded in such a manner, that midway through its execution, it will allow other interrupts to occur (breakpoint). When this point is reached, routine Y is activated since its interrupt occurred during execution of the first half of routine X.

DD

After routine Y is finished, routine X resumes at its break-point, completes its execution, and return finally to the WAIT instruction in TRAPS.

There are 32 interrupts which the hardware interrupt processor polls. When the hardware interrupt processor has control, (i.e., the software is in an inactive state or is at a break-point,) and it has found an interrupt 1 through 32, it will cause the software to execute the instruction found in memory locations 0-31 respectively. For example, if interrupt 4 is found, memory location 3 will be executed, if interrupt 28 is found, memory location 27 is executed and so on. If two or more interrupts are pending, the one with the higher order will be executed first. When an interrupt is executed, if the instruction which was in the corresponding memory location caused a change to the PC, i.e., a successful transfer instruction, further interrupts are inhibited from being executed until the software finishes processing this interrupt or reaches a break-point. If the instruction did not cause a change to the PC, interrupts are not inhibited and the next interrupt is processed, or control is returned to the routine at its break-point, or the software returns to the inactive state, depending on what was occurring before the interrupt was processed.

Most of the time, the software is coded to ignore some of the interrupts. Those interrupts which are not presently of interest are masked out so that the hardware interrupt processor will not see them. All 32 interrupts except interrupt 1 may be masked out. If they were all masked and the software is in the inactive (WAIT) state, the only interrupt which the hardware interrupt processor would see, and thus the only one which would 'wake-up' the software would be interrupt 1, the internal error interrupt. Under normal operations, however, most of the interrupts are unmasked.

It was mentioned earlier that if a routine is too long, break-points would have to be coded in to allow the processing of other interrupts. When the hardware was designed it was determined that

21

interrupts 2-8 may occur so fast as to make programming the routines difficult since many break-points would be required. To help this situation, a special feature was added, namely the 2-8 interrupt override. This feature allows the hardware interrupt processor to interrupt the software at any time, not just at specific break-points or while the software is inactive. Thus, when one of these interrupts occurs, the hardware interrupt processor stops the software and causes it to execute the corresponding memory location immediately. To prevent the interrupted software from becoming confused by this 'unexpected interrupt,' only non-transfer type instructions may be in the memory locations executed by interrupts 2-8. Thus, only this one instruction is executed, and control is then given back to the interrupted routine. The routine which was momentarily stopped, would not know the difference. The interrupt is, therefore, like a cycle-steal from a DMA controller.

3.2.1.1.1 Interrupt 1: Internal Error. This interrupt, when it occurs, will cause the instruction found in location 0 to be executed. This is the only interrupt which may not be masked out, and when it occurs, the software is immediately halted, no matter what state it is in, and is forced to execute the instruction in location 0. There are four (4) conditions which may set this interrupt: (1) Power fail. After the power supply comes into tolerance, the processor is unclamped and interrupt 1 is set, (2) Primary command error. If interrupt 16 was not processed when another interrupt 16 is received the hardware will generate an interrupt 1, (3) M/T/E error. If protected memory is attempted to be written into without the special IEX write key being set, or if the hour interrupt (number 26) was not processed 64 seconds after it was set, or if an XEC instruction attempted to execute an XEC instruction, or if the hardware bit time generator went to an illegal state, the hardware will generate an interrupt 1, and (4) when the SE releases the processor clamp, the hardware will set interrupt 1.

22

When this interrupt is processed, the ERROR routine will be called (see 3.2.2.3). The ERROR routine will attempt to diagnose the cause of the interrupt and respond accordingly. Typically, the response will include stopping all sequencing, resetting of internal software indicators, possibly generating a few commands to put the spacecraft in a safe state, and then waiting for ground commanding.

3.2.1.1.2 Interrupt 2: Internal Interrupt F — Not Used

3.2.1.1.3 Interrupt 3: 100 PPS. This interrupt, when it occurs, will cause the instruction found in location 2 to be executed. It is set every 10 milliseconds by the hardware and if unmasked, will cause the location PPCCNT to be decremented once for each interrupt processed. Since this interrupt is under the special 2-8 interrupt override control (see para. 3.2.1.1), which is usually set, it will be processed immediately upon its occurrence and not bother the software. If the location PPCCNT is decremented to zero by this interrupt, the hardware will automatically set interrupt 10 (see 3.2.1.1.10).

The 10 millisecond time interval is derived from the 2.4 KHZ power supply, and therefore is 'in phase' with the spacecraft master clock. This interrupt, in conjunction with the 1 PPS (seconds) and 1 PPH (hours) interrupts, is the main source of timing for the CCS in performing its sequencing function (see 3.2.2.2).

3.2.1.1.4 Interrupt 4: DTR Forward TIC. This interrupt, when it occurs, will cause the instruction found in location 3 to be executed. It is set when the DTR (digital tape recorder) has advanced 3.8 inches and its electronic direction indicator is set to the forward direction. The location DTRCNT will be decremented once for each interrupt if the interrupt is unmasked, which should always be the case. Since this interrupt is under the special

23

2-8 interrupt override control, which is usually set, it will be processed immediately upon its occurrence and not bother the software. If the location DTRCNT is decremented to zero by this interrupt, the hardware will automatically set interrupt 10. The data to be decremented by this interrupt is calculated by the routine DTRPRC in the sequencing function of the flight software (see para 3.2.2.2.3).

3.2.1.1.5 Interrupt 5: DTR Reverse TIC. This interrupt, when it occurs, will cause the instruction found in location 4 to be executed. It is set when the DTR (digital tape recorder) has advanced 3.8 inches and its electronic direction indicator is set to the reverse direction. The location DTRCNT will be incremented once for each interrupt if the interrupt is unmasked, which should always be the case. Since this interrupt is under the special 2-8 interrupt override control (see para. 3.2.1.1), which is usually set, it will be processed immediately upon its occurrence and not bother the flight software. If the location DTRCNT is incremented to zero by this interrupt, the hardware will automatically set interrupt 10.

The data to be incremented by this interrupt is calculated by the routine DTRPRC in the sequencing function of the flight software (see para. 3.2.2.2.3).

3.2.1.1.6 Interrupt 6: Internal Interrupt E – Not Used.

3.2.1.1.7 Interrupt 7: Internal Interrupt D – Not Used.

3.2.1.1.8 Interrupt 8: 1 PPS. This interrupt when it occurs, will cause the instruction found in location 7 to be executed. It is set every second by the hardware and if unmasked, will cause the location PPSCNT to be decremented. Since this interrupt is under the special 2-8 interrupt override control (see para. 3.2.1.1), which

is usually set, it will be processed immediately upon its occurrence and not bother the software. If the location PPSCNT is decremented to zero, the hardware will automatically set interrupt 10.

The 1-second time interval is derived from the 2.4 KHZ power supply, and therefore is "in phase" with the spacecraft master clock. This interrupt, in conjunction with the 100 PPS (centi-seconds) and 1 PPH (hours) interrupts, is the main source of timing for the CCS in performing its sequencing function.

3.2.1.1.9 Interrupt 9: External Level 1. This interrupt, when it occurs, will cause the instruction found in location 10_8 to be executed. The interrupt will be processed if unmasked (always) and the software is in an inactive state, or is at a break-point (see para. 3.2.1.1). The interrupt is set whenever the hardware detects a change of state of the A, B, C, or D (4 of 6) power code levels from the AACS. When processed, the interrupt will cause a transfer to the AACSIN routine (see 3.2.1.3).

The 6 power code levels from the AACS are a means of communicating information from the AACS to the CCS. They are used to indicate the status of the AACS (locked on the SUN or CANOPUS), to indicate errors in any coded commands the CCS has sent to AACS, and to request a device to be turned on or off by the CCS, since the AACS does not control the power subsystem.

3.2.1.1.10 Interrupt 10: Internal Counter Zero. This interrupt, when it occurs, will cause the instruction found in location 11_8 to be executed. The interrupt will be processed if unmasked and the software is in an inactive state, or is at a break-point. The interrupt is set whenever the hardware detects a zero-result during an interrupt cycle such as when interrupts 2-8 are processed and a counter reaches zero. When processed, the interrupt will cause a transfer to the COINTS routine (see para. 3.2.1.2).

This interrupt, used in conjunction with interrupts 2-8, allows counters to be incremented/decremented by fast occurring interrupts while the flight software is active, performing some other function. When the counter reaches zero, this interrupt is set, thereby informing the flight software that one of the counters has reached zero, and that it is time to perform an additional function. It is the duty of the routine COINTS to determine what counter reached zero and then call the appropriate routine to process the associated function.

3.2.1.1.11 Interrupt 11: Internal CHECKSUM Request. This interrupt, when it occurs, will cause the instruction found in location 12_8 to be executed. The interrupt will be processed if unmasked (always) and the software is in an inactive state, or is at a break-point. The interrupt is set by a processor executing a special IEX instruction. When processed, the interrupt will cause a transfer to the CHKSUM routine (see para. 3.2.3.4).

3.2.1.1.12 Interrupt 12: FDS Engineering Bit Sync. Not Used.

3.2.1.1.13 Interrupt 13: Output Unit 1 Telemetry Available. This interrupt, when it occurs, will cause the instruction found in location 14_8 to be executed. The interrupt will be processed if unmasked and the software is in an inactive state, or is at a break-point. The interrupt is set by the hardware whenever the telemetry register within output unit 1 is available for loading. When processed, the interrupt will cause a transfer to the TLMDRV routine (see para. 3.2.3.3).

This interrupt is unmasked (allowed to occur) whenever the TLMDRV routine has data ready to be telemetered via the output units. When finished sending the information out, the interrupt is masked.

-26

3.2.1.1.14 Interrupt 14: Output Unit 2 Telemetry Available. This interrupt, when it occurs, will cause the instruction found in location 15_8 to be executed. The interrupt will be processed if unmasked and the software is in an inactive state, or is a break-point. The interrupt is set by the hardware whenever the telemetry register within Output Unit 2 is available for loading. When processed, the interrupt will cause a transfer to the TLMDRV routine (see para. 3.2.3.3).

This interrupt is unmasked (allowed to occur) whenever the TLMDRV routine has data ready to be telemetered via the output units. When finished sending the information out, the interrupt is masked.

3.2.1.1.15 Interrupt 15: Primary Bit Sync Lock. This interrupt, when it occurs, will cause the instruction found in location 16_8 to be executed. The interrupt will be processed if unmasked (always) and the software is in an inactive state, or is at a break-point. The interrupt is set when the Command Demodulation Unit (CDU) goes into or out of bit sync lock. When processed, the interrupt will cause a transfer to the CMDPRC routine (see para. 3.2.2.1).

This interrupt is used to initialize the command decoding routine and is effectively OR'd with interrupt 18 (see para. 3.2.1.1.18). The bit sync lock signal from CDU A goes to interrupt 15 of processor A and interrupt 18 of processor B, and the bit sync lock signal from CDU B goes to interrupt 18 of processor A and interrupt 15 of processor B.

3.2.1.1.16 Interrupt 16: Primary Bit Sync. This interrupt, when it occurs, will cause the instruction found in location 17_8 to be executed. The interrupt will be processed if unmasked (always) and the software is in an inactive state, or is at a break-point. The interrupt is set when the CDU has a command data bit ready to be read from the input level matrix. When processed, the interrupt will cause a transfer to the CMDPRC routine (see para. 3.2.2.1).

2.7

This interrupt is used to inform the command decoding routine that the next command bit is ready for reading. This interrupt is effectively OR'ed with interrupt 19 (see para. 3.2.1.1.19). The bit sync signal from CDU A goes to interrupt 16 of processor A and interrupt 19 of processor B, and the bit sync signal from CDU B goes to interrupt 19 of processor A and interrupt 16 of processor B.

3.2.1.1.17 Interrupt 17: Secondary Command Error. This interrupt, when it occurs, will cause the instruction found in location 20_8 to be executed. The interrupt will be processed if unmasked (always) the software is in an inactive state, or is at a break-point. The interrupt is set by the hardware when interrupt 19 (secondary bit sync) is not processed by the time another interrupt 19 is received. When processed, the interrupt will cause a transfer to the ERROR routine (see para. 3.2.2.3).

3.2.1.1.18 Interrupt 18: Secondary Bit Sync Lock. This interrupt, when it occurs, will cause the instruction found in location 21_8 to be executed. The interrupt will be processed if unmasked (always) and the software is in an inactive state, or is at a break-point. The interrupt is set when the CDU goes into or goes out of bit-sync lock. When processed, the interrupt will cause a transfer to the CMDPRC routine (see para. 3.2.2.1).

This interrupt is used to initialize the command decoding routine and is effectively OR'ed with interrupt 15 (see 3.2.1.1.15). The bit-sync lock signal from CDU A goes to interrupt 15 of processor A and interrupt 18 of processor B, and the bit-sync lock signal from CDU B goes to interrupt 18 of processor A and interrupt 15 of processor B.

3.2.1.1.19 Interrupt 19: Secondary Bit Sync. This interrupt, when it occurs, will cause the instruction found in location 22_8 to be executed. The interrupt will be processed if unmasked (always) and the

software is in an inactive state, or is at a break-point. The interrupt is set when the CDU has a command data bit ready to read from the input level matrix. When processed, the interrupt will cause a transfer to the CMDPRC routine (see para. 3.2.2.1).

This interrupt is used to inform the command decoding routine that the next command bit is ready for reading. This interrupt is effectively OR'ed with interrupt 16 (see para. 3.2.1.1.16). The bit-sync signal from CDU A goes to interrupt 16 of processor A and interrupt 19 of processor B, and the bit-sync signal from CDU B goes to interrupt 19 of processor A and interrupt 16 of processor B.

3.2.1.1.20 Interrupt 20: Output Unit Reject. Not Used.

3.2.1.1.21 Interrupt 21: Output Unit Demand Read. This interrupt, when it occurs, will cause the instruction found in location 24_8 to be executed. The interrupt will be processed if unmasked (always) and the software is in an inactive state, or is at a break-point. The interrupt is set by an output unit when it has been instructed to do so by the other processor. When processed, the interrupt will cause a transfer to the DMLOAD routine (see para. 3.2.1.4). This interrupt is used for emergency reloading of a memory in case of a failure in the CMDPRC routine.

3.2.1.1.22 Interrupt 22: Output Unit Initiate Read. This interrupt, when it occurs, will cause the instruction found in location 25_8 to be executed. The interrupt will be processed if unmasked, which is most of the time except for certain conditions, and the software is in an inactive state or is at a breakpoint. The interrupt is set by an output unit when it has been instructed to do so by a processor. When processed, the interrupt will cause a transfer to the OUTDRV routine (see para. 3.2.3.1). This interrupt is used for inter-processor communication.

29

3.2.1.1.23 Interrupt 23: Output Unit(s) Available. This interrupt, when it occurs, will cause the instruction found in location 26_8 to be executed. The interrupt will be processed if unmasked and the software is in an inactive state, or is at a break-point. The interrupt is set by the hardware whenever an output unit is available for accepting commands. When processed, the interrupt will cause a transfer to the OUTDRV routine (see para. 3.2.3.1). This interrupt is unmasked (allowed to occur) whenever the OUTDRV routine has data (commands) for the output unit to execute. When finished sending commands out, the interrupt is masked.

3.2.1.1.24 Interrupt 24: FDS ISS-WA Frame Start. This interrupt, when it occurs, will cause the instruction in location 27_8 to be executed. The interrupt will be processed if unmasked and the software is in an inactive state, or is at a breakpoint. The interrupt is set by the hardware on the leading edge of a 96 second square wave signal from the FDS. When processed, the interrupt will cause a transfer to the routine COINTS (see para. 3.2.1.2).

This interrupt is unmasked whenever a sequencing function requires the use of this timing signal. The software effectively OR's this signal with interrupt 25 (see para. 3.2.1.1.25), thus generating a periodic interrupt on 48 second intervals. Its main use is in driving science sequences which are designed to work in conjunction with the basic 48 second ISS frame shutter rate.

3.2.1.1.25 Interrupt 25: FDS ISS-NA Frame Start. This interrupt, when it occurs, will cause the instruction in location 30_8 to be executed. The interrupt will be processed if unmasked and the software is in an inactive state, or is at a breakpoint. The interrupt is set by the hardware on the trailing edge of a 96 second square wave signal from the FDS. When processed, the interrupt will cause a transfer to the routine COINTS (see para. 3.2.1.2).

This interrupt is unmasked whenever a sequencing function requires the use of this timing signal. The software effectively

30

OR's this signal with interrupt 24 (see para. 3.2.1.1.24) thus generating a periodic interrupt on 48 second intervals. Its main use is in driving science sequences which are designed to work in conjunction with the basic 48 second ISS frame shutter rate.

3.2.1.1.26 Interrupt 26: 1 PPH. This interrupt, when it occurs, will cause the instruction in location 31_8 to be executed. The interrupt will be processed if unmasked (always) and the software is in an inactive state or is at a breakpoint. The interrupt is set every hour by the hardware. When processed, the interrupt will cause a transfer to the routine COINTS (see para. 3.2.1.2).

The 1 hour time interval is derived from the 2.4 KHZ power supply, and therefore is "in phase" with the spacecraft master clock. This interrupt, in conjunction with the 100 PPS (centi-seconds) and 1 PPS (seconds) interrupts, is the main source of timing for the CCS in performing its sequencing function.

3.2.1.1.27 Interrupt 27: External Level 2. This interrupt, when it occurs, will cause the instruction found in location 32_8 to be executed. The interrupt will be processed if unmasked (always) and the software is in an inactive state, or is at a break-point. The interrupt is set whenever the hardware detects a change of state of either the E or F (2 of 6) power code levels from the AACS, or the pre-separation input from the CENTAUR launch vehicle, or the power low volt indicator from PWR. When processed, the interrupt will cause a transfer to the AACSIN routine (see para. 3.2.1.3).

3.2.1.1.28 Interrupt 28: External Level 3. This interrupt, when it occurs, will cause the instruction found in location 33_8 to be executed. The interrupt will be processed if unmasked (always) and the software is in an inactive state, or is at a break-point. The interrupt is set whenever the hardware detects a change of state of any of the 4 level-inputs from the radio frequency subsystem (RFS). When processed, the interrupt will cause a transfer to the RFLOSS routine (see para. 3.2.2.3.5).

The four levels from RFS indicate possible failure conditions within that subsystem. The RFLOSS routine will read these levels and respond accordingly.

3.2.1.1.29 Interrupt 29: External Level 4. This interrupt, when it occurs, will cause the instruction found in location 34_8 to be executed. The interrupt will be processed if unmasked (always) and the software is in an inactive state or is at a break-point. The interrupt is set whenever the hardware detects a change of state of the MIRIS primary power supply.

When processed, the interrupt will cause a transfer to the IRSPWR routine (see para. 3.2.2.3.3).

3.2.1.1.30 Interrupt 30: External Level 5. Not Used.

3.2.1.1.31 Interrupt 31: DTR EOT/BOT. This interrupt, when it occurs, will cause the instruction found in location 36_8 to be executed. The interrupt will be processed if unmasked (always) and the software is in an inactive state, or is at a break-point. The interrupt is set whenever the digital tape recorder senses an End Of Tape sensor or Beginning Of Tape sensor. When processed, the interrupt will cause a transfer to the DTRPRC routine (see para. 3.2.2.2.3).

The DTRPRC routine will initialize its TIC counters according to which end of the tape the DSS is residing at.

3.2.1.1.32 Interrupt 32: Internal Self-Test Request. This interrupt, when it occurs, will cause the instruction found in location 37_8 to be executed. The interrupt will be processed if unmasked (always) and the software is in an inactive state or is at a break-point. The interrupt is set by a processor executing a special IEX instruction. When processed, the interrupt will cause a transfer to the COINTS routine (see para. 3.2.1.2).

This interrupt is set by the OUTDRV or TLMDRV routine whenever they need access to an output unit, or is set by COINTS once each second as long as there is data in the output buffers of the OUTDRV or TLMDRV routines. This interrupt will cause the

routine COINTS to initiate a processor self-test, and if successful, allow communications to the output units (see para 3.2.2.3.1.2) for an explanation of processor self-test.

3.2.1.2 Counting Interrupt Support (COINTS). This routine is activated by the internal interrupt (10), FDS ISS frame interrupts (24 and 25), the CCS one hour interrupt (26), and the self-test interrupt (32).

Internal interrupt 10 occurs whenever a count-word associated with interrupts 2 through 8 attains a zero condition. This routine in responding to interrupt 10, determines which of the above interrupt count-words is (are) at zero and takes appropriate action as follows:

Interrupt (Count-Word at Zero)	Action
Internal F (2)	Not used.
100 PPS (3)	Jump to 100 PPS entry point in sequence support routine.
DSS Tape Increment (4)	Jump to DSS routine.
DSS Tape Decrement (5)	Jump to DSS routine.
Internal E (6)	Not used.
Internal D (7)	Not used.
1 PPS (8)	Jump to 1 PPS entry point in sequence support routine.

The FDS ISS frame start interrupts (24 and 25) are processed as one interrupt by this routine. This processing consists of decrementing a count-word and jumping to the ISS frame start entry point in the sequence support routine. Although FDS frame starts normally occur only every 48 seconds, they may occur much closer together when the FDS is first turned on. For this reason, re-entry into this routine once it is activated is prevented until all processing has been completed.

The CCS 1 PPH interrupt is processed by incrementing an absolute hour clock word and then jumping to the 1 PPH entry point in the sequence support routine. After the sequence support routine is finished processing the 1 PPH input, program control is returned to this routine to output the following telemetry:

1. Uplink Command Counter
2. CCS Output Event Counter
3. CDU Lock-change Counter
4. Master Automatic Counter
5. DTR TIC Counter

After this telemetry, the AACS Heartbeat failure counter and Scan abort counter are reset, Finally, this routine determines if a command has been received by the CCS within a specified period of time. If it has not, a jump to the command loss routine occurs.

Processing of the self-test interrupt causes this routine to verify that command, AACS power code, or telemetry information is ready for output, and if it is, initiate the self-test subroutine, and set up software to automatically restart the self-test subroutine each second until data is no longer available for output. Figure 3-3 illustrates the flow diagram for this routine.

3.2.1.3 AACS Power Code Processing (AACSIN). This routine responds to any change associated with external levels 1 and 2 (Level matrix rows 3 and 4). Because the Power Low Voltage input is also assigned to Level 2, this routine must differentiate between a power low voltage condition and an AACS power code.

Entry to this routine also occurs whenever a self-test is initiated for any reason (command or telemetry output), or after the 4-5 second delay at the beginning of the power recovery routine has timed out.

3.4

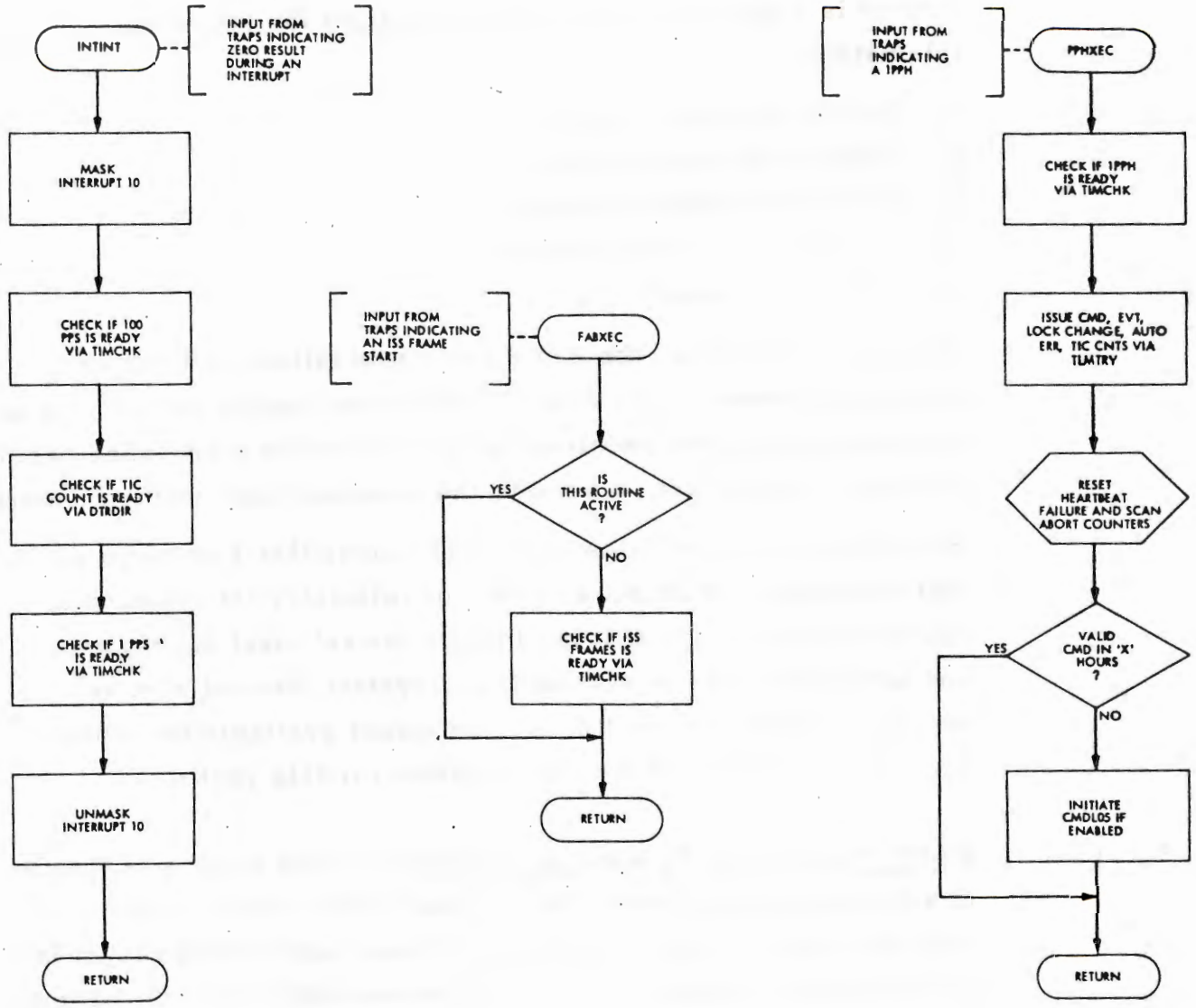


Fig. 3-3. Flow Diagram of the Routine COINTS (Sheet 1 of 2)

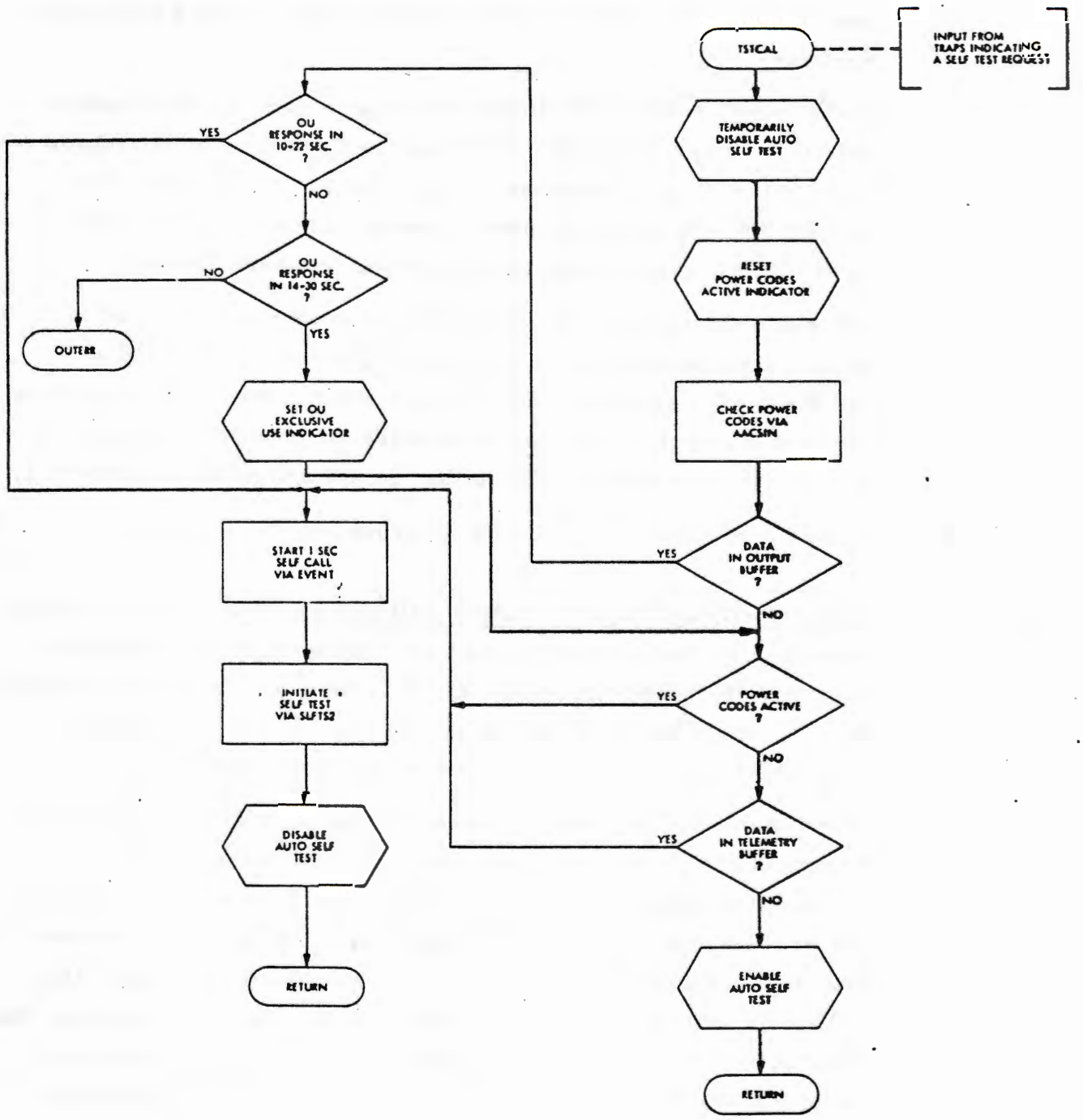


Fig. 3-3. Flow Diagram of the Routine COINTS (Sheet 2 of 2)

26

When processing an AACS power code, this routine determines if it is different from the previously processed power code and exits if it is not. If it is different, the routine waits 3 to 4 milliseconds and verifies that the power code has not changed. If the power code has not changed, this routine then determines the type of power code (functional or informational) and initiates the proper response.

If the power code is functional, the power code itself is used to generate the proper coded command to the power subsystem to initiate the proper response. (See MJS77-3-290, Table 4b.) In addition, the received power code is echoed back to AACS after the associated command to power has been issued.

If the power code is informational, the power code is used to determine what response, if any, is required by the CCS and initiates that response. See Table 3-1 for power code responses. In most instances a received and acted-on power code requires an echo back to AACS. Those that do not are noted in Table 3-1.

Figure 3-5 illustrates the flow diagram for this routine.

3.2.1.4 Direct Memory Load via Output Unit (DMLOAD). This is a stand alone routine which will be used only in the case that the main command decoding routine (CMDPRC) fails due to a memory fault. It uses the DEMAND READ interrupt in conjunction with the EMLOAD routine which is loaded in the other processor.

When unable to command a processor and a memory fault is suspected as the cause, the good processor is loaded with the EMLOAD routine. After determining that it is safe to do so, the bad processor then must be unclamped. The EMLOAD routine then sends via 'his' output unit two header messages which the DMLOAD routine checks. If the two header messages are ok, the DMLOAD routine will mask all processing (including command decoding) and will only process data from the other processor.

27

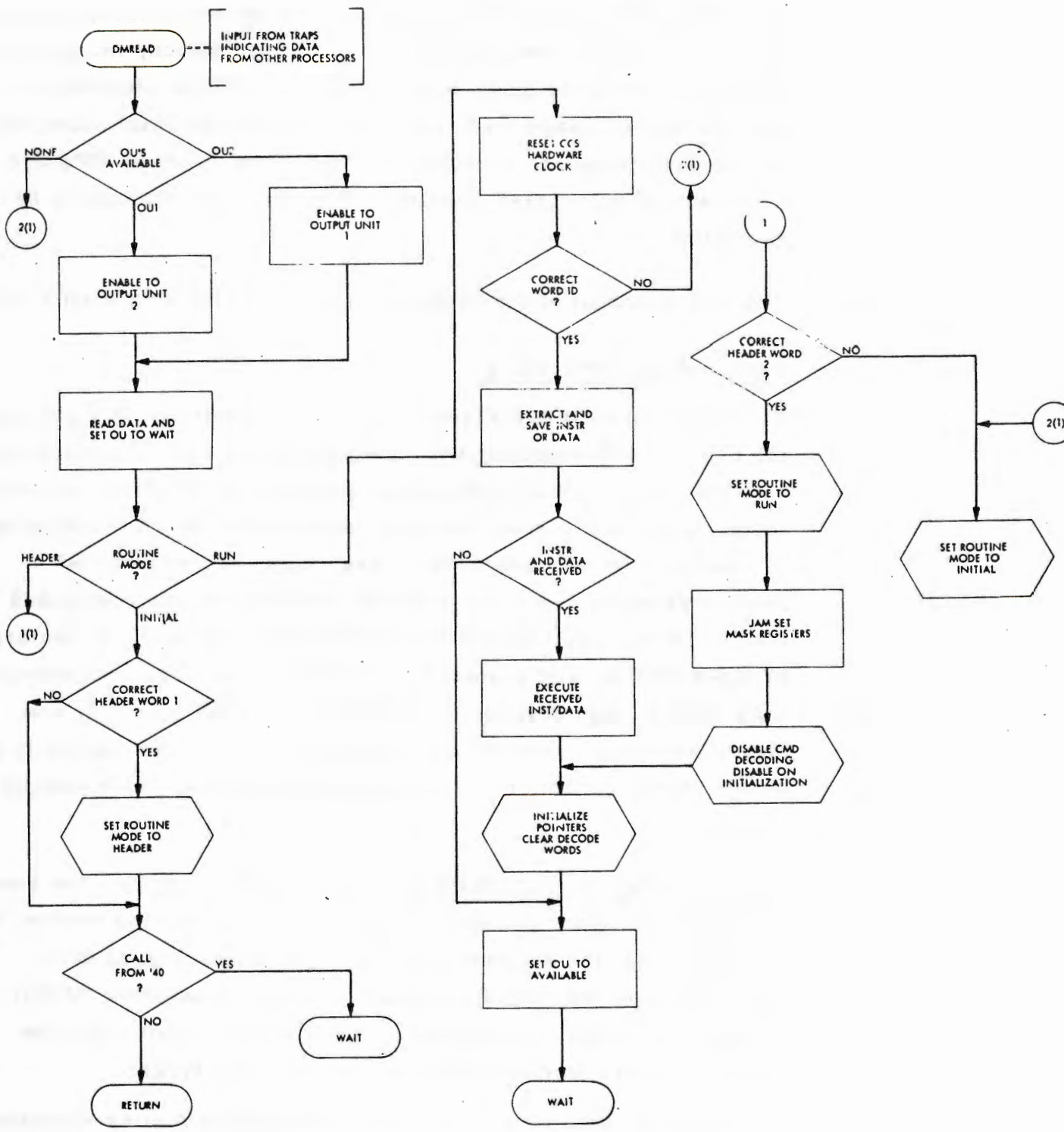


Fig. 3-4. Flow Diagram of the Routine DMLOAD

38

The DMLOAD routine will expect data following the two header words to come in groups of four messages. The nine (9) most significant bits of each message is an ID, with the 9 least significant bits representing the following information: message 1, LSH of an instruction, message 2, MSH of an instruction, message 3, LSH of accumulator data, and message 4, MSH of accumulator data. After message 4 is received, the gathered instruction and accumulator data are executed. If any of the message ID's are incorrect, the DMLOAD routine will abort, and will have to be restarted.

The flow diagram of the routine DMLOAD is shown in Figure 3-5.

3.2.2 Intermediate Processing

This is Section (3) and Figure 3-2. The majority of CCS processing occurs in this section with the three main functions of the CCS represented: Command Decoding: processing information from the ground and initiating CC's, DC's, AC's, SC's, PC's and uplink blocks as directed, Sequencing: executing preprogrammed sequences as loaded by the ground, including single commands (CC's, DC's), synthesized commands (AC's, SC's) CCS memory changes (PC's), and sequence algorithms (automatic sun sensor bias update, tape positioning, MOSAIC's, SCAN's, etc.), and Error Recovery· monitoring spacecraft levels and responding in an appropriate manner if an unexpected and undesirable change occurs.

3.2.2.1 Command Decoding (CMDPRC). This routine performs the command decoding function of the intermediate processing section. It is responsible for accepting and decoding serial digital data received from the Modulation-Demodulation Subsystem (MDS). These data identify commands sent to the spacecraft from the ground. These commands are of the following types:

1. Single commands to be executed immediately after reception (base commands).

Table 3-1. CCS Response to AACS Power Codes (Sheet 1 of 5)

Power Code	Name	Response
PC000	No-OP	None. Do not echo PC000 power code.
PC001	Dummy power command	Unused command to power.
PC002	Dummy power command	Unused command to power.
PC003	Dummy power command	Unused command to power.
PC004	$\overline{\text{ADET}}$ not acquired ($\overline{\text{ADET}}$)	• Enable restart after power fail. Enable 2ER switch 18 hours (or other time set during encounter) after CA or $\overline{\text{CA}}$. Start IRIS flash-off heater sequence (disabled during encounter). Issue Low Gain antenna command 2ER. Wait 15 seconds. Inhibit scan algorithm. Slew platform to neutral position: AC7SPM 2, 1, 1, 049, 120 and AC7SPK 1, 1, 1, 096, 127 for S/C 31 and AC7SPM 1, 12, 1, 124, 157 and AC7SPK 2, 12, 1, 165, 123 for S/C 32. Wait 481 seconds. Slew platform to safe position: AC7SPM 2, 1, 1, 17, 99 and AC7SPK 1, 1, 1, 118, 147 for S/C 31 and AC7SPM 1, 12, 1, 124, 157 and AC7SPK 2, 12, 1, 165, 123, for S/C 32. • Send SC06BB command to FDS 38 sec after every ISS frame time until PC005 received (disabled during encounter). • This power code will be preceded by PC064 (omen).
PC005	ADET acquired (ADET)	Wait 497 seconds. Slew platform to neutral position: AC7SPM 1, 1, 1, 096, 127 and AC7SPK 2, 1, 1, 049, 120 for S/C 31 and AC7SPM 1, 12, 1, 124, 157 and AC7SPK 2, 12, 165, 123 for S/C 32. Wait 480 seconds. Enable scan algorithm.
PC006	Star not required	Enable restart after power fail. Enable 2ER switch 18 hours (or other time set during encounter) after CA or $\overline{\text{CA}}$. Start IRIS flash-off heater sequence (disabled during encounter). Increment CA counter. Issue 2E if enabled. Wait 18 hours (or other time set during encounter) for ground response. If no response, issue 2ER. • This power code will be preceded by PC064 (omen).
PC007	Star acquired	Exit star-not-acquired routine if received within 18 hours (or other time set during encounter) of PC006. If later, issue 2E if enabled, then wait 18 hours (or other time set during encounter) for ground response. If no ground response, issue 2ER.

40

Table 3-1. CCS Response to AACS Power Codes (Sheet 2 of 5)

Power Code	Name	Response
PC010	ISO valve BR 1 power off	CC7V1P, CC7V1RP, CC7V1S, or CC7V1RS to PWR.*
PC011	ISO valve BR 1 power on	CC7V1P, CC7V1RP, CC7V1S, or CC7V1RS to PWR.*
PC012	Dummy power command	Unused command to power.
PC013	Dummy power command	Unused command to power.
PC014	HYBIC 1 off/HYBIC 2 on	Wait 2.68 seconds and issue CC7H1P, CC7H1RP, CC7H1S, or CC7H1RS to PWR* then CC7H2P, CC7H2RP, CC7H2S, or CC7H2RS to PWR,* then CC6AB01010, then CC6AD3200.
PC015	HYBIC 2 off/HYBIC 1 on	Wait 2.68 seconds and issue CC7H2P, CC7H2RP, CC7H2S, or CC7H2RS to PWR* then CC7H1P, CC7H1RP, CC7H1S, or CC7H1RS to PWR,* then CC6AB01010, then CC6AD3200.
PC016	HYBIC 1 off/HYBIC 2 on	Same as PC014.
PC017	HYBIC 2 off/HYBIC 1 on	Same as PC015.
PC020	ISO valve BR2 power off	CC7V2P, CC7V2RP, CC7V2S, or CC7V2RS to PWR.*
PC021	ISO valve BR2 power on	CC7V2P, CC7V2RP, CC7V2S, or CC7V2RS to PWR.*
PC022	CST sun shutters off	CC7TSP, CC7TSRP, CC7TSS, or CC7TSRS to PWR.*
PC023	CST sun shutters on	CC7TSP, CC7TSRP, CC7TSS, or CC7TSRS to PWR.*
PC024	Power supply fail	Store occurrence in CCS memory. This power code preceded by PC064 (omen).
PC025	Memory refresh fail	Store occurrence in CCS memory. This power code preceded by PC064 (omen).
PC026	IDET not acquired. ($\overline{\text{IDET}}$)	• Enable restart after power fail. Enable 2ER switch 18 hours (or other time set during encounter) after CA or $\overline{\text{CA}}$. Start IRIS flash-off heater sequence (disabled during encounter). Issue Low Gain antenna command 2ER. Wait 15 seconds. Inhibit scan algorithm. Slew platform to neutral position: AC7SPM 2, 1, 1, 049, 120 and AC7SPK 1, 1, 1, 096, 127 for S/C 31 and AC7SPM 1, 12, 1, 124, 157 and AC7SPK 2, 12, 1, 165, 123 for S/C 32. Wait 481 seconds. Slew

*Which command of set to be issued depends on the power relay state table in the CCS.

41

Table 3-1. CCS Response to AACS Power Codes (Sheet 3 of 5)

Power Code	Name	Response
		platform to safe position: AC7SPM 2, 1, 1, 17, 99 and AC7SPK 1, 1, 1, 118, 147 for S/C 31 and AC7SPM 1, 12, 1, 124, 157 and AC7SPK 2, 12, 1, 165, 123 for S/C 32. • Send SC06BB command to FDS 38 sec after every ISS frame time until PC027 received (disabled during encounter). • Wait 570 seconds. Issue sun search command 7SS1. Start 360° yaw turn: Wait 2160 seconds. If PC027 not received, start 38° pitch turn. Wait 360 seconds. Repeat each turn 4 times or until PC027 received. If 10 turns done with no PC027, disable power codes and power low voltage response to force FCP switch. Repeat pitch/yaw turns. • This power code preceded by PC064 (omen).
PC027	IDET acquired (IDET)	Exit IDET not acquired routine.
PC030	Dummy power command	Unused command to power.
PC031	Dummy power command	Unused command to power.
PC032	Star tracker 1 off	CC7T1P, CC7T1RP, CC7T1S, or CC7T1RS to PWR.*
PC033	Star tracker 1 on	CC7T1P, CC7T1RP, CC7T1S, or CC7T1RS to PWR.*
PC034	FCP self-test passed	Increment self-test-passed counter.
PC035	FCP self-test passed	Increment self-test-passed counter.
PC036	FCP self-test passed	Increment self-test-passed counter.
PC037	AACS heartbeat	Check for reception of one or more power codes every 10 seconds. If no reception, send self-test 7ST, non-prime FCP off 77PR, MAM command disable 77XR, self-test 7ST, MAM off 77MRP and 77MS commands. If two or more failures in one hour, wait 5 seconds and switch HYBIC/FCP. Reconfigure FDS. Enable power codes and power low voltage response. Wait 15-19 seconds. Start FCP initilization sequence. Do not echo PC037 power code.
PC040	Dummy power command	Unused command to power.
PC041	Dummy power command	Unused command to power.

*Which command of set to be issued depends on the power relay state table in the CCS.

92

Table 3-1. CCS Response to AACS Power Codes (Sheet 4 of 5)

Power Code	Name	Response
PC042	Canopus Star Tracker 2 off	CC7T2P, CC7T2RP, CC7T2S, or CC7T2RS to PWR.*
PC043	Canopus Star Tracker 2 on	CC7T2P, CC7T2RP, CC7T2S, or CC7T2RS to PWR.*
PC044	TCM burn abort	Terminate maneuver and start unwind turns.
PC045	MM/PM commanded turn complete	Compare with CCS timer started at beginning of turn. If within TBD time limits, continue maneuver, otherwise abort.
PC046	Sequence check spare	
PC047	Sequence check spare	
PC050	Dummy power command	Unused command to power.
PC051	Dummy power command	Unused command to power.
PC052	Gyro A off	CC7GAP, CC7GARP, CC7GAS, or CC7GARS to PWR.*
PC053	Gyro A on	CC7GAP, CC7GARP, CC7GAS, or CC7GARS to PWR.*
PC054	Scan slew abort	If TBD aborts in one hour, issue power code to switch HYPIC's. If ≤ TBD aborts, wait 15 seconds. Inhibit scan algorithm. Turn off PPS for S/C 31 only. Slew platform to neutral position: AC7SPM 2, 1, 1, 049, 120, and AC7SPK 1, 1, 1, 096, 127 for S/C 31 and AC7SPM 1, 12, 1, 124, 157 and AC7SPK 2, 12, 1, 165, 123 for S/C 32. Wait 480 seconds. Enable scan algorithm. If more than TBD aborts disable scan slew abort. Wait 15 seconds. Inhibit scan algorithm. Slew platform to neutral position: AC7SPM 2, 1, 1, 049, 120 and AC7SPK 1, 1, 1, 096, 127 for S/C 31 and AC7SPM 1, 12, 1, 124, 157 and AC7SPK 2, 12, 1, 165, 123 for S/C 32. Wait 481 seconds. Slew platform to safe position: AC7SPM 2, 1, 1, 17, 99 and AC7SPK 1, 1, 1, 118, 147 for S/C 31 and AC7SPM 1, 12, 1, 124, 157 and AC7SPK 2, 12, 1, 165, 123 for S/C 32. This power code preceded by PC064 (omen).
PC055	Command parity error	Echo only. This power code preceded by PC064 (omen).

*Which command of set to be issued depends on the power relay state table in the CCS.

Table 3-1. CCS Response to AACS Power Codes (Sheet 5 of 5)

Power Code	Name	Response
PC056	Bad/no echo response	Switch CCS processor "prime" indicator and echo. This power code preceded by PC064 (omen), only if prior PC056 not echoed.
PC057	Command sequence error	Echo only. This power code preceded by PC064 (omen).
PC060	Heater A on	CC7BAP, CC7BARP, CC7BAS, or CC7BARS to PWR.*
PC061	Heater A off	CC7BAP, CC7BARP, CC7BAS, or CC7BARS to PWR.*
PC062	Gyro B off	CC7GBP, CC7GBRP, CC7GBS, or CC7GBRS to PWR.*
PC063	Gyro B on	CC7GBP, CC7GBRP, CC7GBS, or CC7GBRS to PWR.*
PC064	Omen	Store next three power codes in CCS memory.
PC065	Omen	Store next three power codes in CCS memory.
PC066	TCAPU failure	Store occurrence in CCS memory. This power code preceded by PC064 (omen).
PC067	Gyro failure	Store occurrence in CCS memory. This power code preceded by PC064 (omen).
PC070	Heater B on	CC7BBP, CC7BBRP, CC7BBS, or CC7BBRS to PWR.*
PC071	Heater B off	CC7BBP, CC7BBRP, CC7BBS, or CC7BBRS to PWR.*
PC072	Gyro C off	CC7GCP, CC7GCRP, CC7GCS, or CC7GCRS to PWR.*
PC073	Gyro C on	CC7GCP, CC7GCRP, CC7GCS, or CC7GCRS to PWR.*
PC074	Spare	Not used.
PC075	Spare	Not used.
PC076	Spare	Not used.
PC077	Spare	Not used.

*Which command of set to be issued depends on the power relay state table in the CCS.

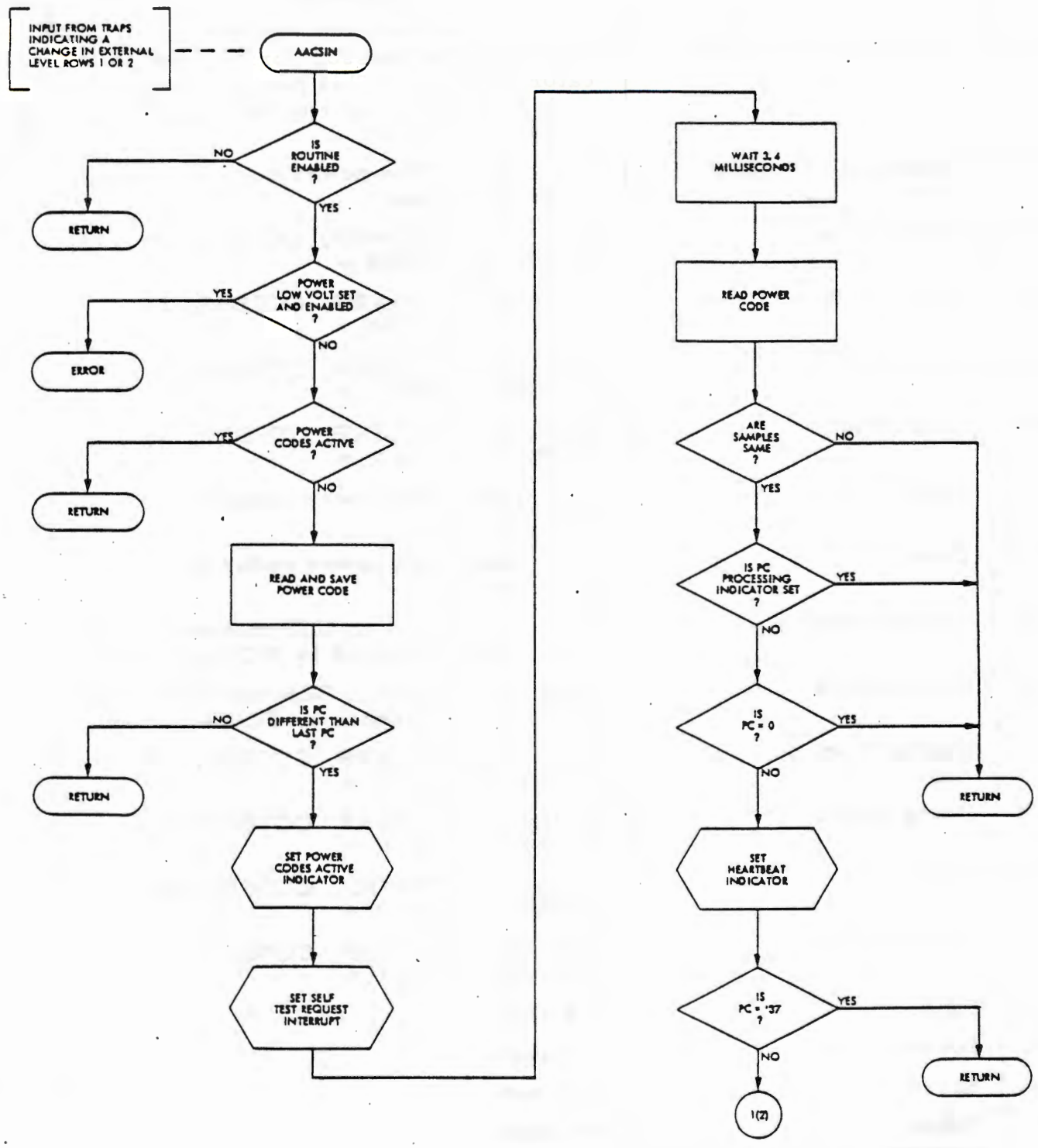


Fig. 3-5. Flow Diagram of the Routine AACSIN (Sheet 1 of 9)

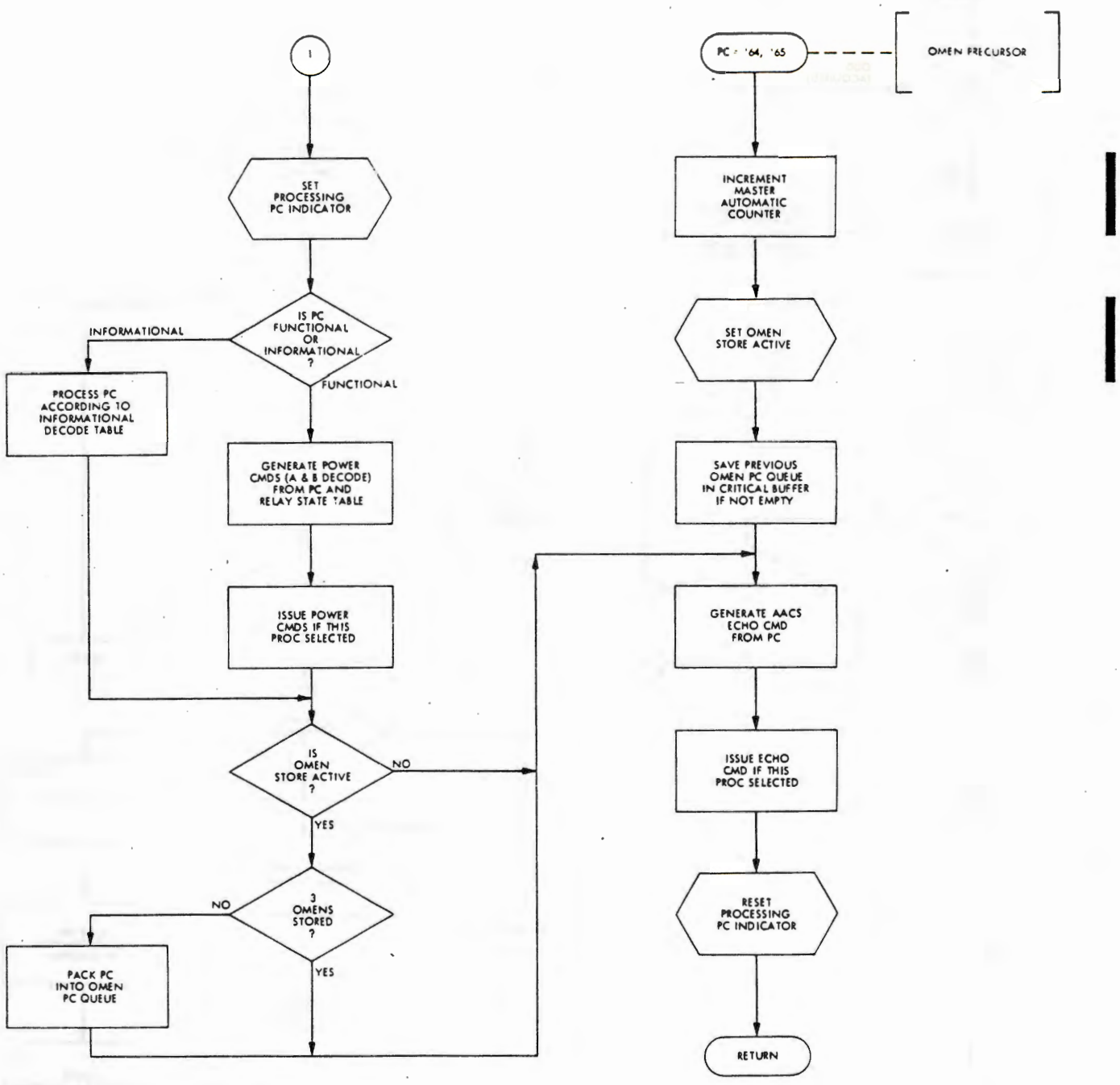
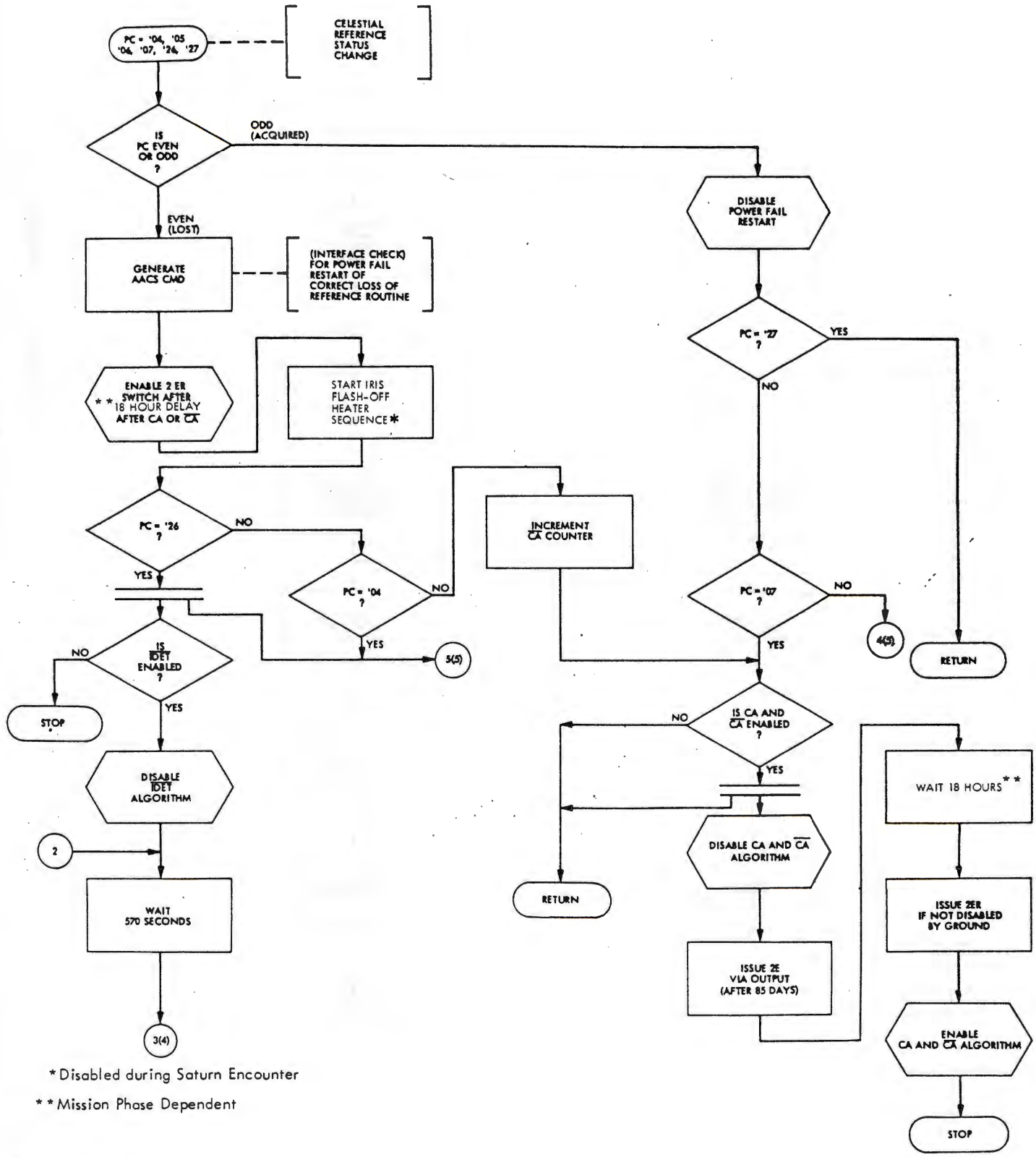


Fig. 3-5. Flow Diagram of the Routine AACSIN (Sheet 2 of 9)

Change #4
16 July 1981

46



*Disabled during Saturn Encounter

**Mission Phase Dependent

Change #4
16 July 1981

Fig. 3-5. Flow Diagram of the Routine AACSIN (Sheet 3 of 9)

47

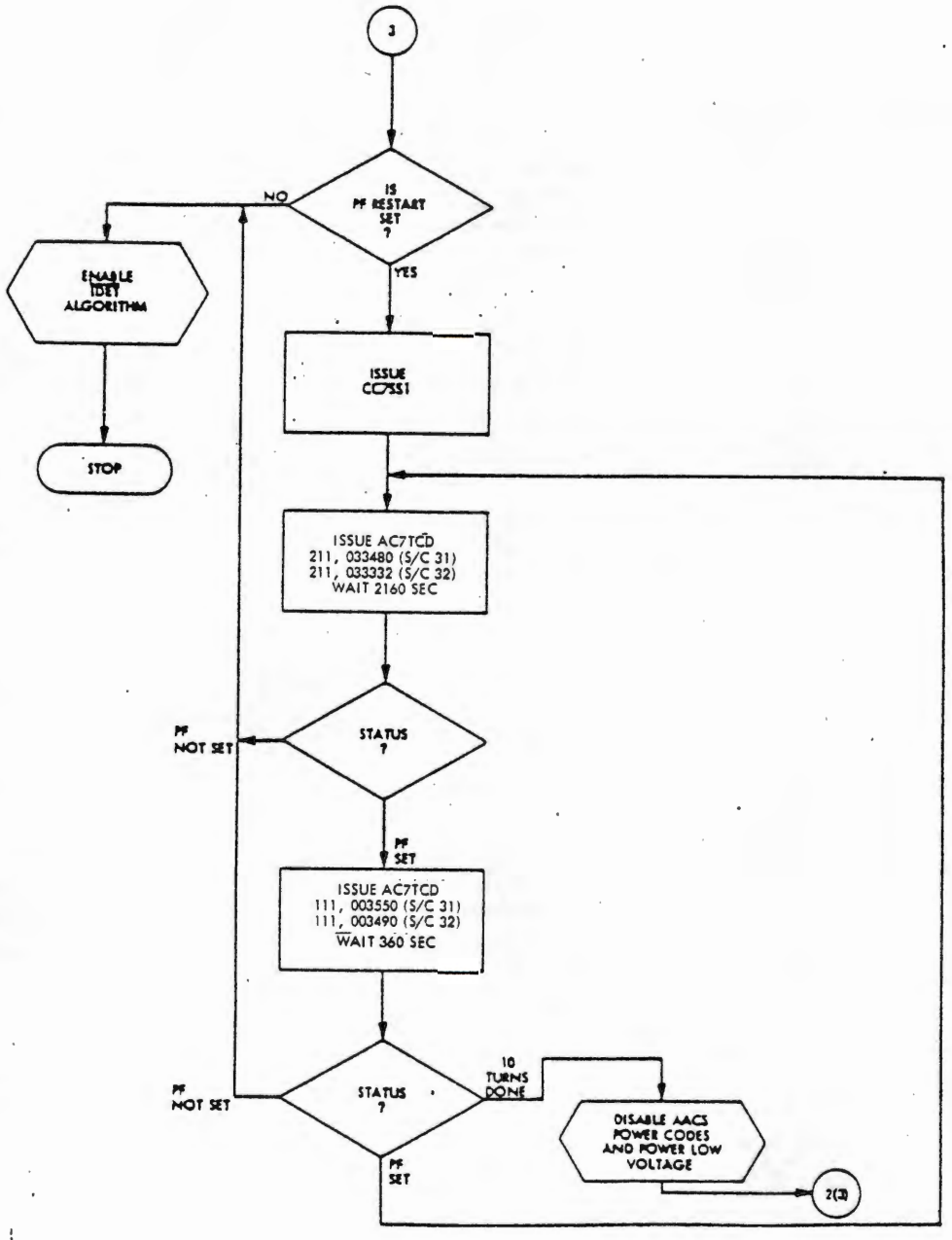


Fig. 3-5. Flow Diagram of the Routine AACSIN (Sheet 4 of 9)

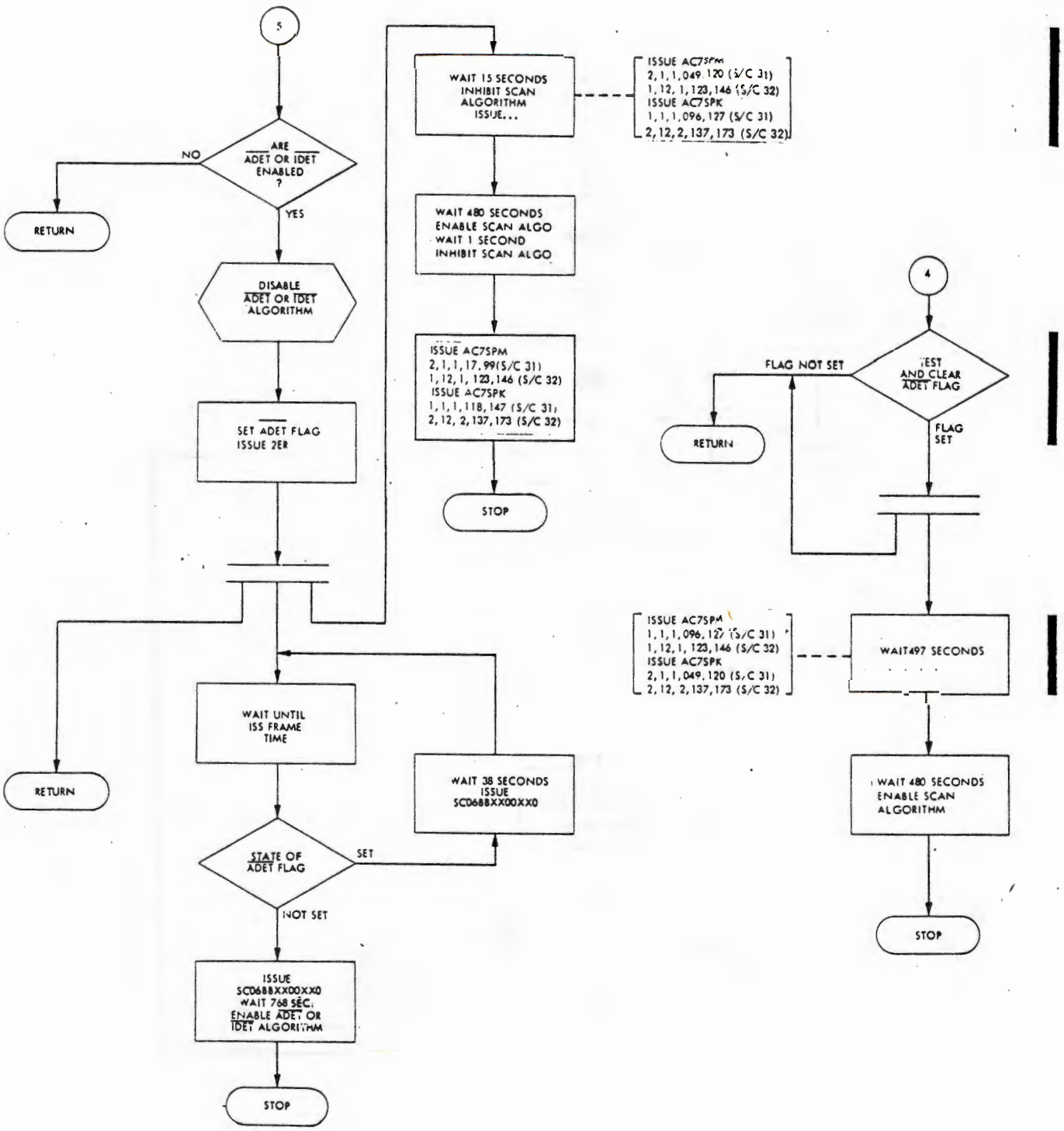


Fig. 3-5. Flow Diagram of the Routine AACSIN (Sheet 5 of 9)

49

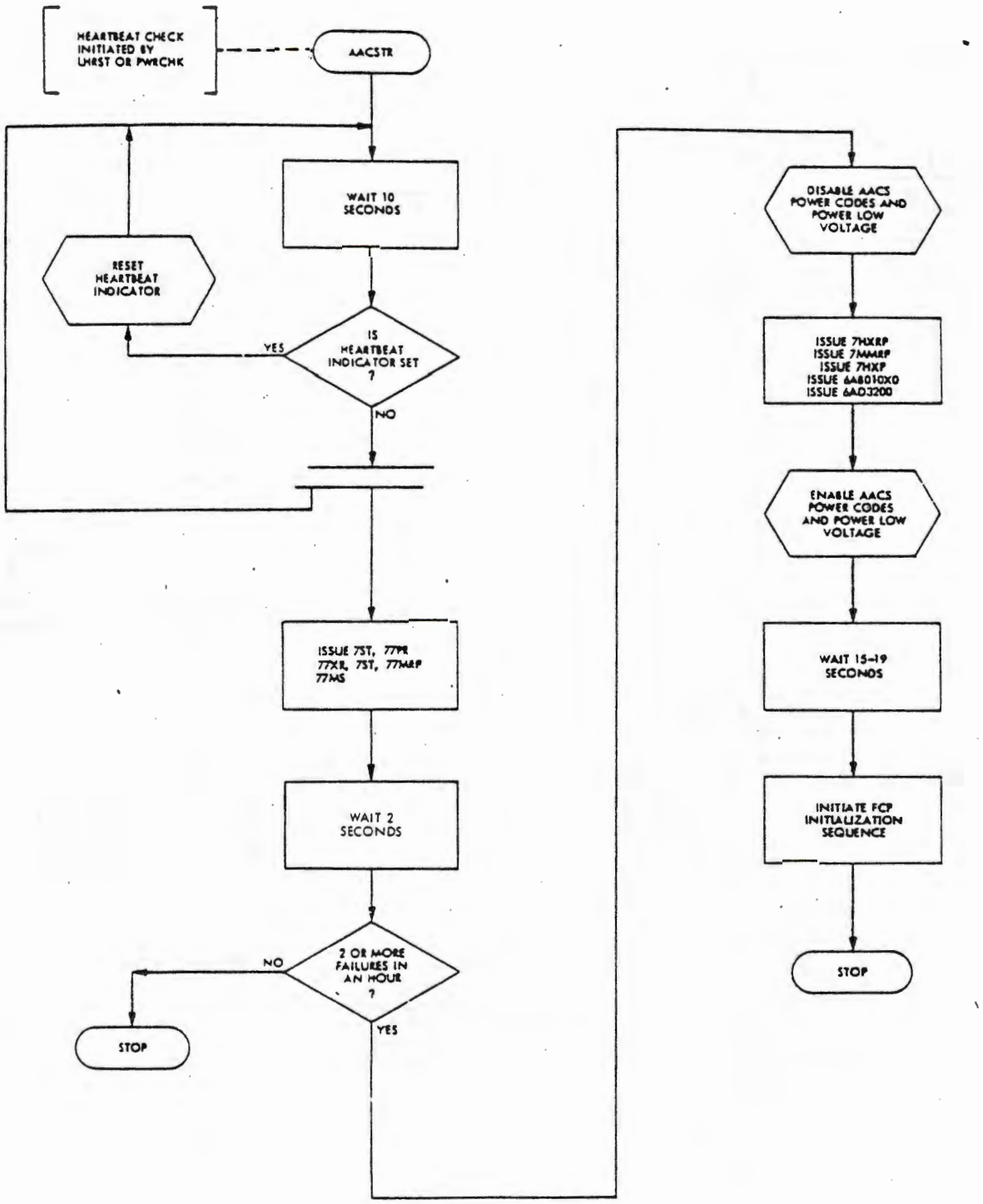


Fig. 3-5. Flow Diagram of the Routine AACSIN (Sheet 6 of 9)

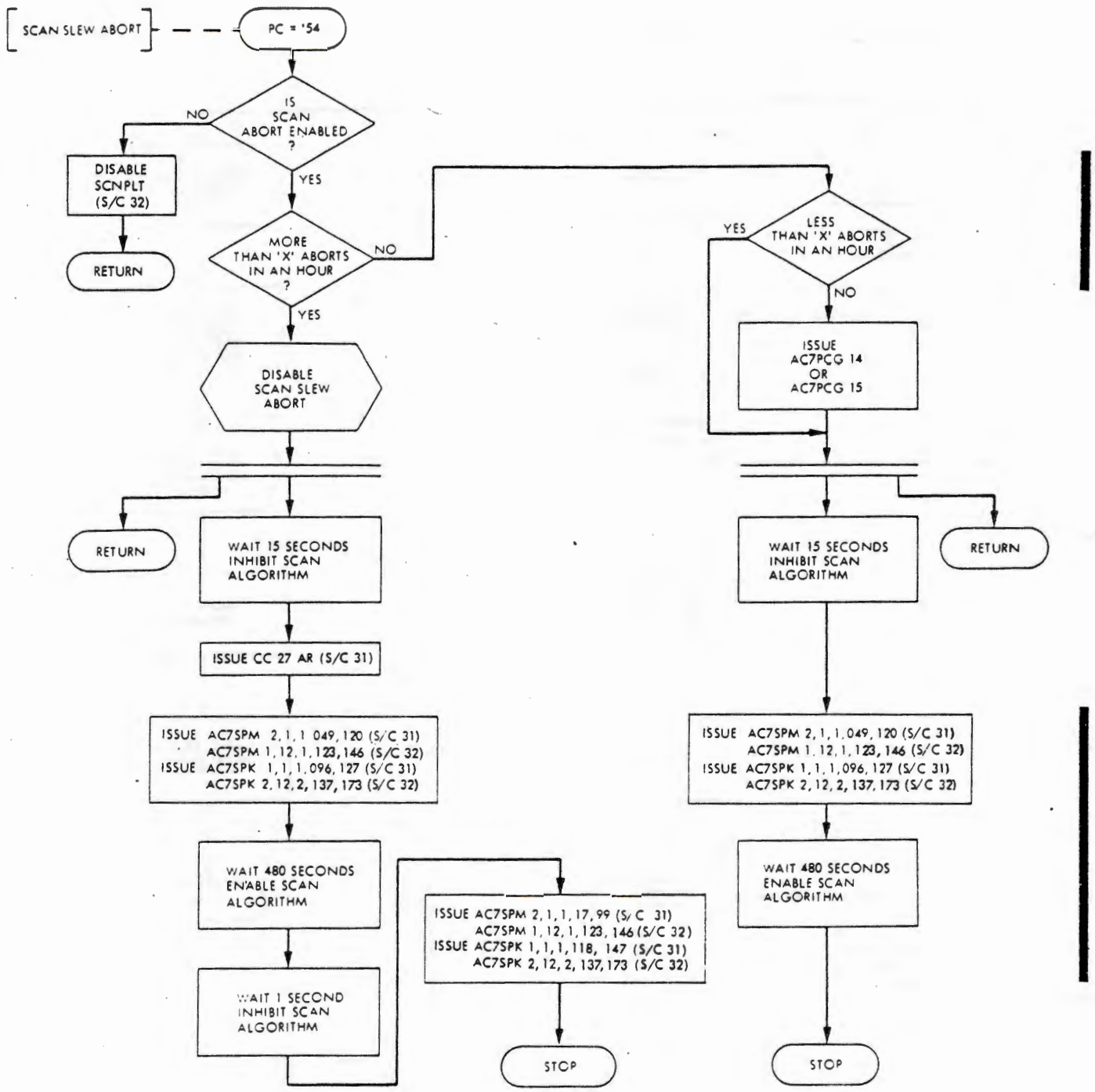


Fig. 3-5. Flow Diagram of the Routine AACSIN (Sheet 7 of 9)

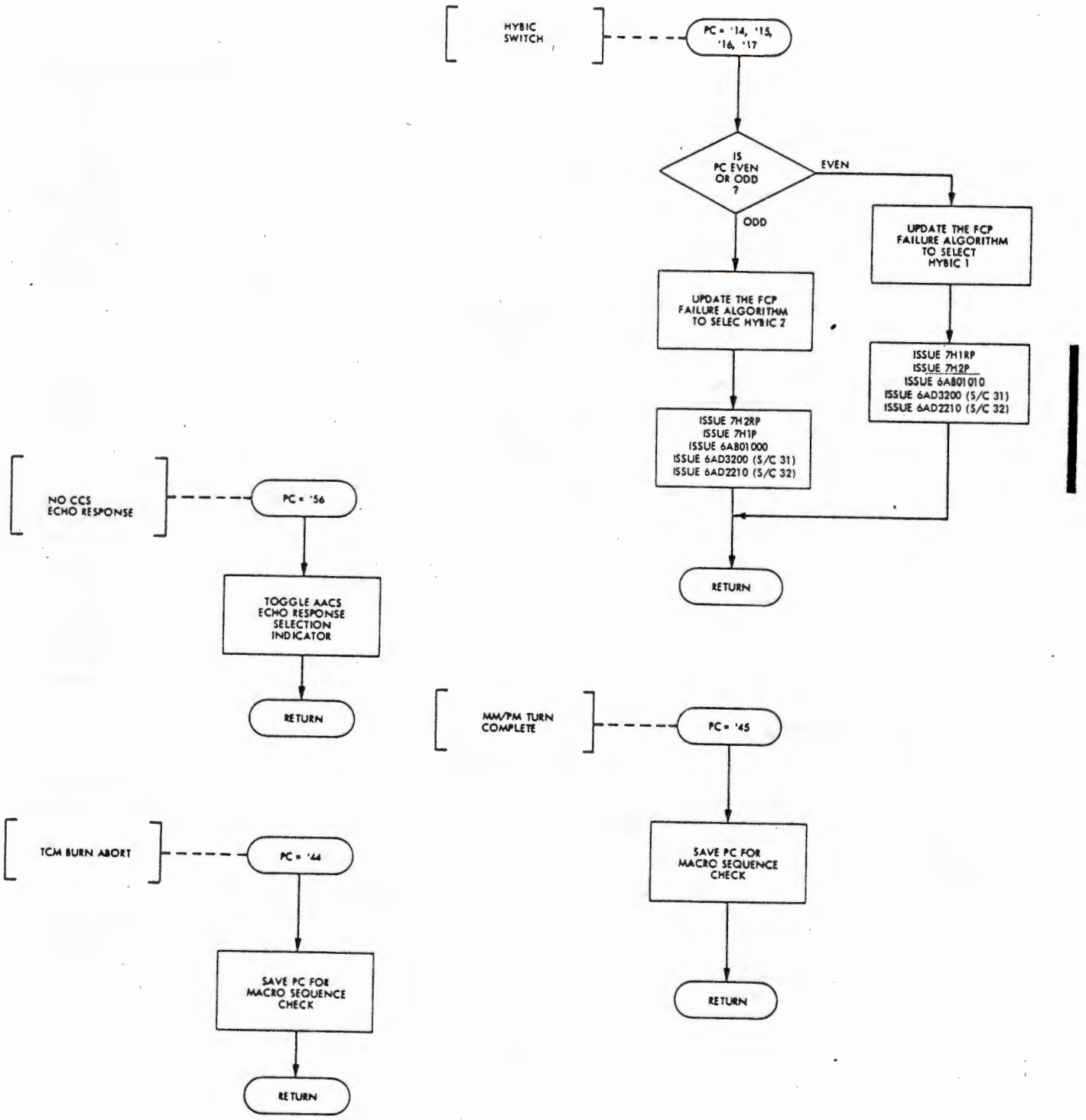


Fig. 3-5. Flow Diagram of the Routine AACSIN (Sheet 8 of 9)

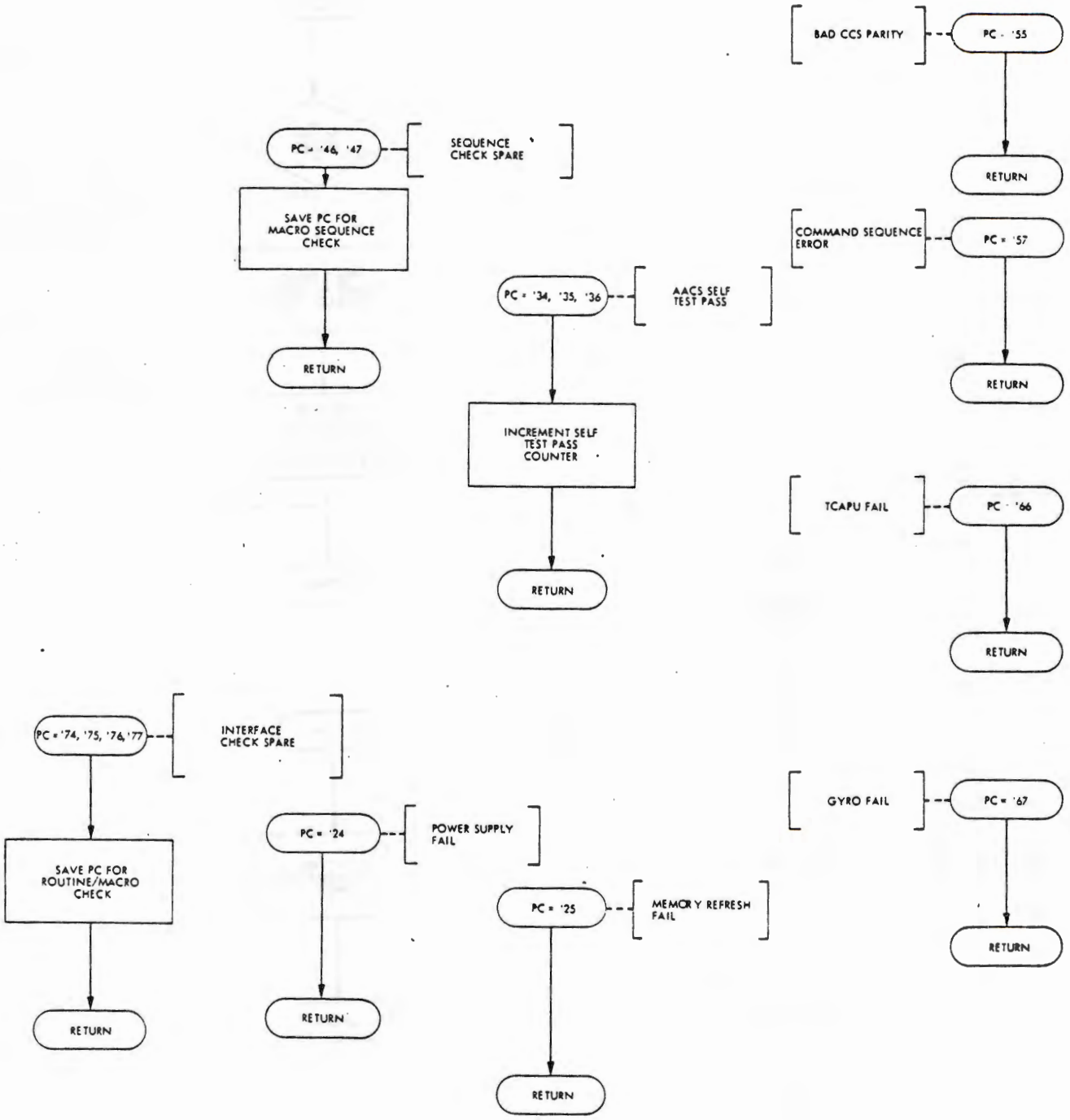


Fig. 3-5. Flow Diagram of the Routine AACSIN (Sheet 9 of 9)

53

2. Single words to be stored in or executed by the CCS (processor commands).

3. Block load commands for AACS, FDS, or CCS (block commands).

The digital data is received by the CCS via a bit-sync interrupt and a data level. A lock-change interrupt is also provided which is used to initialize the command decoding routine prior to receipt of command data. Because of the importance of command decoding to the spacecraft, these interrupts are never masked out. However, as with all interrupts, they are temporarily disabled whenever a CCS routine is active.

3.2.2.1.1 Description. This routine responds to three inputs received from the MDS. They are: a lock-change interrupt, a bit-sync interrupt, and a data level.

Whenever the MDS goes in or out of lock, a lock-change interrupt is generated. A lock-change interrupt causes this routine to initialize variables to begin command decoding and to search for the 5-bit command start code (11010). A running count of lock-change interrupts is kept in the CCS software and this count is telemetered from the CCS each hour, as illustrated in Figure 3-6.

After responding to the lock-change interrupt, this routine looks for the 5-bit command-start by responding to the bit-sync interrupt. As each bit-sync interrupt is processed, the routine samples the state of the data input to determine whether each bit of the command is a "1" or a "0." After the command-start code is detected, the routine accumulates and stores data bits in blocks of 18 which correspond to each word of the command. A 3-word base command always follows the command start code, and is illustrated in Figure 3-7.

The first word of the command is the operator word which identifies the type of command. Table 3-2 illustrates the eight different types of commands.

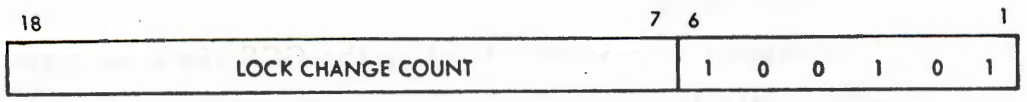


Fig. 3-6. Lock-Change Telemetry Word

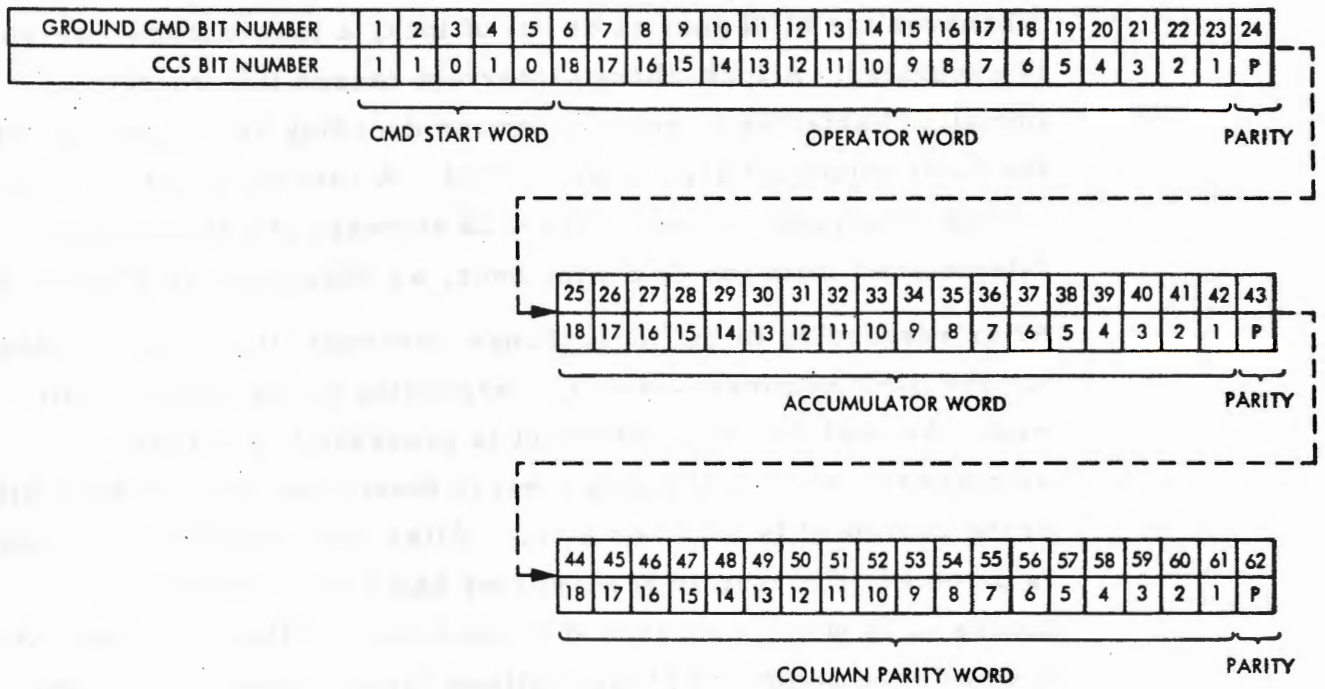


Fig. 3-7. Command Word Format

Table 3-2. CCS Operator Words

	GROUND CMD BIT NUMBER			31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
	CCS BIT NUMBER			18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
CC OR DC	CC OR DC			0	0	0	0	0	0	0	0	0	0	0	0	0	0	1			
PC	STORE			1	MEM ADDRESS											0	1	1			
	TRANSFER			MEM ADDRESS												0	1	0			
	INTERNAL EXECUTE			OPERAND												1	0	0			
	BLOCK TYPE	0	COND. CCS EXECUTE	0	0	0	0	0	0	BLOCK SIZE						0	0	0			
		1	COND. EVENT EXEC	0	0	0	1														
		2/3	CCS LOAD	0	0	1	X														
		4/5	CCS LOAD	0	1	0	X														
		6	CCS EXECUTE	0	1	1	0														
		7	EVENT EXEC	0	1	1	1														
		8	AACS LOAD	1	0	0	0														
		11	FDS H/W LOAD	1	0	1	1														
		10/14	FDS PRI MEM LOAD	1	A	1	0														
		9/13	FDS SEC MEM LOAD	1	A	0	1														
	RETURN			MEM ADDRESS												1	0	1			
	STORE INDIRECT			MEM ADDRESS												1	1	0			
	EXECUTE			MEM ADDRESS												1	1	1			

A = 1 FOR MEM PROTECT OVERRIDE

X = DON'T CARE

MODIFIER

MODIFIER ID

PROCESSOR ID { 1 = PROCESSOR B; 1 = PROCESSOR A }

SPACECRAFT ID { 0 = S/C 31; 1 = S/C 32 }

Change #1
6/11/79

The second word is the accumulator word and provides additional data depending on the type of command as follows:

CC or DC: The accumulator word contains the actual command to be executed. These commands are listed in Command Structure and Assignments FR, VGR77-3-290.

Store: A store command allows new information to be stored in any non-protected memory location. The memory location is specified by bits 7 through 18 of the operator word, with the new information to be stored contained in the accumulator word.

Transfer: A transfer command will cause CCS program control to transfer to a subroutine anywhere in memory. Bits 7 through 18 of the operator word contain the address of the first word of the subroutine. The accumulator may contain data to be used by the subroutine if required, otherwise it is ignored.

Internal Execute: The internal execute command was originally provided to allow processor-to-output-unit communication paths to be easily changed if desired. However, because the philosophy of having one processor clamp or inhibit the other processor from outputting commands (unless a special self-test algorithm is successfully passed) has been implemented in the software, the utility of these commands has been reduced. Only the commands associated with enabling or disabling the other processor's outputs can be used. However, there is no known requirement to ever use these commands.

Block: Block commands are utilized to load multiple memory words into the CCS, FDS, or AACS memories (block load commands), or to execute a special algorithm which is coded within the block command itself (program or event execute block commands). Each block has, as a header, a 62-bit base command. The "block" of data that follows may contain up to 36 18-bit words (plus parity), plus a column parity word. The basic block command format is illustrated in Figure 3-8.

The operator word associated with a block command is coded according to the type of block command, as shown in Table 3-2. The accumulator word contains block sequence numbers which are used by the CCS to verify the correct sequence of block commands when required. These numbers and their use are illustrated in Figure 3-10.

The basic command block format for CCS and AACS block loads is illustrated in Figure 3-9. It consists of a memory base address, into which memory word 1 is to be loaded, and additional memory words which are to be loaded contiguously to word 1 within the affected memory. More than one base address is allowed within the command block to facilitate "scatter loading" of the CCS or AACS.

The basic block format for FDS loads is illustrated in Figure 3-11. A slightly different format is allowed for FDS since their memory words are only 16 bits long. This allows the two extra bits to be used to identify the type of data contained in each word.

Conditional program or event execute block commands are used in conjunction with CCS memory block loads. CCS memory loads typically load new sequence information into the sequence software portion of the CCS memory. As these new memory words are received, they are stored in the proper memory location but are not acted upon (dormant). Activation of the new sequence information is typically done at the end of the memory loading by a conditional program execute block. The data within the block contains the software necessary to activate the previously loaded data. This activation is conditional, however, on the reception of the prior memory load blocks in the correct sequence as determined from the block sequence numbers explained in Figure 3-9. It should be noted that memory load blocks will be stored in memory as they are received – regardless of whether or not they are in order. They just won't be activated if they are not in order.

58

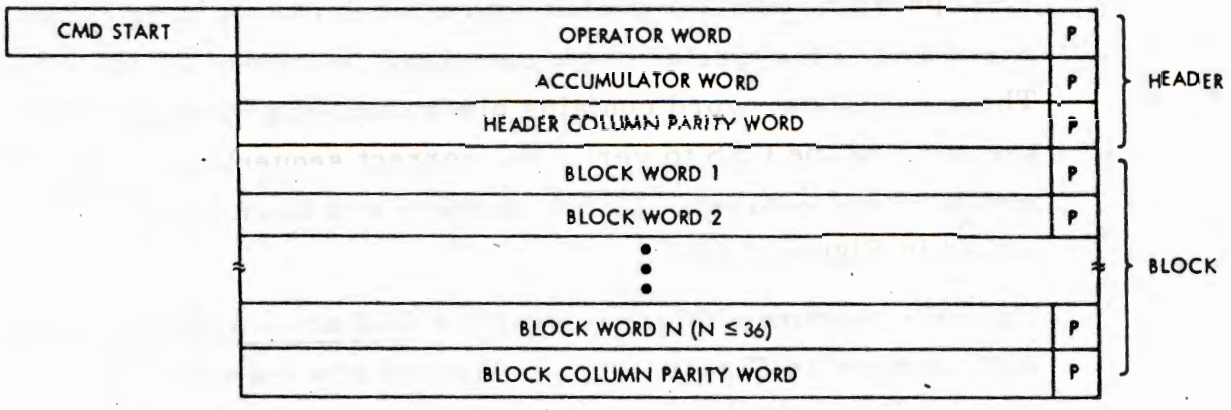


Figure 3-8. Basic Block Command Format

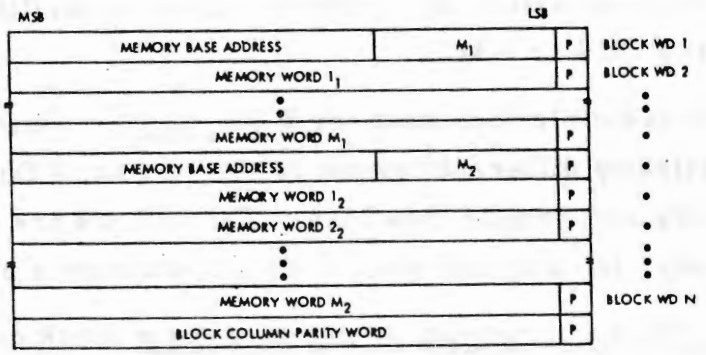


Figure 3-9. CCS/AACS Memory Load Format

Unconditional program or event execute blocks are used to transmit a special sequence of events or routine that is to be executed only once, immediately upon receipt by the CCS. This special routine or sequence is not retained in memory but is executed right from the command buffer within the command decode routine itself. There is a definite constraint associated with use of this command, and that is that no other command should be sent to the CCS until the execute block command has finished executing. Overlaying new command data in the command buffer before completion of a previous execute command could cause undesirable operation of the CCS.

As each bit of command data is received by the CCS, the terminate indicator is checked to see if the previous bit caused the

GROUND COMMAND BIT NUMBER	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42
CCS BIT NUMBER	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
BLOCK ID	MSB											LSB						
BLOCK NO.													MSB					LSB

THE BLOCK ID AND NO. ARE USED TO GIVE EVERY COMMAND BLOCK A UNIQUE IDENTIFIER. THE BLOCK ID AND NO. ARE ALSO USED TO CONTROL THE CCS OUT-OF-SEQUENCE INDICATOR UNDER THE FOLLOWING GROUND RULES.

a. ANY ACCUMULATOR WORD WITH A ZERO BLOCK NUMBER HAS NO EFFECT ON THE OUT-OF-SEQUENCE INDICATOR.

b. THE OUT-OF-SEQUENCE INDICATOR IS RESET ONLY BY RECEIVING A NEW COMMAND WITH A BLOCK NO. EQUAL TO ONE.

BLOCK ID	BLOCK NO.	OUT-OF-SEQUENCE INDICATOR
6	1	RESET
6	2	NO CHANGE
7	0	NO CHANGE
6	3	NO CHANGE
6	5	SET*
6	6	
8	0	
7	1	RESET

*SINCE 6-4 IS MISSING IN THE SEQUENCE, 6-5, 6-6, AND 8-0 WILL ALL BE MARKED OUT OF SEQUENCE. BECAUSE OF THIS, THE OUT-OF-SEQUENCE INDICATOR WILL BE SET, AND ANY CONDITIONAL PROGRAM EXECUTE BLOCK WILL BE REJECTED UNLESS THE INDICATOR IS FIRST RESET. THE OUT-OF-SEQUENCE INDICATOR WILL RESET WITH 7-1.

Figure 3-10. Block Accumulator Word

6D

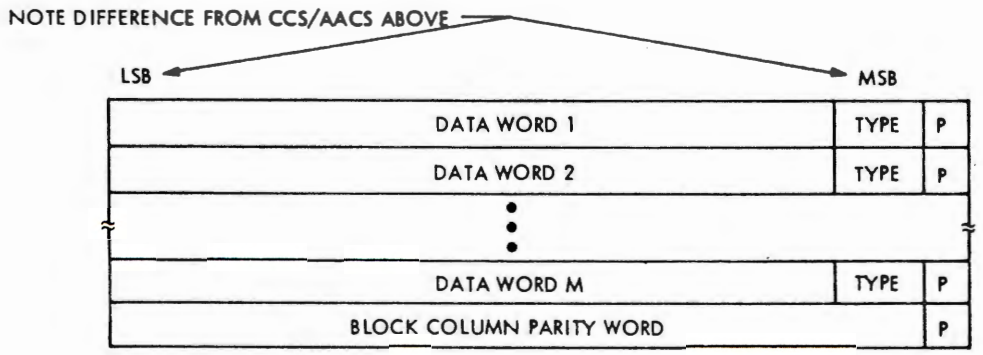


Figure 3-11. FDS Memory Load Format

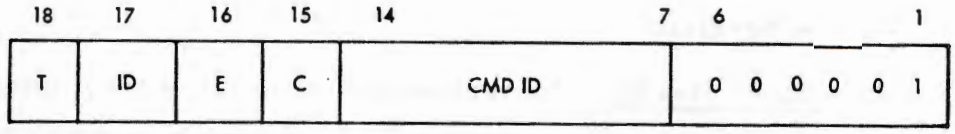
indicator to be set. If it is set, the command decode routine stops processing data, outputs a telemetry word as in Figure 3-12 or 3-13, and waits for a new lock-change interrupt. If it is not set, the routine continues.

As each 18 bit word is accumulated, row parity is checked. If a parity error is detected, and it is the first one of the command, it is stored for later correction. If it is the second or more error, the reject indicator is set, and the routine continues.

After all command words or rows have been decoded, column parity is checked. If there are no errors, the routine continues. If there is one column error and a corresponding row error, it is corrected before continuing. If there is one column error without a row error or two or more column errors, the reject indicator is set.

If the reject indicator is set and the received command is a base command, command-terminate telemetry is generated (Figure 3-11), and command decoding is terminated until a new lock-change interrupt is received. If the reject indicator is set, and the received command is a block command, the command is rejected, reject- telemetry is generated (Figure 3-12), and the routine continues by searching for a command start code.

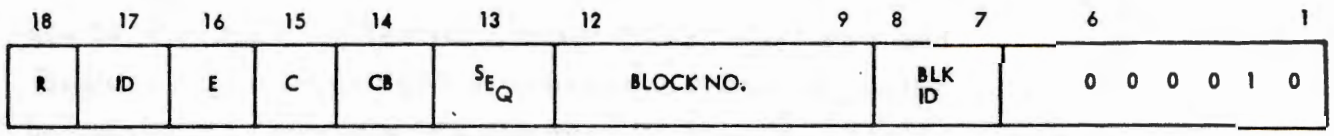
61



T: COMMAND DECODING TERMINATED IF SET TO 1
ID: S/C OR PROCESSOR ID IN ERROR IF SET TO 1
E: COMMAND START CODE HAD ONE ERROR IF SET TO 1
C: ONE BIT ERROR CORRECTED IF SET TO 1
CMD ID: BITS 18-11 OF ACCUMULATOR WORD

IF T BIT IS SET, TELEMETRY WORD WILL BE SENT TO THE CCS CRITICAL TELEMETRY BUFFER AS WELL AS THE NORMAL BUFFER.

Figure 3-12. Base Command Telemetry



R: COMMAND REJECTED IF SET TO 1
ID: S/C OR PROCESSOR ID IN ERROR IF SET TO 1
E: COMMAND START CODE HAD ONE ERROR IF SET TO 1
C: ONE BIT ERROR CORRECTED IN BASE COMMAND IF SET TO 1
CB: ONE BIT ERROR CORRECTED IN BLOCK COMMAND IF SET TO 1
SEQ: COMMAND BLOCK OUT OF SEQUENCE IF SET TO 1
BLOCK NO: FOUR LEAST SIGNIFICANT BITS OF COMMAND BLOCK NUMBER
BLK ID: TWO LEAST SIGNIFICANT BITS OF COMMAND BLOCK ID
} SEE FIGURE 3-9

IF R BIT IS SET, TELEMETRY WORD WILL BE SENT TO THE CCS CRITICAL TELEMETRY BUFFER AS WELL AS THE NORMAL BUFFER.

Figure 3-13. Block Command Telemetry

62

If there were no errors or one error that was corrected, the routine then executes the command depending on the type as follows:

1) Base Command: Wait ten bit times; reset command loss routine hour-count; generate command telemetry; send command to event output routine for transmission to affected subsystem.

2) Block Header: Generate command telemetry; check block sequence numbers and set out-of-sequence indicator if command is out of sequence with previous command; begin decoding of block.

3) Block Command (Except Conditional Execute): Wait 10 bit times; reset command loss routine hour count; generate command telemetry; then, depending on block type, continue as follows:

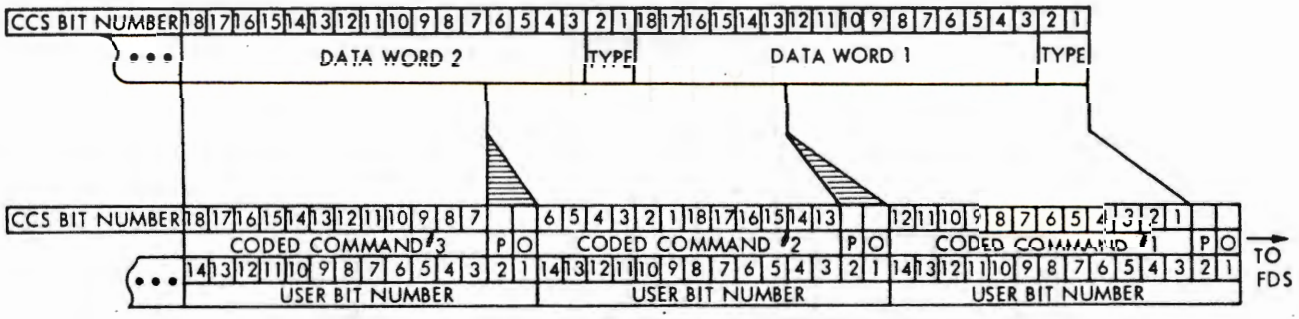
   a. Unconditional Execute: Jump to command word 1 in command buffer and execute software code therein.

   b. CCS Block Load: Store command word 2 and subsequent, according to address stored in command word 1 (Figure 3-8).

   c. FDS Block Load: Send CC command to alert FDS of block load and which type: hardware or software; segment command words in command buffer to be compatible with 12-bit CC command format per Figure 3-14 and output those CC commands through event output routine; at end of block load generate an "all-ones" CC command which will identify the completion of the block load for FDS.

   d. AACS Block Load: Segment command words in command buffer to be compatible with 13-bit CC command format per Figure 3-15, and output those CC commands through event output routine.

The flow diagram for this routine is illustrated in Figure 3-16.



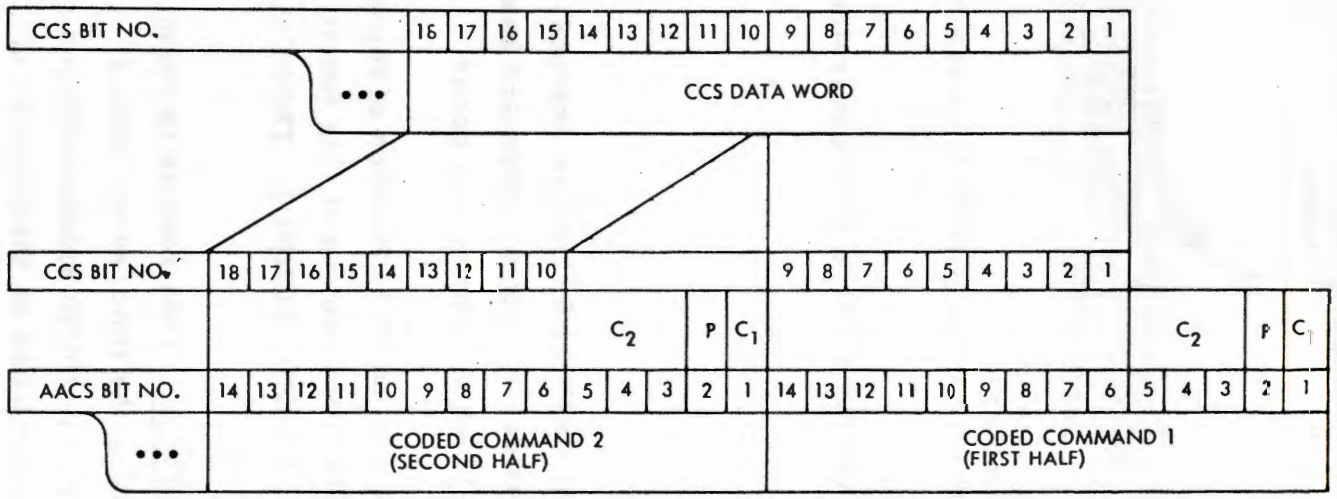
NOTES: 1) USER BIT #1 RECEIVED FIRST BY USER SUBSYSTEM
2) P REPRESENTS ODD PARITY ON USER BITS 2-14
3) USER BITS 1 AND 2 ARE GENERATED BY CCS AND NOT INCLUDED IN THE CCS BLOCK LOAD

Figure 3-14. FDS Command Word to CC Conversion

3.2.2.1.2 Constraints. Any command that follows an execute-block command must be delayed until after the execute command has terminated or anomalous execute-block operation may occur.

3.2.2.2 Sequencing. This is the second main function of the intermediate processing section. The main routine of this function is TARMEX, all others, ANTUPD, DTRPRC, SCNPLT, TRNSUP, and LHRST are dedicated subroutines.

3.2.2.2.1 Sequence Support (TARMEX). This routine is responsible for processing and controlling the time/event "tables" typically stored within the CCS memory to perform spacecraft sequence functions. Time/event tables are developed on the ground using system test or flight operations support software and loaded into the CCS by block load. Time/event tables are normally dormant when loaded into the CCS. They are typically activated by ground command, other routines within the CCS, or by an entry in an already active time/event table. "Events" contained within a time/event table are most often commands to other subsystems, but, on occasion,



NOTES: (1) USER BIT 1 RECEIVED FIRST BY USER SUBSYSTEM
(2) C_1 AND C_2 ARE CODE BITS GENERATED BY CCS AND NOT INCLUDED IN THE CCS BLOCK LOAD
(3) P REPRESENTS ODD PARITY ON USER BITS 1-14

Figure 3-15. CCS Command Word to AACS CC Conversion

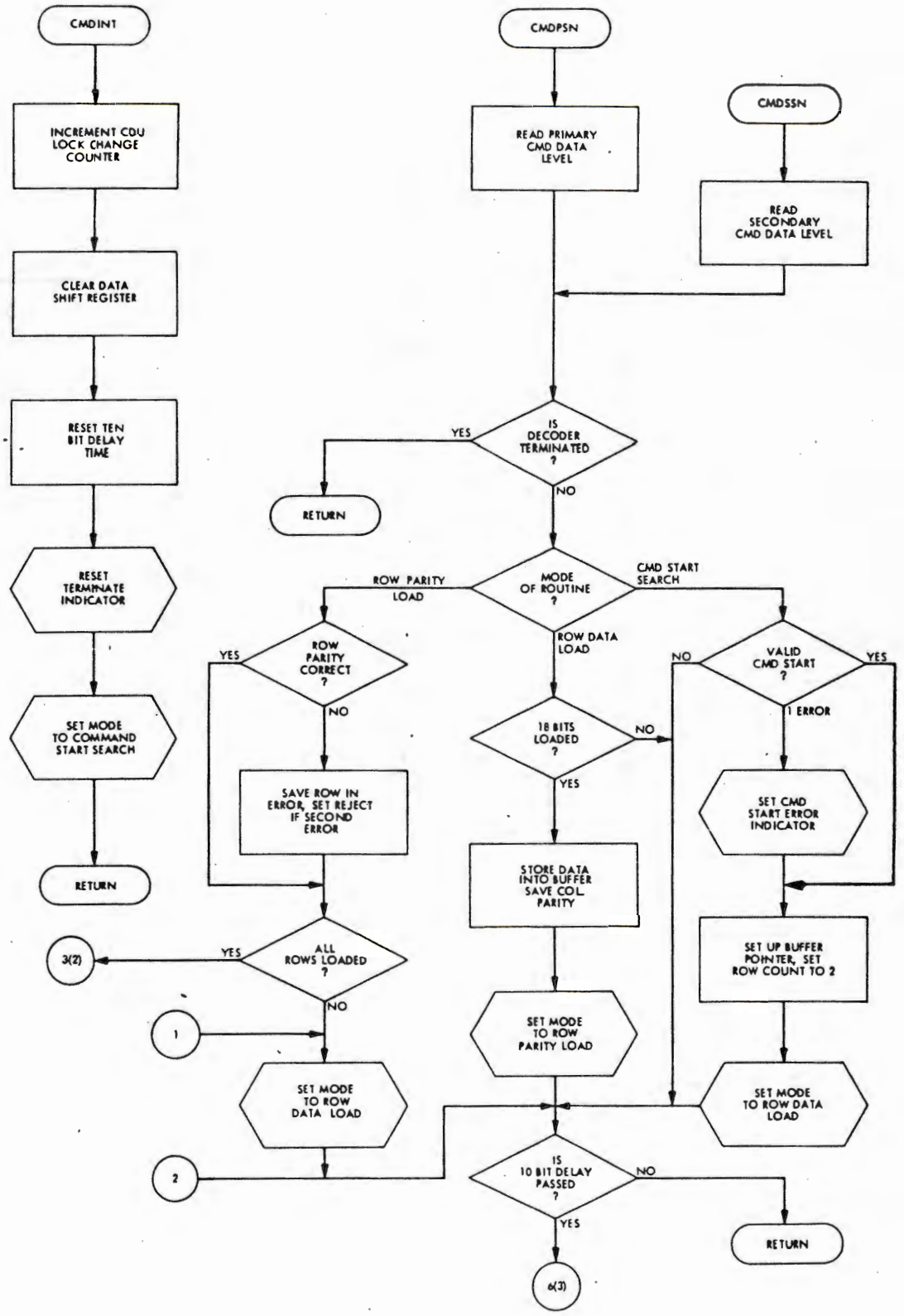


Figure 3-16. Flow Diagram of the Routine CMDPRC (Sheet 1 of 3)

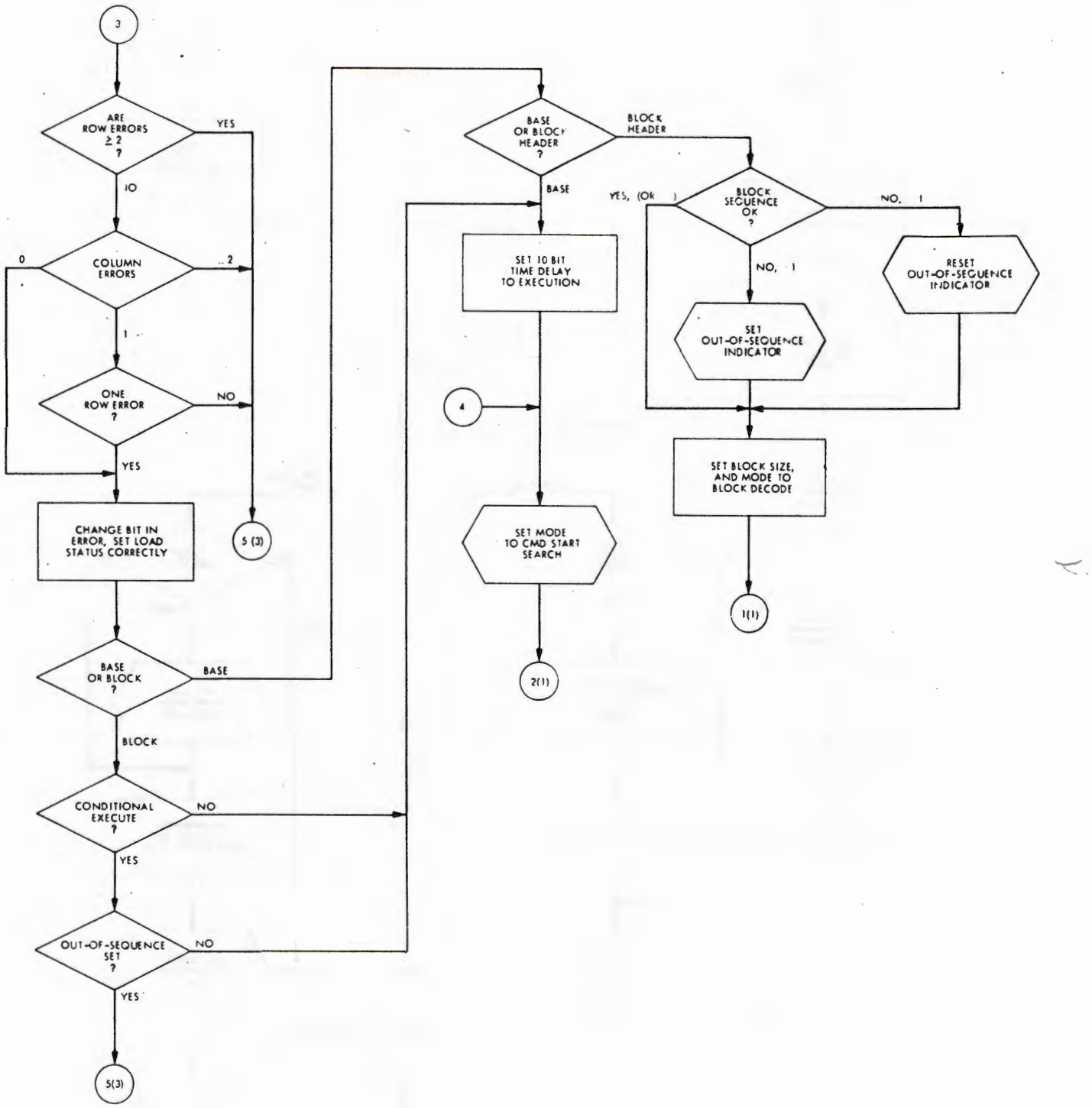


Figure 3-16. Flow Diagram of the Routine CMDPRC (Sheet 2 of 3)

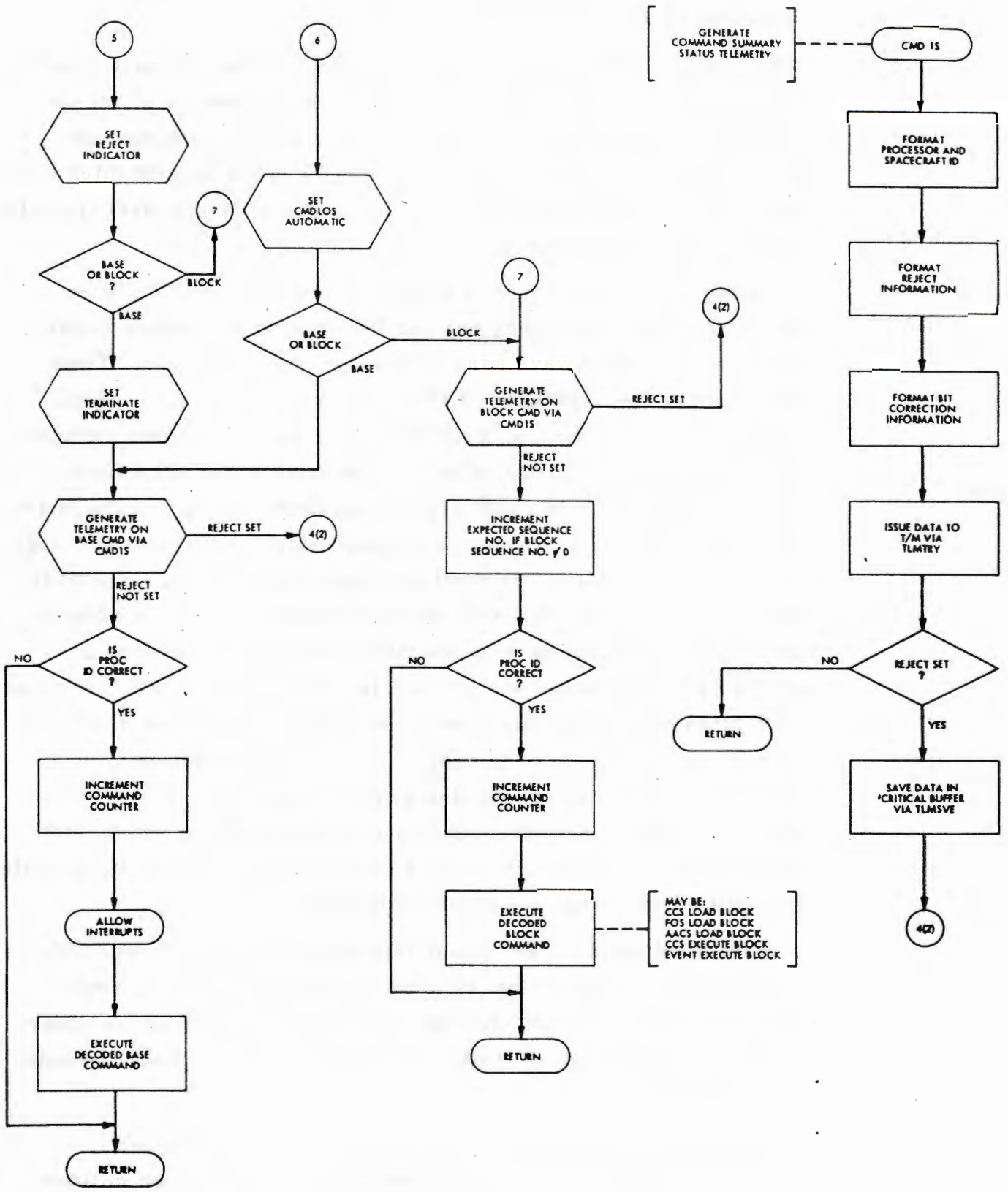


Figure 3-16. Flow Diagram of the Routine CMDPRC (Sheet 3 of 3)

fn.8

can be special instructions or "pseudo events" meaningful only to the CCS software as discussed below.

3.2.2.2.1.1 Description. The sequence support routine is the primary control and processing routine for preprogrammed sequences of events called time/event tables. This routine is rather complex and some details of how the routine operates will not be presented here. However, its basic operation will be discussed to the level typically visible at the system level.

A typical time/event entry is illustrated in Figure 3-17. It is typically a two-word entity but can involve more than two words. The first word is always associated with time, however. Time bases within the capability of this routine to handle are: centi-seconds, seconds, hours, FDS ISS frame starts. Times included within time/event tables are referenced to the preceding event, and are therefore relative times not absolute, except for the hour time base which is absolute. The hour "clock" will start at launch or shortly before and continue throughout the mission, or until it makes sense to reset it. One of the "problems" associated with timekeeping when more than one table of a given time-base is active, is to determine which event in which table is really the next event to be executed. Obviously, each table must be checked and time deltas kept track of in order that the next event be identifiable after each preceding event is executed. This routine has been designed with the capability to handle thirty active time/event tables of various time bases concurrently. The end of a table is identified by a negative number, typically -1.

As mentioned earlier, an "event" entry within a time/event table is typically a command that is to be sent to another subsystem. However, other types of "events" have been provided for in order to increase the utility of time/event tables. These pseudo-events are as follows:

1) Multiple Time Events: This pseudo-event is designed to take advantage of commands that are equally spaced in time such that a time word is not needed for each command – only one that identifies the same time spacing for all commands

19

in the block. This pseudo-event is illustrated in Figure 3-18. Other pseudo-events, except a time-base change or additional multiple time events, are allowable within this pseudo-event.

2) CCS Instruction Execution: This pseudo-event allows a certain subset of CCS instructions to be executed from a time/event table. This event is either one or two words in length depending on whether the accumulator register is to be used to transfer additional data. The details of this pseudo-event are illustrated in Figure 3-19. This pseudo-event is used to perform non-event functions in sequence with normal events or commands.

3) Table Time-Base Change: This pseudo-event is used to change the time-base of a time/event table. It allows related events to be included within the same table even though they may not have the same time resolution. Figure 3-20 illustrates this event.

4) Conditional Event: This pseudo-event allows selection between two events by any of three means: (1) Depending on the condition of the indicator word itself (zero or non-zero). (2) Depending on the state of a level input specified by the indicator word. (3) Depending on the state of a location elsewhere in memory specified by the indicator word. This event is illustrated in Figure 3-21. Only commands and single-word events are allowed in this event.

5) Activate Time/Event Table: This pseudo-event allows a dormant time/event table to be activated. It contains the starting address of the dormant table and the time-base to be used with the table. It is illustrated in Figure 3-22.

6) DSS Tape Position: This pseudo-event is used to position the tape recorder at any desired place on the tape prior to a record or playback sequence. This event simply contains an absolute tape position which is related to a tape position algorithm driven by the indexing pulses received by the CCS from the DSS. The algorithm is located in the DSS slew and

10

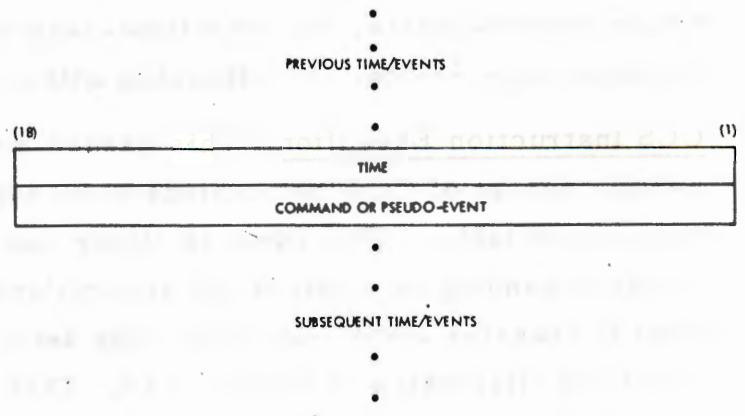


Figure 3-17. Time/Event Entry in Table

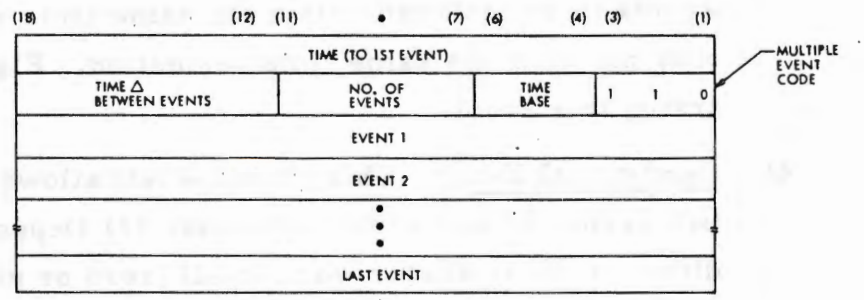


Figure 3-18. Multiple Event Block

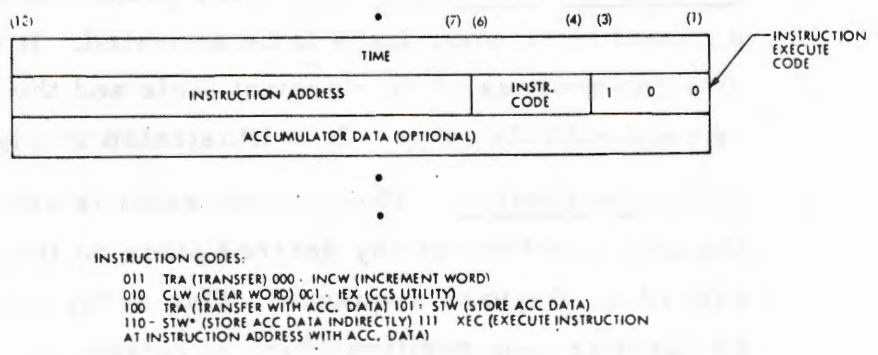


Figure 3-19. Instruction Execution Event

in the block. This pseudo-event is illustrated in Figure 3-18. Other pseudo-events, except a time-base change or additional multiple time events, are allowable within this pseudo-event.

2) CCS Instruction Execution: This pseudo-event allows a certain subset of CCS instructions to be executed from a time/event table. This event is either one or two words in length depending on whether the accumulator register is to be used to transfer additional data. The details of this pseudo-event are illustrated in Figure 3-19. This pseudo-event is used to perform non-event functions in sequence with normal events or commands.

3) Table Time-Base Change: This pseudo-event is used to change the time-base of a time/event table. It allows related events to be included within the same table even though they may not have the same time resolution. Figure 3-20 illustrates this event.

4) Conditional Event: This pseudo-event allows selection between two events by any of three means: (1) Depending on the condition of the indicator word itself (zero or non-zero). (2) Depending on the state of a level input specified by the indicator word. (3) Depending on the state of a location elsewhere in memory specified by the indicator word. This event is illustrated in Figure 3-21. Only commands and single-word events are allowed in this event.

5) Activate Time/Event Table: This pseudo-event allows a dormant time/event table to be activated. It contains the starting address of the dormant table and the time-base to be used with the table. It is illustrated in Figure 3-22.

6) DSS Tape Position: This pseudo-event is used to position the tape recorder at any desired place on the tape prior to a record or playback sequence. This event simply contains an absolute tape position which is related to a tape position algorithm driven by the indexing pulses received by the CCS from the DSS. The algorithm is located in the DSS slew and

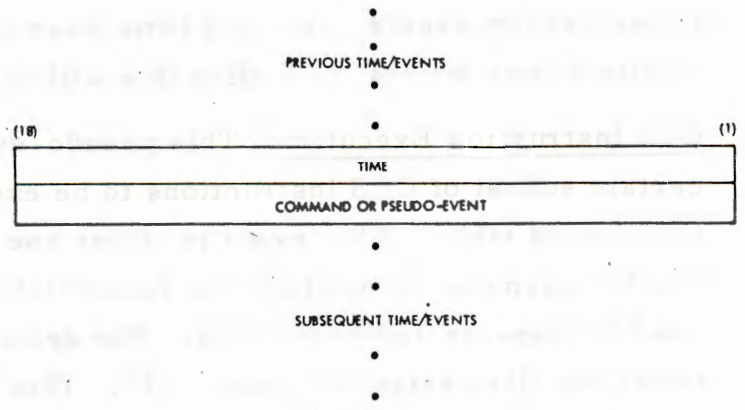


Figure 3-17. Time/Event Entry in Table

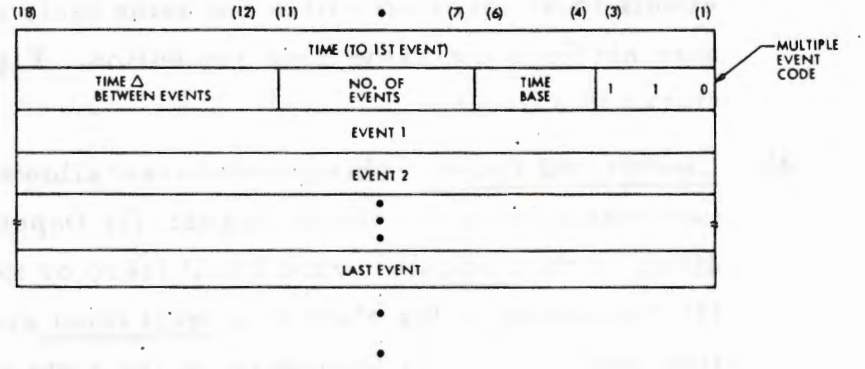


Figure 3-18. Multiple Event Block

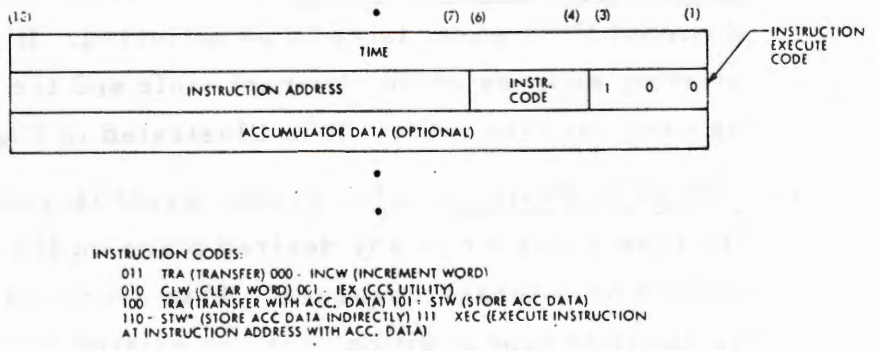


Figure 3-19. Instruction Execution Event

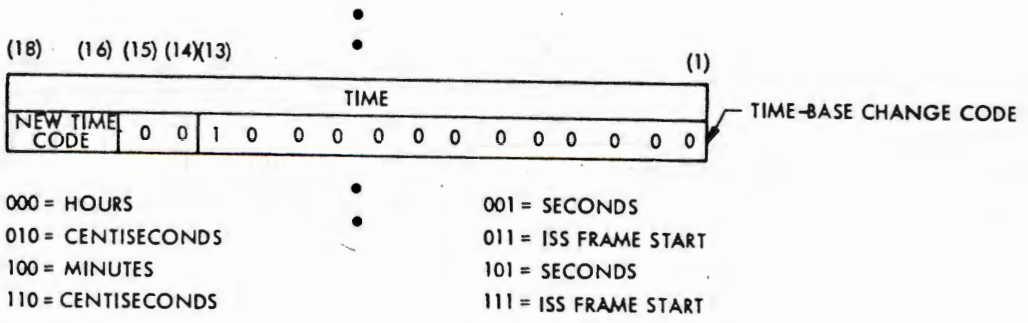


Figure 3-20. Time-base Change

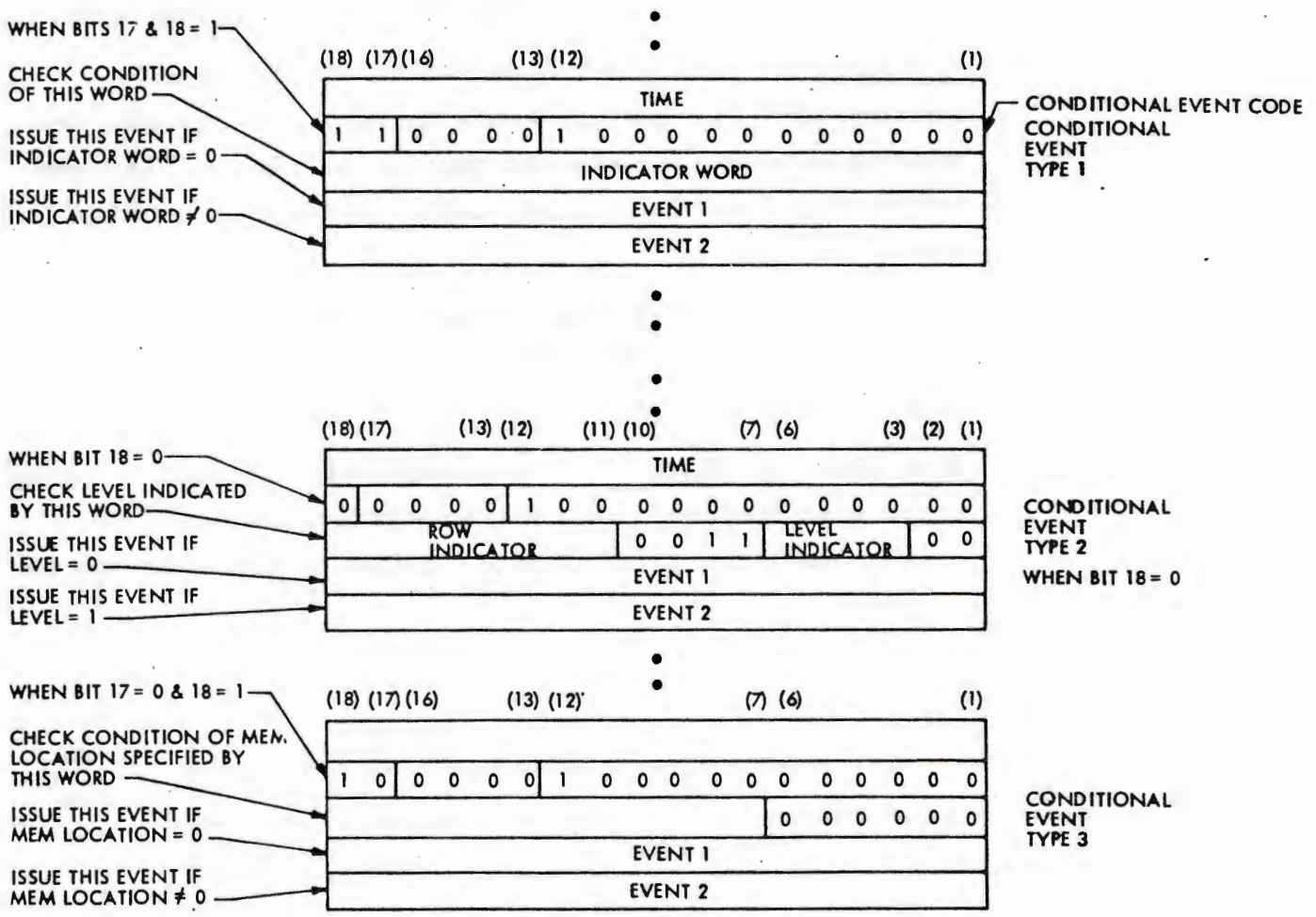


Figure 3-21. Conditional Event

72

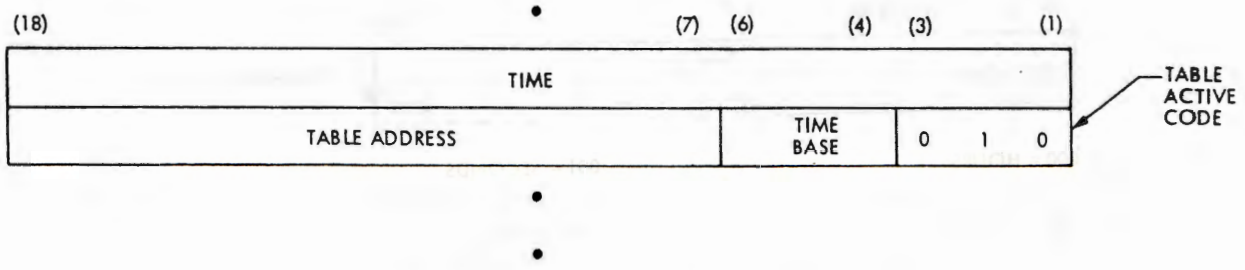


Figure 3-22. Table Activate

record (para. 3.2.2.4) routine, and that routine generates the proper DSS start and stop commands to effect the actual positioning. Figure 3-23 illustrates this positioning event.

7) DSS Record: This pseudo-event allows recording to be accomplished in segments where the number of, length, and spacing between the segments may be specified. This event is illustrated in Figure 3-24, and is discussed in detail in para. 3.2.2.4.

8) Dummy Event: An event word that contains all zeros is allowed as an event. No action will be taken when this event is processed, however.

9) Scan Platform Position: This pseudo-event allows positioning of the scan platform either incrementally (relative to its present position), or absolutely. Either or both axes may be

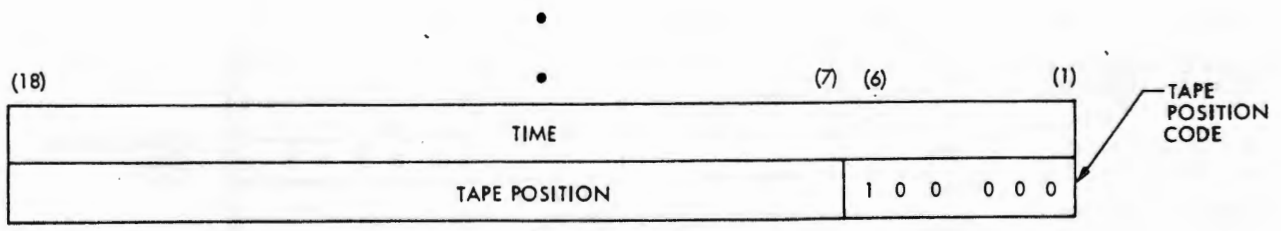
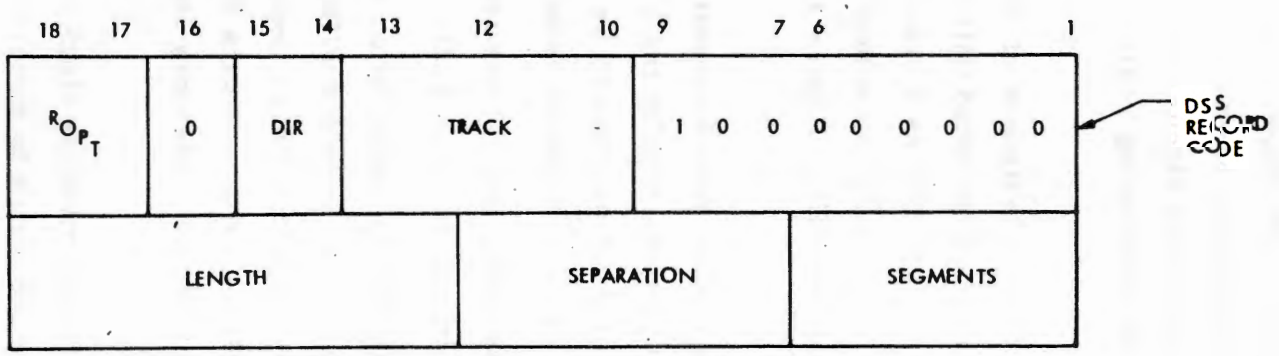


Figure 3-23. Tape Position

72



ROPT: 00 = READY MODE BETWEEN SEGMENTS
11 = RECORD AT 7.2 kbps BETWEEN SEGMENTS

DIR: 00 = NO CHANGE
01 = FORWARD
10 = REVERSE
11 = NO CHANGE

TRACK: 0000 = NO CHANGE
0001 = TRACK 1
,
,
1000 = TRACK 8

LENGTH = NUMBER OF FDS FRAMES PLUS 1
SEPARATION = NUMBER OF FDS FRAMES BETWEEN SEGMENTS
SEGMENTS = NUMBER OF SEGMENTS

Figure 3-24. DSS Record Event

controlled. In the case of a two-axis slew, the slews are sequential, not parallel. See Figure 3-25, and para. 3.2.2.6 for details of this event.

10) Scan Platform "Mosaic" and "Scan": Two algorithms of scan platform movement for science acquisition have been designed as CCS pseudo-events. These events are discussed in detail in the section on the scan platform positioning routine (para. 3.2.2.6).

11) Sun Sensor Bias Update (Single): A single update of the sun sensor bias control in the AACS is accomplished with this pseudo-event. Processing of this event causes a transfer to the sun sensor bias routine (para. 3.2.2.3) for actual generation of the required bias command to AACS. Figure 3-26 illustrates this event.

12) Sun Sensor Bias Update (Multiple): This pseudo-event allows the sun sensor bias to be updated periodically by the CCS. It provides for a specified number of fixed updates to be issued by the CCS with a specified time spacing between updates.

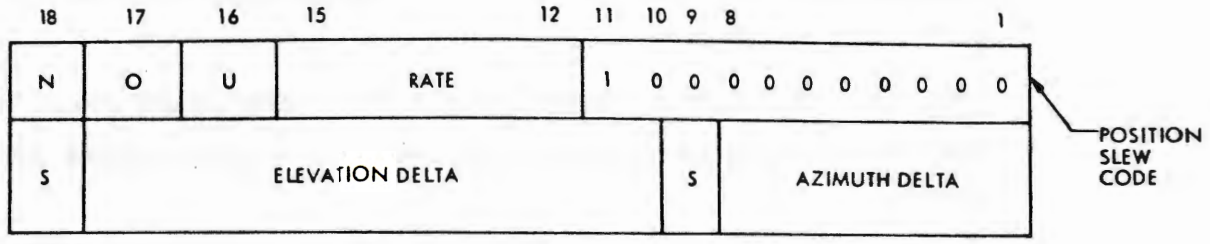
This event is illustrated in Figure 3-27, and is discussed in detail in the sun sensor bias routine (para. 3.2.2.3).

13) FDS/AACS Memory Load: The ability to load FDS or AACS memory from the CCS memory has been included within the CCS software by means of a pseudo-event. This event is illustrated in Figure 3-28 and utilizes command data formats discussed in the command decode routine. Memory loading details may be found in para. 3.2.3.2.

Although the CCS does not have a hardware minutes clock, there is a minutes time-base within this routine which is generated by software. Whenever a time within a time/event table is specified in minutes, this routine converts it to seconds by multiplying by 60 and it is then placed under the control of the seconds clock.

Figure 3-29 illustrates the flow diagram for this routine.

75

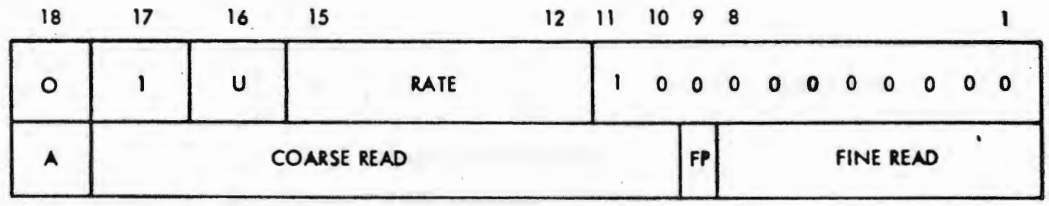


N: NOT USED

RATE, U: PLATFORM RATE = $\frac{1}{(1 + 15\,U) - RATE}$ DEG/SEC

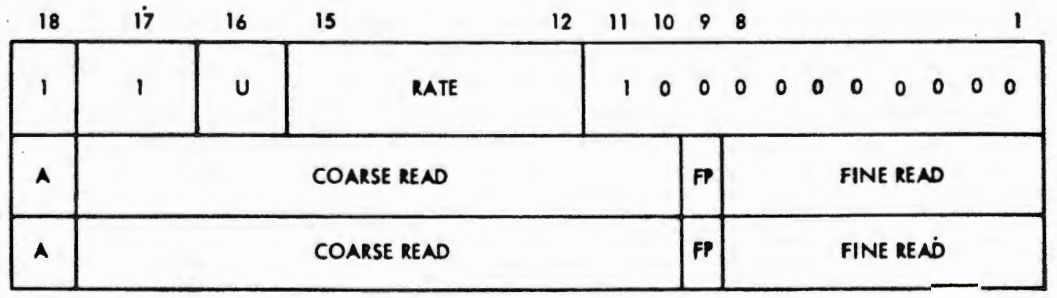
S: 0 = POSITIVE INCREMENT
1 = NEGATIVE INCREMENT (2'S COMPLEMENT)

Figure 3-25A. Incremental Platform Slew



FINE READ
RATE, U: SAME AS ABOVE
A: 0 = ELEVATION
1 = AZIMUTH
FP: 0 = FINE POT A
1 = FINE BOT B

Figure 3-25B. Absolute Slew - Single Axis



RATE, U, A, FP: SAME AS ABOVE

Figure 3-25C. Absolute Slew - Dual Axis

76

3.2.2.2.1.2 Constraints. No more than thirty time/event tables may be active at any time.

This routine does not check for syntax errors in any event, but assumes that all events are coded correctly when loaded into the CCS.

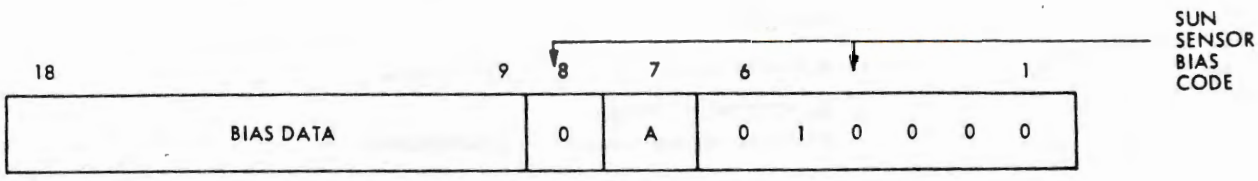


Figure 3-26. Sun Sensor Bias (Single)

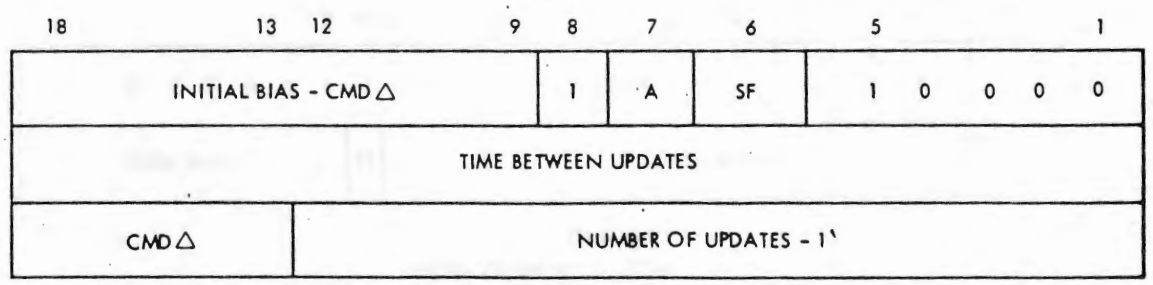


Figure 3-27. Sun Sensor Bias (Multiple)

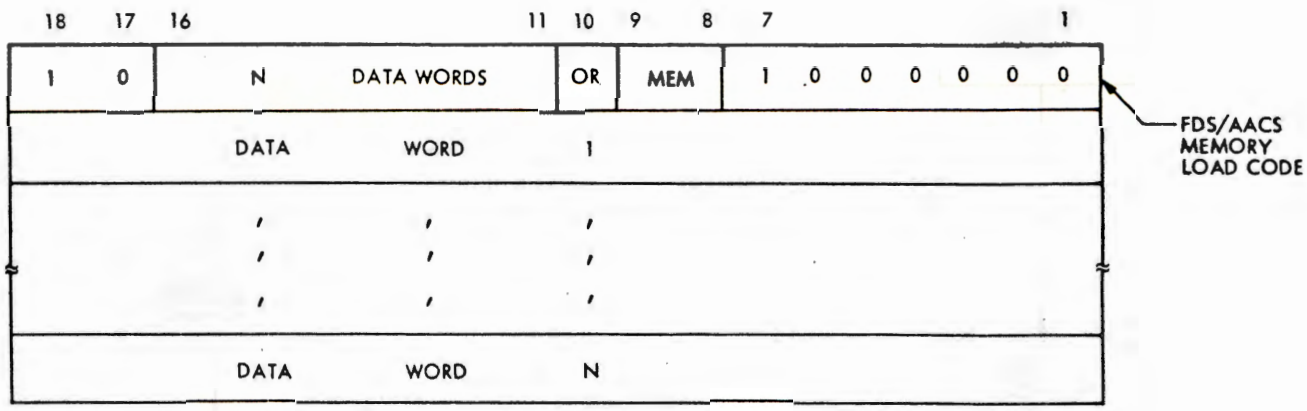


Figure 3-28. FDS/AACS Memory Load

3.2.2.2.2 Sun Sensor Bias (ANTUPD). This routine processes either a single sun sensor bias update request or a multiple update request.

3.2.2.2.2.1 Description. This routine is typically entered from the sequence support routine after that routine has encountered a sun sensor bias update request within an active time/event table. The bias update request is processed to determine if it is a single update or a multiple update (algorithm). (See Figures 3-30 and 3-31.)

If it is a single update, the bias data is extracted from bits 18 to 9 of the request word. This data is then used to generate the appropriate AC7SSB command (see MJS77-3-290).

7.8

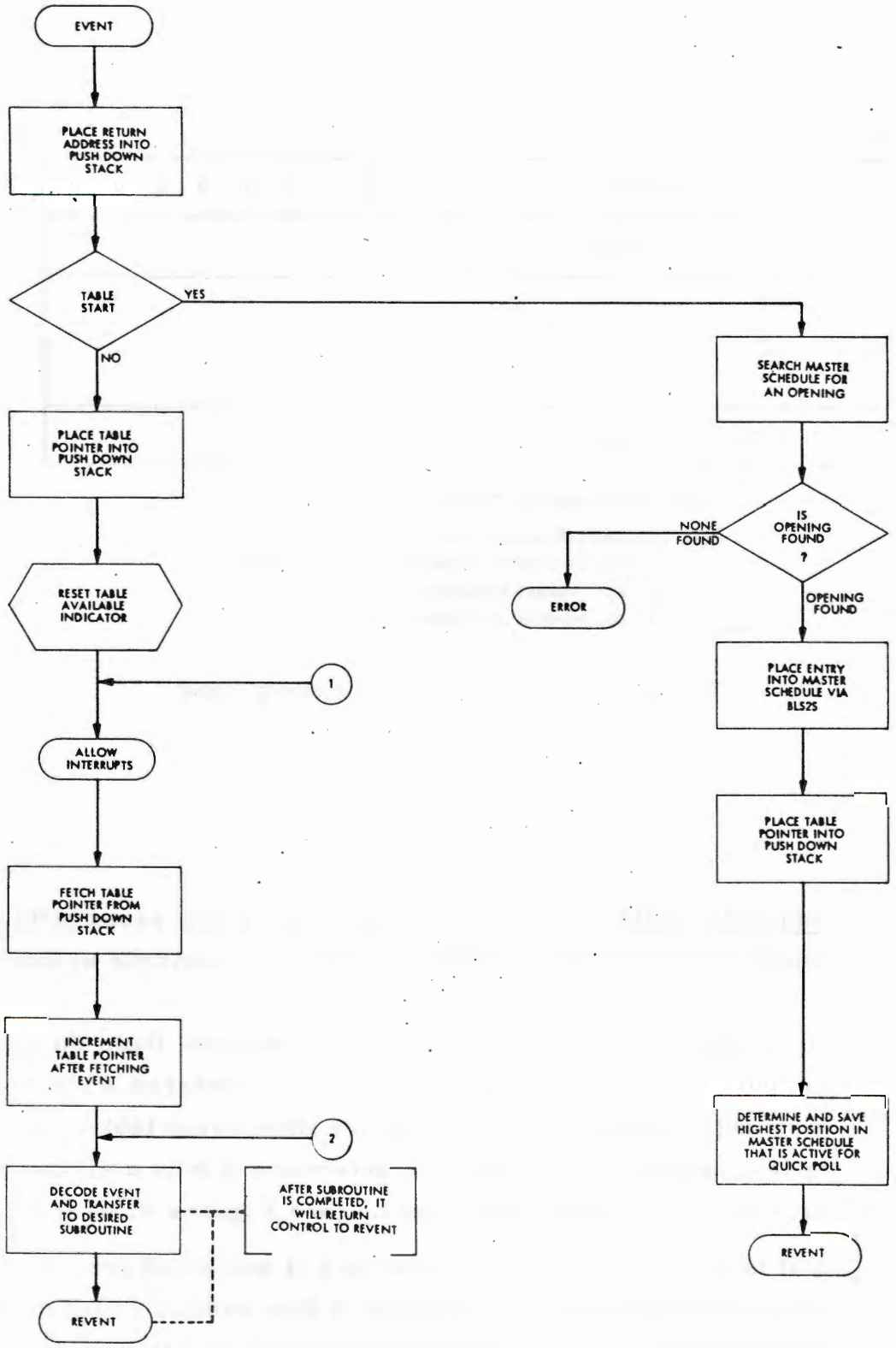


Figure 3-29. Flow Diagram of the Routine TARMEX (Sheet 1 of 5)

79

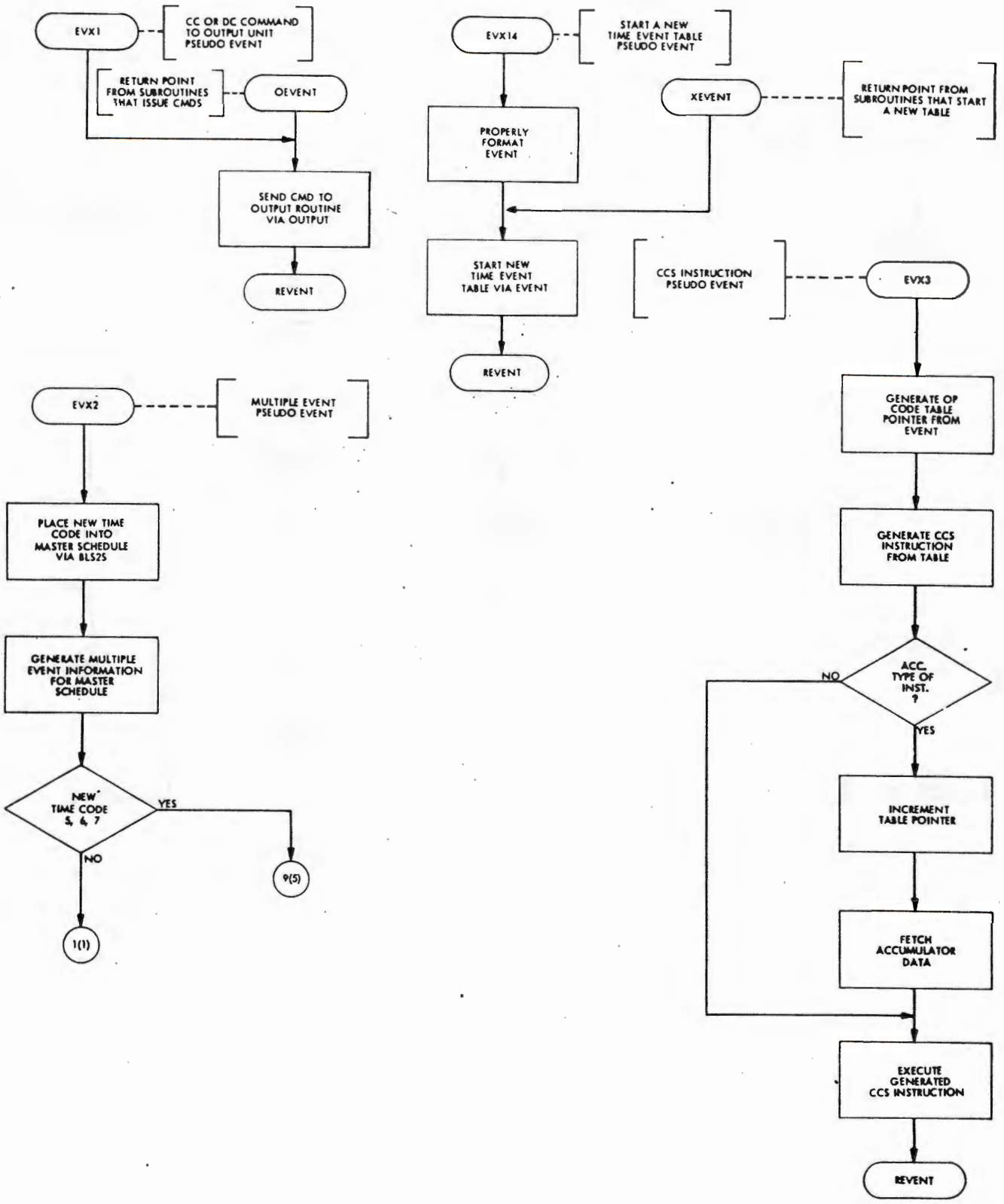


Figure 3-29. Flow Diagram of the Routine TARMEX (Sheet 2 of 5)

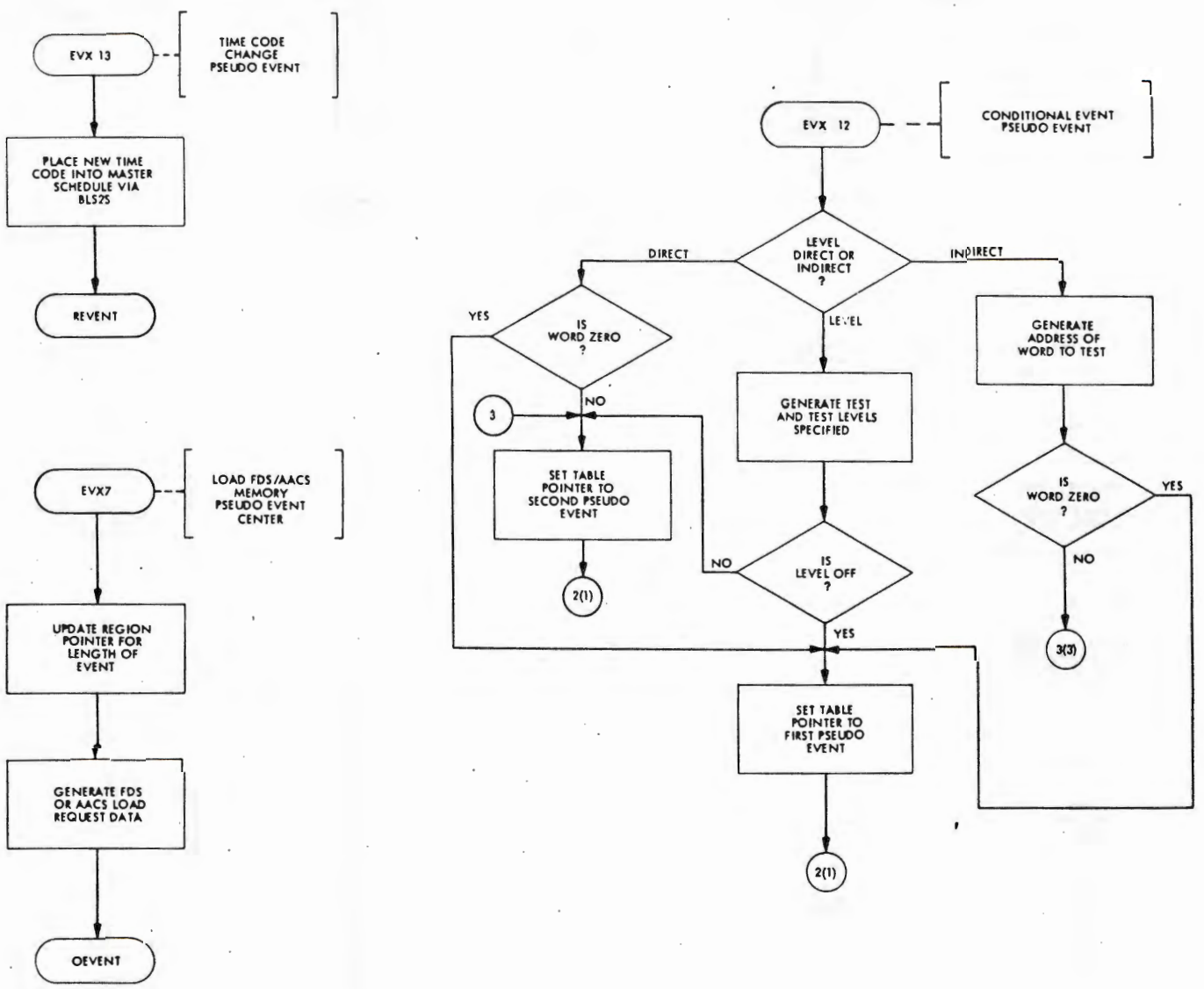


Figure 3-29. Flow Diagram of the Routine TARMEX (Sheet 3 of 5)

81

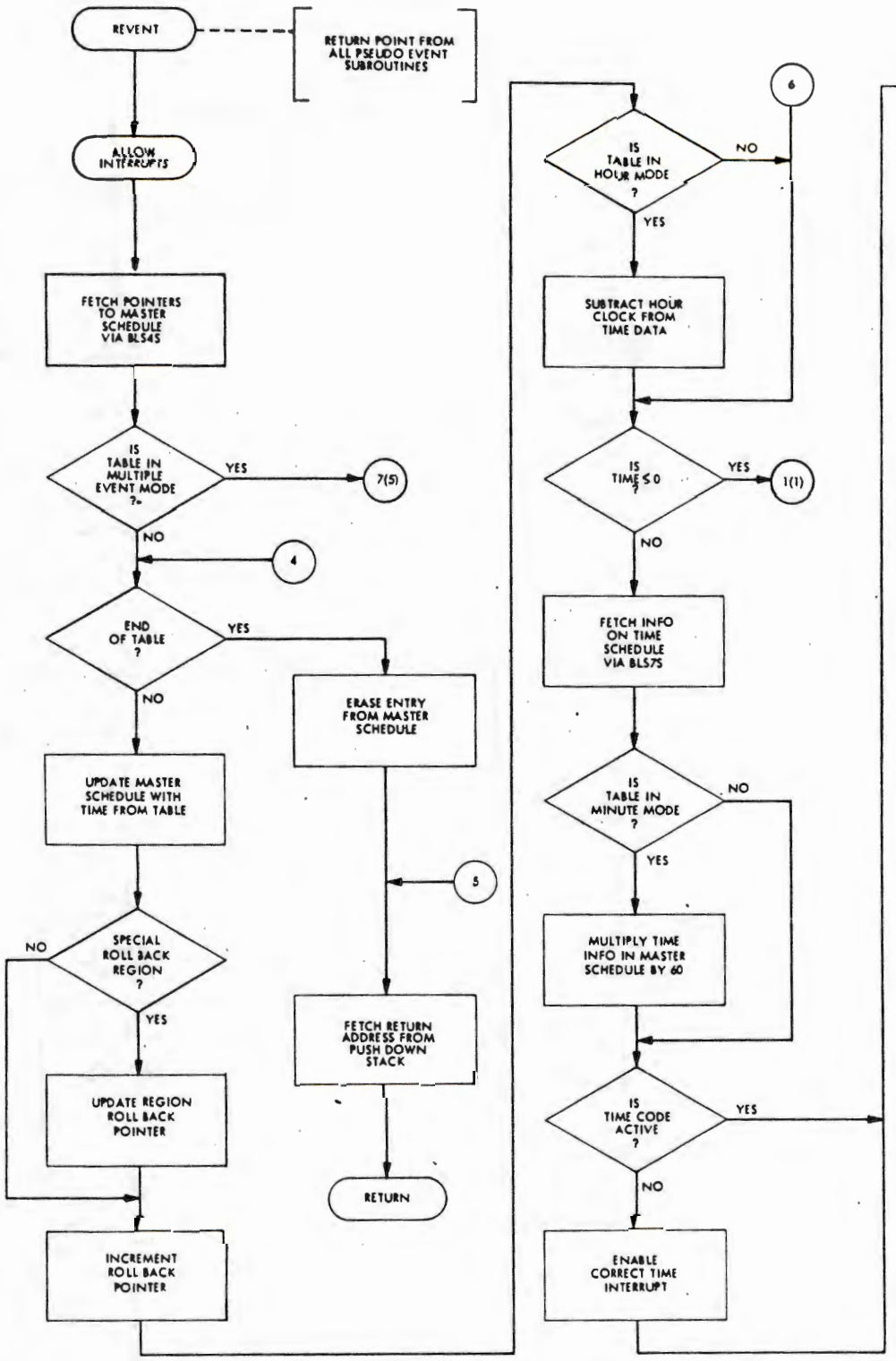


Figure 3-29. Flow Diagram of the Routine TARMEX (Sheet 4 of 5)

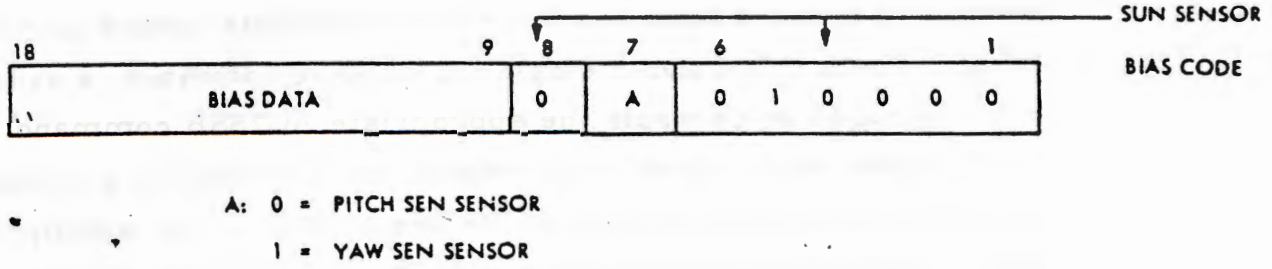


Figure 3-30. Sun Sensor Bias (Single)

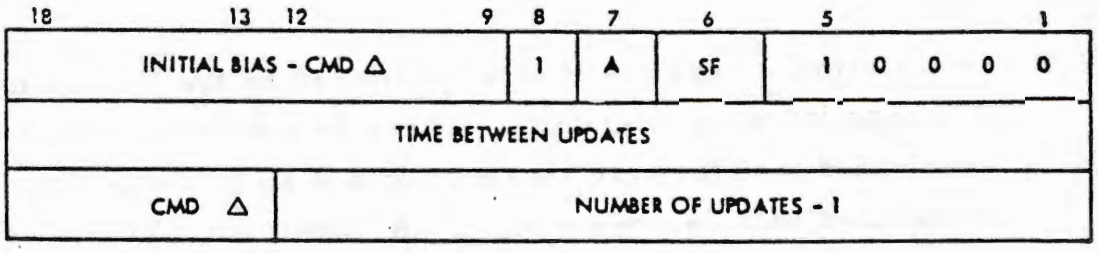


Figure 3-31. Sun Sensor Bias (Multiple)

8.4

If the request is a multiple update, the routine will extract the initial bias position, time between updates, bias increment, and number of updates from the three-word multiple update pseudo-event. From this data, a single bias update command is assembled and processed to generate the appropriate AC7SSB command. Then a time/event table is assembled and activated to provide for the assembly and generation of the next update at the appropriate time. This time/event table is re-activated after each update until a counter underflows to indicate the last update.

The capability to keep a multiple update cycling but inhibit outputting of the bias updates has been included. This disabling is accomplished by changing the jump location within a transfer instruction (ANTW03) in read/write memory.

Figure 3-32 illustrates the flow diagram for this routine.

3.2.2.2.3 DSS Slew and Record (DTRPRC). This routine is used to perform two functions.

(1) To keep track of tape position by processing DSS indexing interrupts.

(2) To position the recorder for a record or playback sequence.

3.2.2.2.3.1 Description. Tape position is monitored by the CCS by processing DSS tape indexing interrupts. There is a decrementing interrupt associated with forward tape motion and an incrementing interrupt associated with reverse motion. A "tic-count" (TIC) is associated with each absolute position of the tape. At the beginning of tape (start of tracks 1, 3, 5, 7 or end of tracks 2, 4, 6, 8) the tic-count is set at 200 by the CCS. As the tape moves forward towards the end of tape (end of tracks 1, 3, 5, 7 or start of tracks 2, 4, 6, 8) the tic-count is decremented by the decrementing index interrupt from DSS. When the tape moves in the reverse direction the tic-count is incremented by the incrementing index interrupt. Therefore, as the tape moves back and forth between

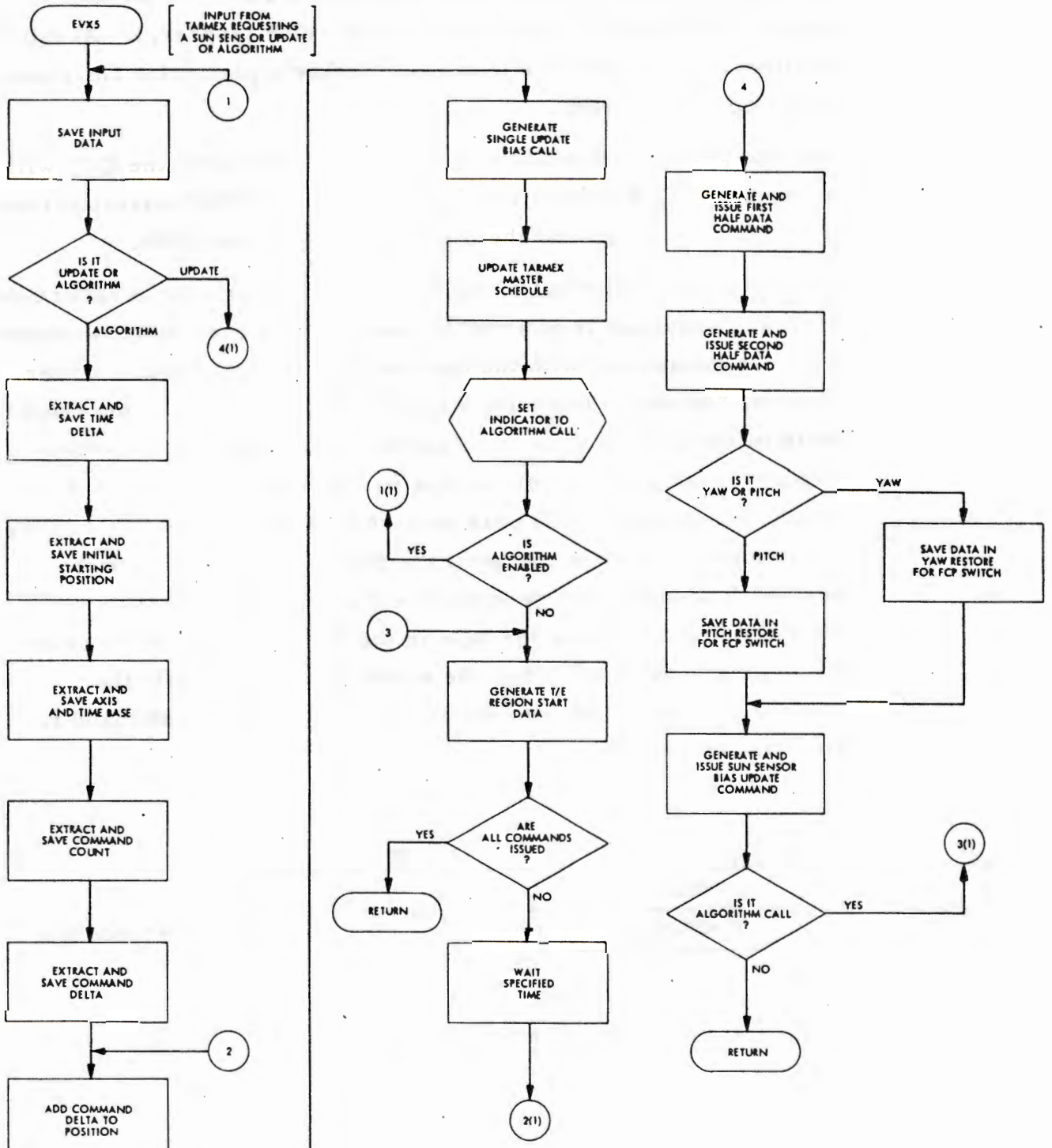


Figure 3-32. Flow Diagram of the Routine ANTUPD

8b

beginning and end of tape (odd-numbered track), the tic-count moves back and forth between 200 and 3400* (approx.). Absolute positions on the tape are thus specified by a particular tic-count between 200 and 3400.

For the purpose of initializing the tic-count of 200, the CCS will assume a BOT has been reached when a BOT/EOT interrupt from the DSS is received and the tic-count is less than 1820.

Positioning (or slewing) of the tape recorder prior to a record or playback sequence is effected by specifying a particular tic-count which is associated with the desired position. A tape recorder position request is typically a pseudo-event within a time/event table in the CCS as shown in Figure 3-33. When the sequence support routine encounters a tape position pseudo-event in a time/event table, it will pass on that tape position to this routine. This routine will then compare the present tic-count with the desired tick-count and generate the appropriate tape recorder slew command to move the tape in the proper direction to reach the desired position. When the actual tic-count equals the desired tic-count, the routine sends out a "ready" command to stop the tape recorder.

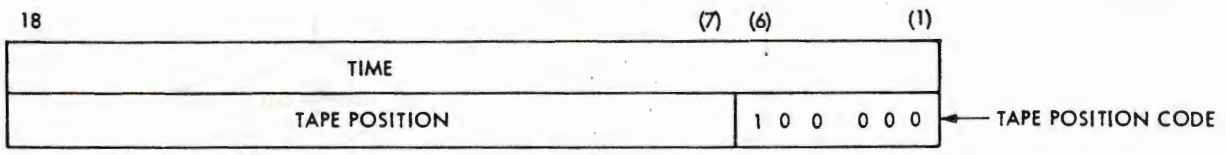


Figure 3-33. Tape Position Request

*The actual end of tape tic-count is adjusted for each spacecraft prior to launch.

If the desired tape position is further than 12 tics away from the present position, the playback routine will command the recorder to slew at the high record rate (115 Kbps) until the tape is within 12 ticks, whereupon the routine will command the DDS to the normal slew rate (7.2 Kbps).

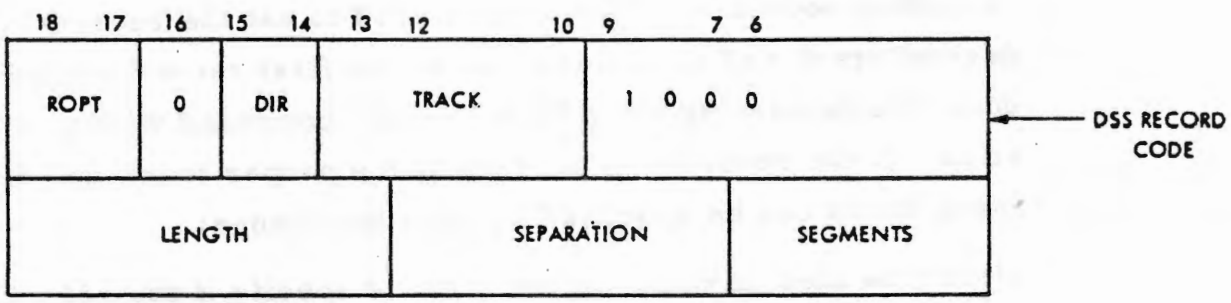
For recording data on tape, this routine can control the tape recorder in a segmented manner with data inputted as shown in Figure 3-34. Word-one of the record request contains the direction of tape movement and tape track-number at the start of the recording sequence. This data is used to set the recorder to the desired track and direction prior to the first record segment. It does this by sending the DTR a "ready" command with this information. If the recorder is already at the proper track and direction, these fields can be specified as zero (no change).

Word-one also contains an indicator to specify if the recorder should be stopped entirely (ready mode) between segments or continue recording at a slower rate (7.2 Kbps). Word two contains information relative to the segmented recording proper. The length of each segment in FDS frame starts (plus one) is specified with a maximum record length of 62 frames.

Also specified is the separation between segments – again expressed in FDS frame starts with a maximum of 63. And finally, the number of segments is specified with a maximum of 63.

The flow diagram for this routine is illustrated in Figure 3-35.

3.2.2.2.3.2 Constraints. Whenever a direction change of the DSS is to occur (record or playback), the DSS should be commanded to the ready mode first, time allowed for rundown, and then commanded in the opposite direction. This procedure will allow CCS to process the DSS indexing signals properly. If a direction change is commanded while the tape is moving, the DSS indexing logic will switch instantaneously and start outputting indexing pulses associated with the new direction. However, the tape requires a finite time to rundown and change direction which means that the first few indexing pulses after a direction change has been commanded will not correspond to the direction that the tape is actually moving.



ROPT: 00 = READY MODE BETWEEN SEGMENTS
11 = RECORD AT 7.2 KBPS BETWEEN SEGMENTS

DIR: 00 = NO CHANGE
01 = FORWARD
10 = REVERSE
11 = NO CHANGE

TRACK: 0000 NO CHANGE
0001 = TRACK 1
1000 = TRACK 8

LENGTH = NUMBER OF FDS FRAMES PLUS 1

SEPARATION = NUMBER OF FDS FRAMES BETWEEN SEGMENTS

SEGMENTS = NUMBER OF SEGMENTS

Figure 3-34. DSS Record Request

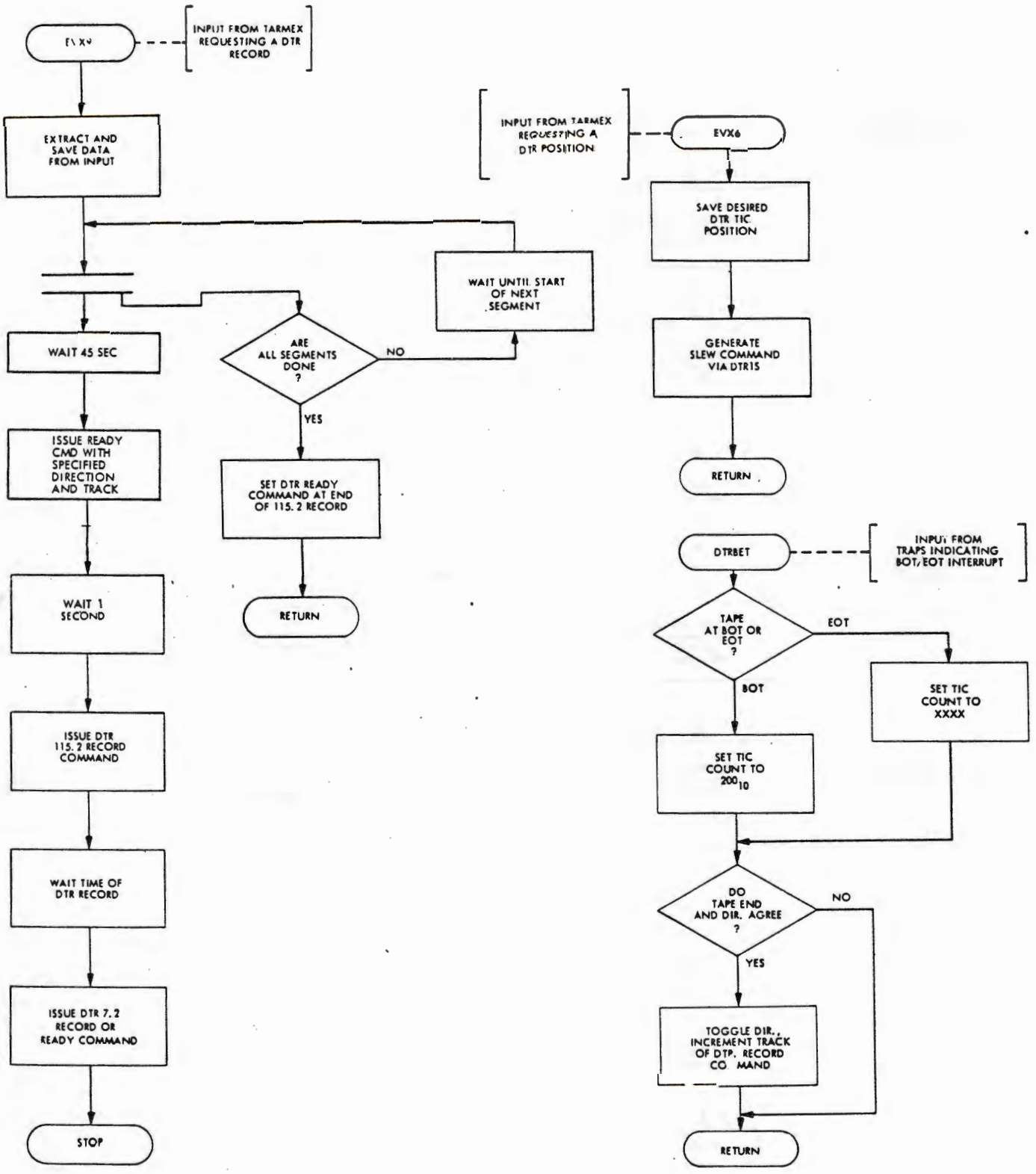


Figure 3-35. Flow Diagram of the Routine DTRPRC (Sheet 1 of 2)

Change #3
9/8/80

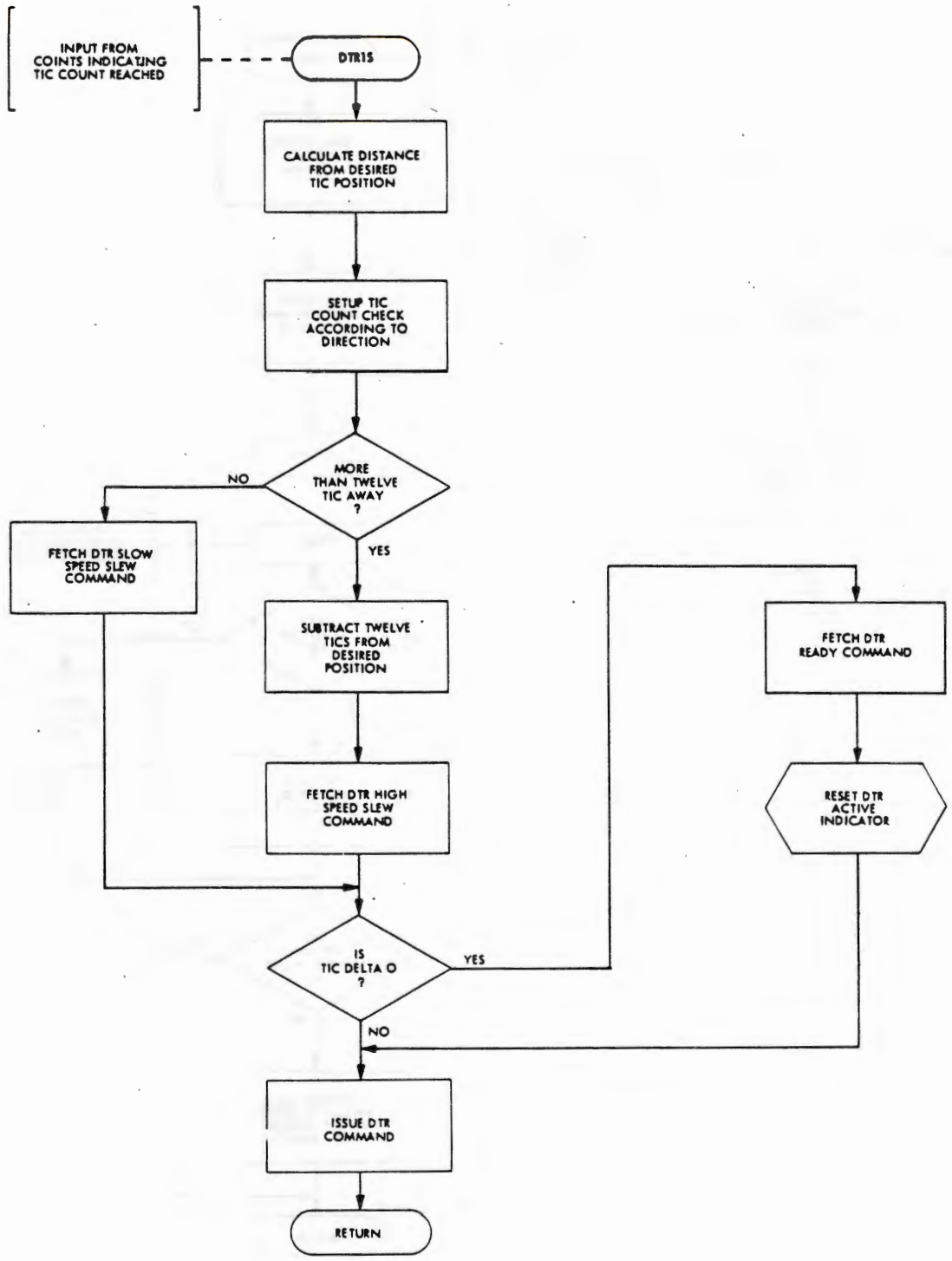


Figure 3-35. Flow Diagram of the Routine DTRPRC (Sheet 2 of 2)

This causes the CCS to think that the tape is at a different position than it actually is. Stopping the recorder before changing direction prevents this anomaly. This constraint is explicitly stated here because although the CCS controls some record sequences, other sequences are typically started and stopped apart from this routine (sequence software, or by FDS). Therefore, particular attention should be paid this anomaly when developing record sequences.

3.2.2.2.4 Launch Hold/Reset (LHRST). The purpose of the launch hold/reset routine is to provide a means of initializing the CCS prior to launch; and to provide for re-initialization if a launch hold occurs after T-8.5 minutes.

3.2.2.2.4.1 Description. The launch hold/reset routine is activated by ground command at T-8.5 minutes. If a launch hold occurs after T-8.5 minutes, another ground command must be sent to stop the routine. When the hold ends and time is recycled back to T-8.5, the activation ground command must be sent again to restart the sequence.

When activated, the routine initializes the CCS as follows:

1. A portion of the global variable region is cleared to zero. However, before the clearing operation starts, a two second delay is provided to allow the MDS to drop lock so that no primary command syncs will occur. The memory clear function is of such duration that a primary command error would occur if command sync inputs were to occur.

A branch to the "RESET" subroutine within the ERROR routine occurs to effect additional initialization as follows: (Items 2 through 10)

2. The processor clamps are applied.
3. The output units are disabled.
4. All interrupts except primary command sync are reset and disabled.

92

5. The self-test subroutine is enabled.
6. Key indicators are reset.
7. The command decoding routine is terminated.
8. The telemetry buffer pointers are initialized to point to the beginning of the telemetry buffer.
9. The event output buffer is cleared to zero and the pointers initialized.
10. The sequence support routine is reset.

Next, the launch hold/reset routine initializes global variables for launch as follows:

11. The interrupt mask registers are set to <u>enable</u> the following inputs:
    (a) Primary and secondary command inputs
    (b) Internal interrupt 10
    (c) The self-test initiate interrupt (internal A)
    (d) The CCS hour interrupt
    (e) AACS power codes and the power low-voltage inputs
    (f) Interprocessor (output unit) communication interrupts

and <u>disable</u> the following inputs:

(a) Internal interrupt F (unused)
(b) 100 PPS
(c) DTR indexing interrupts
(d) Internal interrupts D and E
(e) 1 PPS
(f) Internal interrupt C (checksum)
(g) FDS low rate bit sync
(h) Output unit interrupts (available and reject)

(i) ISS frame start inputs

(j) RFS failure inputs

(k) MIRIS standby supply input

(l) Backup FDS ISS frame start input

(m) DTR BOT/EOT input

12. The TIC count is initialized to 1920_{10}.

13. Enable Power Low Voltage and set launch option.

14. The command counter is reset.

15. The software event counter is reset.

16. The master automatic counter is reset.

17. The lock-change counter is reset.

18. Various routine variables are initialized.

19. The command loss routine time-to-activate is initialized.

20. The prime processor for AACS power code echo commands is identified.

After all the above resets and initializations have occurred the routine then does the checksum. If the checksum is performed correctly, the CCS hardware clock is reset at the time of the next FDS frame start so that the CCS hour clock will be synchronous with FDS timing. This is done for future sequencing convenience. The routine then generates the following subsystem enable or disable commands:

DC3X ENABLE COMMAND TO MDS

DC4X ENABLE COMMAND TO PWR

DC6X ENABLE COMMAND TO FDS

DC7X ENABLE COMMAND TO AACS

DC16X ENABLE COMMAND TO DSS

94

DC2XR DISABLE COMMAND TO RFS
DC8XR DISABLE COMMAND TO PYRO
DC77XR DISABLE COMMAND TO MAM
DC5AR CCS TOLERANCE DETECTOR ENABLE
CC16C4026 DSS TAPE COMMAND
CC7GY3 AACS GYRO CONTROL

The above mix of enable/disable commands reflect the philosophy of disabling command outputs where potential relay chatter during launch could cause an inadvertent command, and enabling those subsystems where no such condition exists.

At this point, the routine initiates the AACS heartbeat check algorithm, starts the launch sequence, and the CCS clock check. The CCS clock check compares CCS time with FDS frame starts in order to detect a fast CCS which could cause the launch sequence to occur prematurely. The CCS clock check has each processor count 48 seconds between FDS frame starts. If a processor is able to count more than 48 seconds between frame starts, it asks the other processor if it detected the same condition. If the other processor counts extra time between frame starts also, then the CCS assumes that the FDS is slow, and both halves of the CCS continue the launch sequence. If only one half of the CCS is fast relative to the FDS, then that half will shut down by transferring to ERROR.

If the checksum is not performed correctly the routine will stop by transferring to WAIT. If there is a ground command sent to stop the CCS, the CCS will stop executing this routine, disable rollback, and then transfer to the ERROR routine.

Figure 3-36 illustrates the flow diagram for this routine.

3.2.2.2.5 Scan Platform Positioning (TV and Scan Support) (SCNPLT). The scan platform positioning routine generates scan platform positioning commands in support of individual platform position commands and basic science sequence algorithms. Individual platform

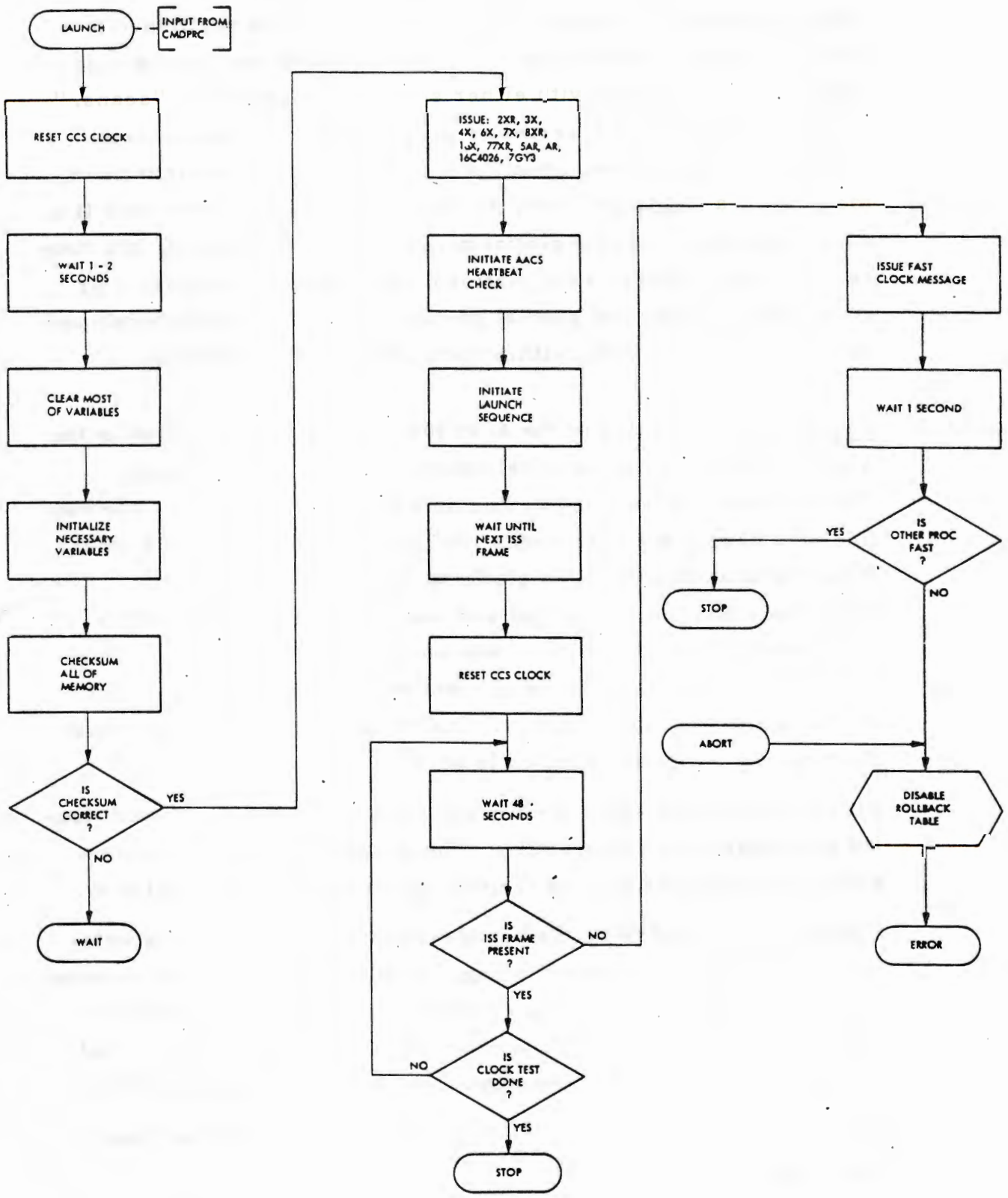


Figure 3-36. Flow Diagram of the Routine LHRST

9b

commands direct that the platform be moved to a new position relative to the old (incremental slew) or that it be positioned to a specific position (absolute slew). The platform positioning algorithms are associated with either science "mosaics" or "scans." A mosaic consists of a series of parallel platform movements intended to map a given area. A scan consists of platform movement along a single line with the option of departing from that line periodically for imaging platform positions. The spacing and time relationships between each platform position within a mosaic or scan are variable, and special pseudo-events have been developed to store these variables within the CCS sequence software.

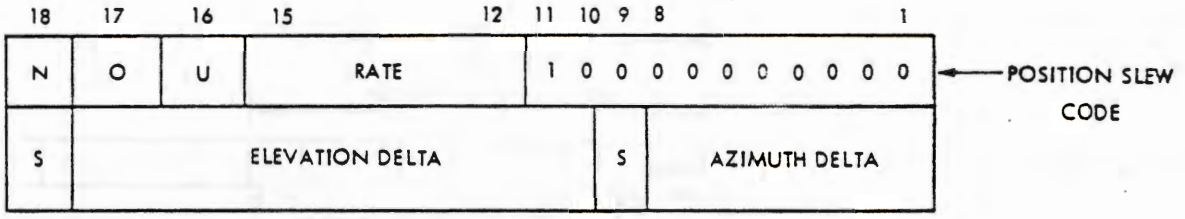
3.2.2.2.5.1 Description. Pointing of the scan platform is accomplished in the AACS by utilizing feedback/telemetry potentiometers (pots). There is one coarse and two fine pots per platform axis. The two fine pots have their dead zones 180° out of phase with each other. When commanding the scan platform to a particular position (absolute slew) the coarse pot and one of the fine pots for each axis must be provided with a reference value. The platform may also be moved relative to its present position (incremental slew) by the use of the fine pots only. Due to the range of the fine pots, incremental slews are limited to ±3.6°.

Figure 3-37A illustrates the pseudo-event associated with positioning the platform incrementally. The pseudo-event contains slew rate information as well as the fine pot references with polarity.

Figure 3-37B illustrates the pseudo-event associated with single-axis absolute platform positioning. In addition to rate information and fine pot reference it also contains the necessary coarse pot reference and the axis. If the coarse pot reference is 127 (ID of 255) the normal AC7SPK command is converted into an AC7SPL.

Figure 3-37C shows the same thing with an added word for two-axis positioning.

The actual coded commands that transfer the necessary information to AACS are shown in Figure 3-38.

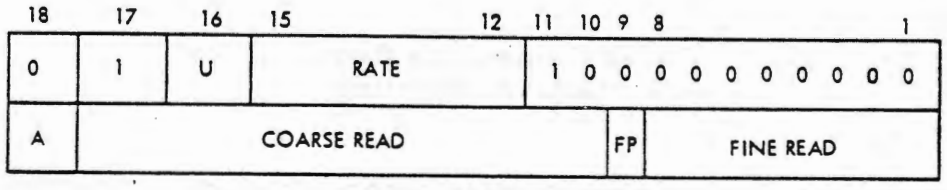


N: NOT USED

RATE, U: PLATFORM RATE = $\frac{1}{(1 + 15\,U) - \text{RATE}}$ DEG/SEC

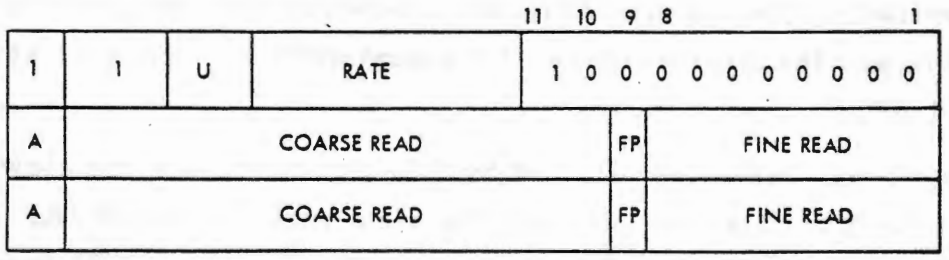
S: 0 = POSITIVE INCREMENT
1 = NEGATIVE INCREMENT (2'S COMPLEMENT)

Figure 3-37A. Incremental Platform Slew



RATE, U: SAME AS ABOVE
A: 0 = ELEVATION
1 = AZIMUTH
FP: 0 = FINE POT A
1 = FINE POT B

Figure 3-37B. Absolute Slew — Single Axis



RATE, U, A, FP: SAME AS ABOVE

Change #6
7/26/82

Figure 3-37C. Absolute Slew — Dual Axis

98

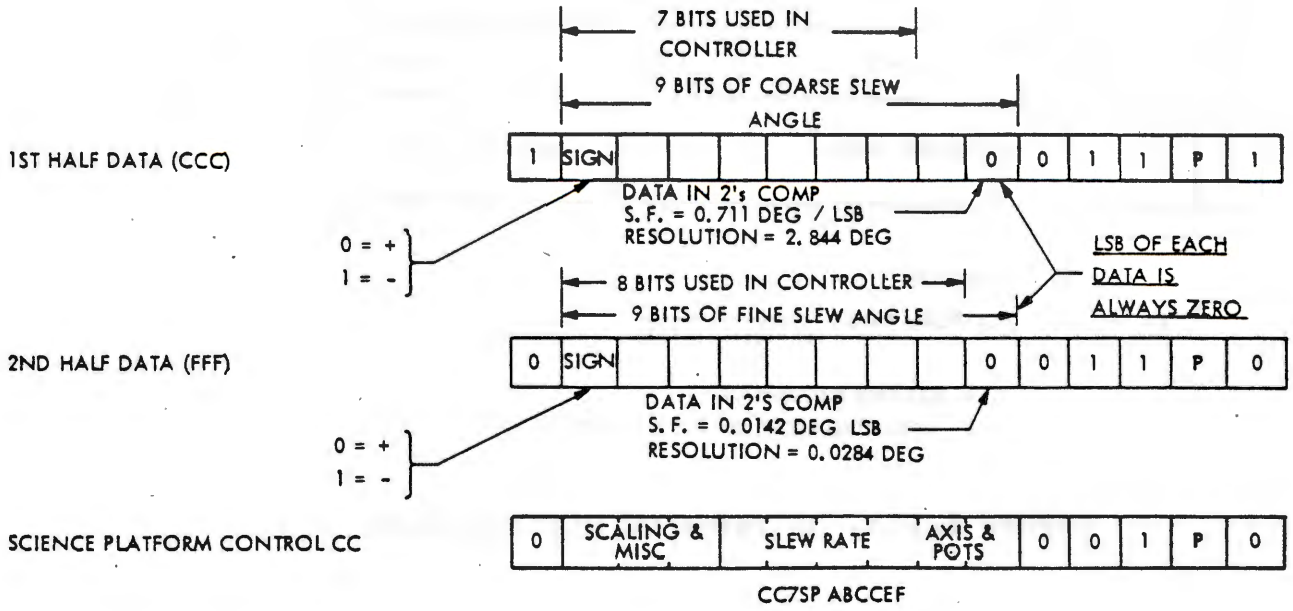
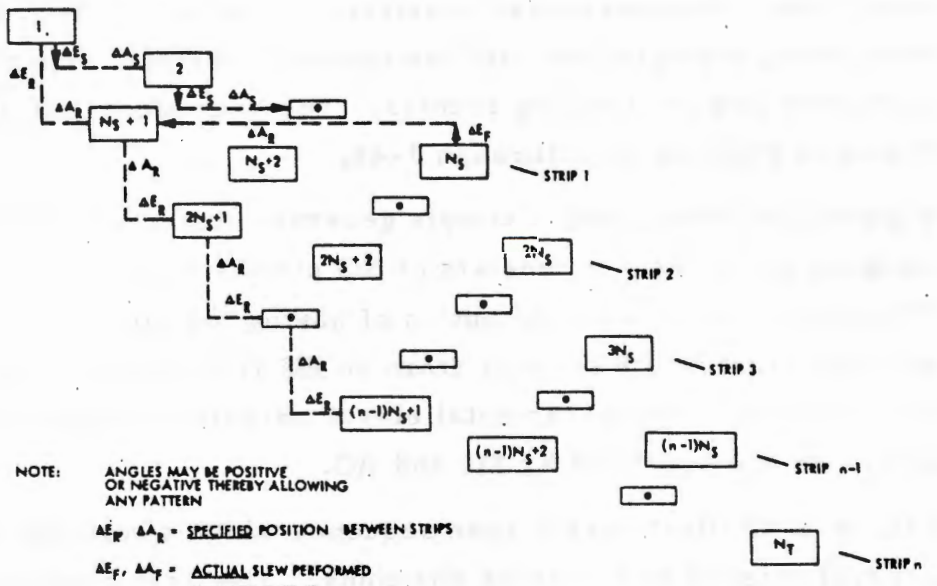


Figure 3-38. Coded Commands Required for Science Platform Pointing

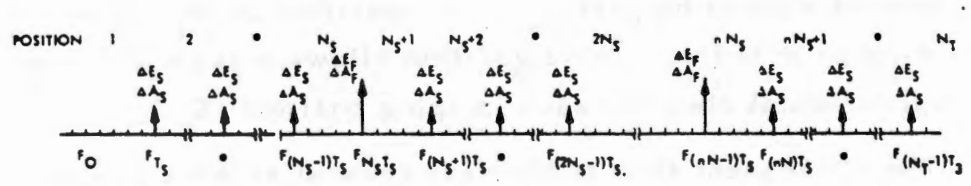
Two science platform positioning algorithms are supported by this routine. One is the mosaic algorithm which generates a series of parallel platform positions (or strips) for science observation. It is illustrated in Figure 3-39.

The number of platform positions in each complete strip is identified as variable NS, while the total number of positions in the mosaic is identified as variable NT. The platform position difference between each position within a strip is identified as ES and AS for the elevation and azimuth angle differences respectively. Similarly, the position difference between the last position of each strip and the first position of the next strip is identified as EF and AF.

Additional variables allowed by this algorithm are the slew rate for the ES/AS slews (R1) and the slew rate for the EF/AF slews (R2). Finally, variable TS specifies the number of ISS frame times between each position in the mosaic.



MOSAIC - POSITION PLANE



NOTES: N_S = NO. OF POSITIONS IN FIRST STRIP = 511
N_T = TOTAL NO. OF POSITIONS IN MOSAIC = 511
T_S = NO. OF FDS FRAME TIMES (F) BETWEEN POSITIONS IN MOSAIC

MOSAIC - TIME PLANE

Figure 3-39. Science Mosaic

190

The storing of the variables for a mosaic within the CCS is accomplished by assembling them into a special pseudo-event that is stored within the sequence table area of the CCS memory. Figure 3-40 illustrates the science mosaic pseudo-event.

The other algorithm supported by this routine is the science scan algorithm. A science scan consists of a series of platform positions along a single line with the option of periodic departures from that line for imaging frames. The scan algorithm is illustrated in Figures 3-41 through 3-44.

Figure 3-41 illustrates a simple general science scan without imaging positions. It consists of NG platform positions spaced TG seconds apart with the option of having the slew to the second position start SOPT seconds from an ISS frame start minus two seconds. The incremental elevation/azimuth slews between positions are specified by EG and AG.

Figure 3-42 illustrates a scan sequence which alternates between general science and imaging positions. The first platform slew is a slew to the first imaging position and occurs nine seconds after an ISS frame start (as do all imaging slews). The incremental slew to be performed is specified by E_1/A_1 for the first imaging position. The algorithm allows a separately specified incremental slew for each imaging position ($E_n A_n$).

The subsequent slew to the next general science position occurs 2 seconds before the next ISS frame start time. In an integrated scan, the general science positions cannot occur more often than every 48 seconds (T_G = 48), and may be farther apart when preempted by an imaging position. The SOPT option is available in an integrated scan but can only be associated with slews between one general science position and another. In the case of alternating positions the SOPT option is not available.

18	17 — 12	11	10	9 — 8	7	6	5	4 — 1
TIME								
NT-2			SOPT		R2	R1	1 0 0 0	
0	NS-1		1 0 0 0 0			TS		FS
S	ES		S		AS			
S	EF		S		AF			

R1 = ES/AS SLEW RATE } 00 = HI RATE, 01 = MED RATE
R2 + EF/AF SLEW RATE } 10 = LO RATE, 11 = VERY LO RATE

SOFT = ES/AS SLEW DELAY OPTION:

00 = 0 SEC DELAY, 01 = 4 SEC DELAY
10 = 8 SEC DELAY, 11 = 12 SEC DELAY

FS = TIME BASE FOR TS: 0 = EVEN SECONDS, 1 = ISS FRAMES

TS = NO. OF EVEN SECONDS OR FRAME TIMES BETWEEN POSITIONS (≤ 31)

NS = NO. OF POSITIONS IN EACH COMPLETE STRIP (≤ 512)

NT = TOTAL NO. OF POSITIONS IN MOSAIC (≤ 513)

S = POLARITY OF INCREMENTAL SLEW: 0 = POS, 1 = NEG*

ES = NO. OF PLATFORM ELEVATION STEPS BETWEEN POSITIONS (≤ 255)

AS = NO. OF PLATFORM AZIMUTH STEPS BETWEEN POSITIONS (≤ 255)

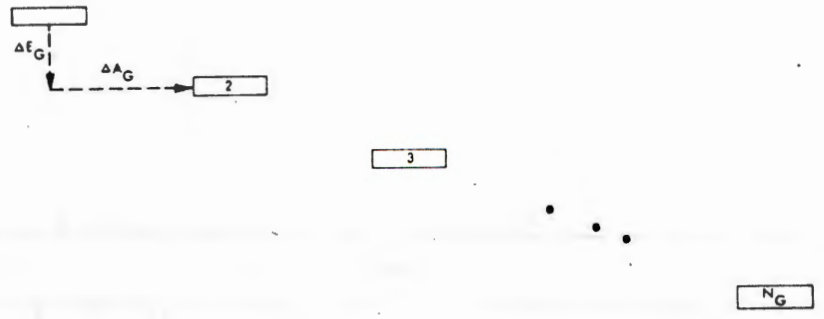
EF = NO. OF PLATFORM ELEVATION STEPS BETWEEN END OF ONE STRIP AND BEGINNING OF NEXT STRIP (≤ 255)

AF = NO. OF PLATFORM AZIMUTH STEPS BETWEEN END OF ONE STRIP AND BEGINNING OF NEXT STRIP (≤ 255)

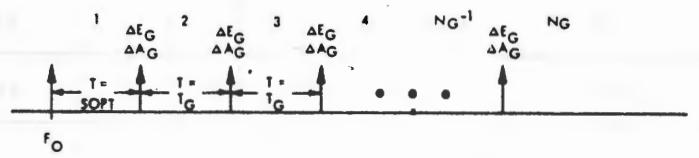
(*) NEGATIVE STEPS ARE IN 2'S - COMPLIMENT

Figure 3-40. Science Mosaic Pseudo-Event

9 Oct 2



SCAN - GENERAL SCIENCE ONLY - POSITION PLANE



F_O = FDS FRAME START - 2 SECONDS
SOPT = SCAN START DELAY OPTION = 0, 4, 8 OR 12 SECONDS
T_G = SCAN SPACING = 2, 4, 6, 8, 12, 16, 24, OR 48 SECONDS
N_G = TOTAL NUMBER OF SCAN POSITIONS = 2 TO 513

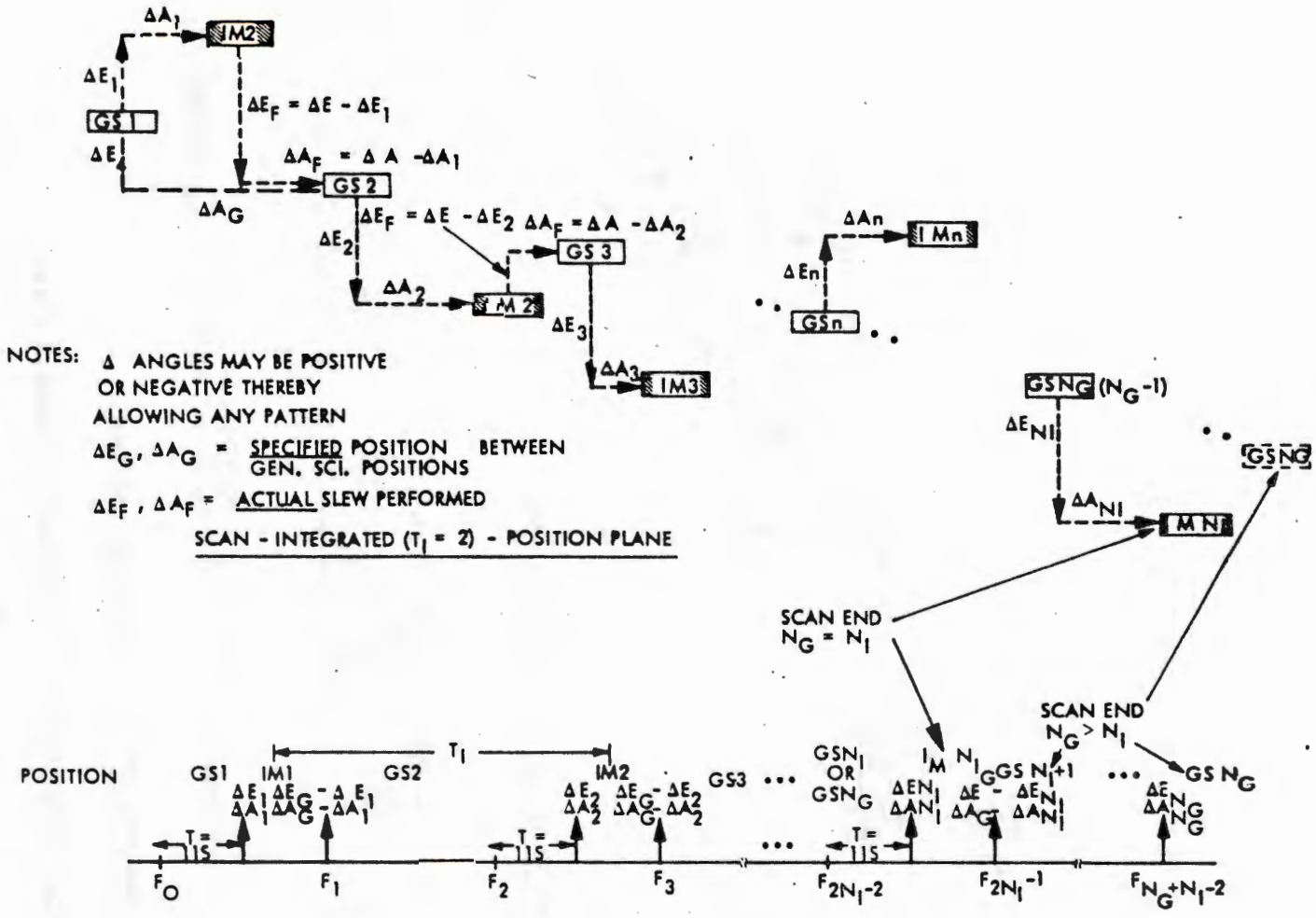
SCAN - GENERAL SCIENCE ONLY - TIME PLANE

Figure 3-41. Science Scan — General Science Only

The slew from the imaging position to the general science position (EF/AF) must be calculated by the CCS by subtracting the incremental slew performed to arrive at the previous imaging position (E_n/A_n) from the specified incremental slew EG/AG. The specified increment, EG/AG, is the platform position difference between general science positions when there is not an intervening imaging slew.

The number of frame times between imaging positions is specified by variable TI, where TI ≤ 16.

Figure 3-43 illustrates the general case for an integrated science scan.



NOTES: Δ ANGLES MAY BE POSITIVE OR NEGATIVE THEREBY ALLOWING ANY PATTERN

ΔE_G, ΔA_G = SPECIFIED POSITION BETWEEN GEN. SCI. POSITIONS

ΔE_F, ΔA_F = ACTUAL SLEW PERFORMED

SCAN - INTEGRATED (T_I = 2) - POSITION PLANE

F = FDS FRAME START - 2 SECONDS

T_I = NO. OF FRAME STARTS BETWEEN IMAGING POSITIONS ≤ 16

N_G = TOTAL NO. OF GEN. SCI POSITIONS

N_I = TOTAL NO. OF IMAGING POSITIONS

$(N_G+N_I \leq 513)$

SCAN-INTEGRATED (T_I = 2) - TIME PLANE

Figure 3-42. Science Scan — Integrated (T_I = 2)

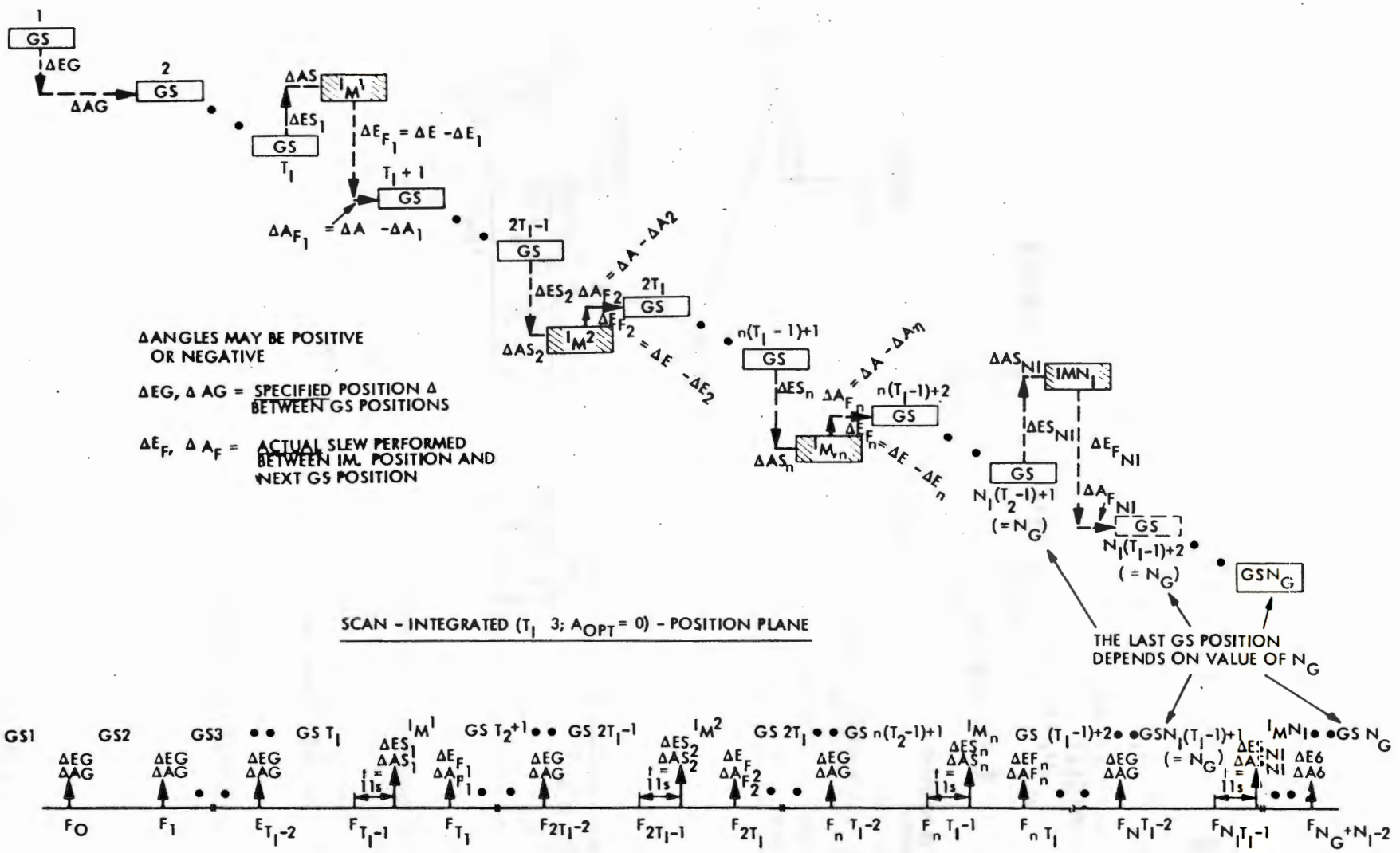


Figure 3-43. Integrated Science Scan — General Case

Figure 3-44 illustrates an integrated scan incorporating the AOPT option. This option provides for long platform slews to an imaging position by starting the slew one frame time earlier than normal to allow an extra 48 seconds for the imaging slew. It should be noted that this advance option replaces a general science slew with an imaging slew, and therefore, there are NI fewer science positions with the AOPT option than without it.

As with the science mosaic, the variables associated with the scan algorithm are stored within a sequence table as a pseudo-event. Figure 3-45 illustrates the science scan pseudo event.

When the sequence support routine encounters a mosaic or scan pseudo-event within a sequence table, the sequence support routine transfers the pseudo-event to the scan platform positioning routine and informs it as to which type pseudo-event it is. This routine then extracts the data from the pseudo-event, and generates the required platform slew commands at the proper times.

The flow diagram for the scan platform positioning routine is illustrated in Figure 3-46.

3.2.2.2.6 Tandem and Turn Command Support Routine (TRNSUP). This routine is typically used to monitor the correct operation of the AACS during a spacecraft commanded turn maneuver. The maneuver may be checked in the tandem mode where both halves of the CCS must agree on the proper commands, start-time, and turn duration, or in the non-tandem mode wherein one or both halves merely check for proper turn duration. Use of this routine for command checks (maneuver or other) requires that the command or commands for each tandem-checked function be assembled into a multiple event block with a centisecond time-base. Also, all commands must be assembled as non-execute commands with Bit 1=0. Additionally all discrete commands must have Bit 2=1.

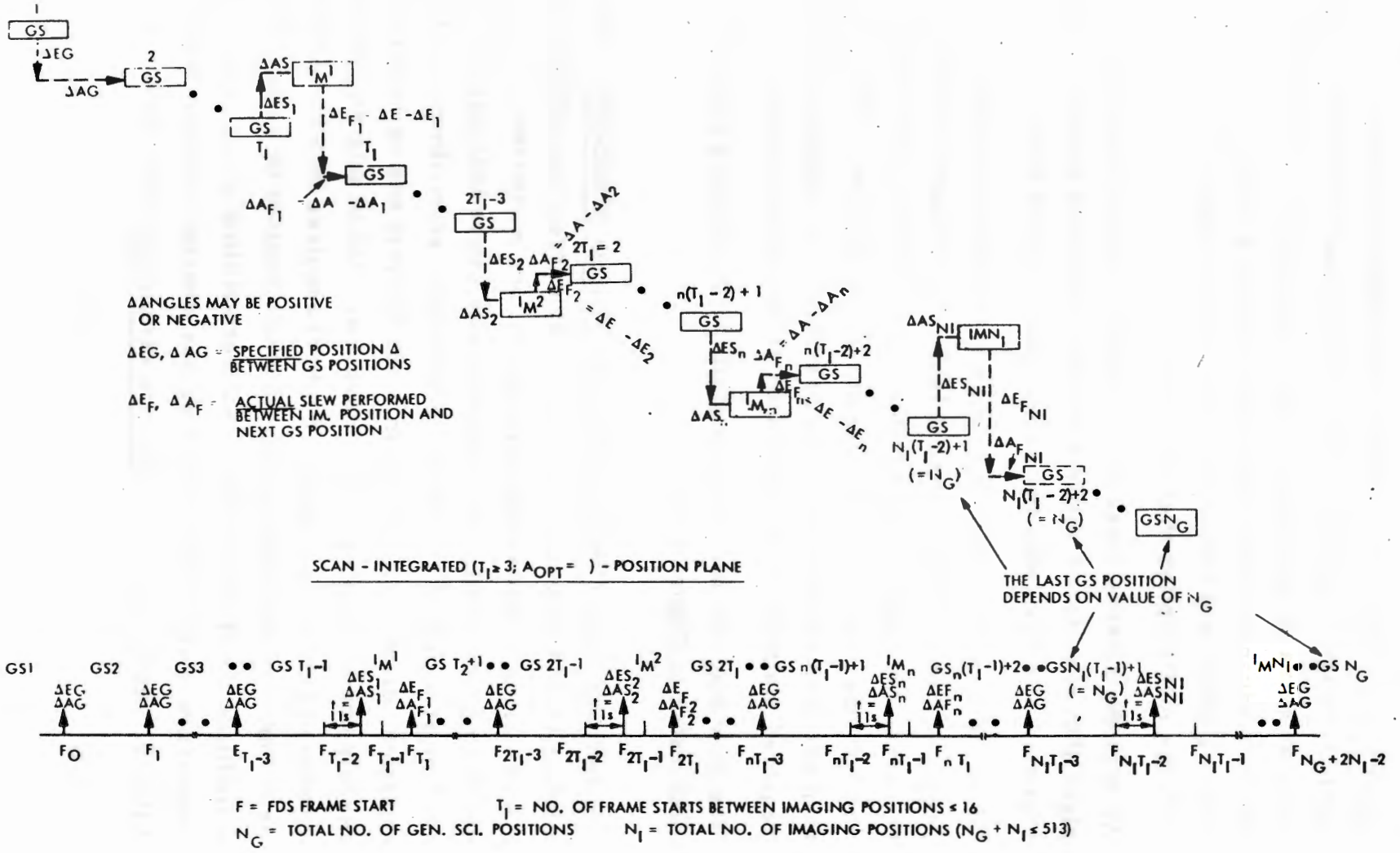
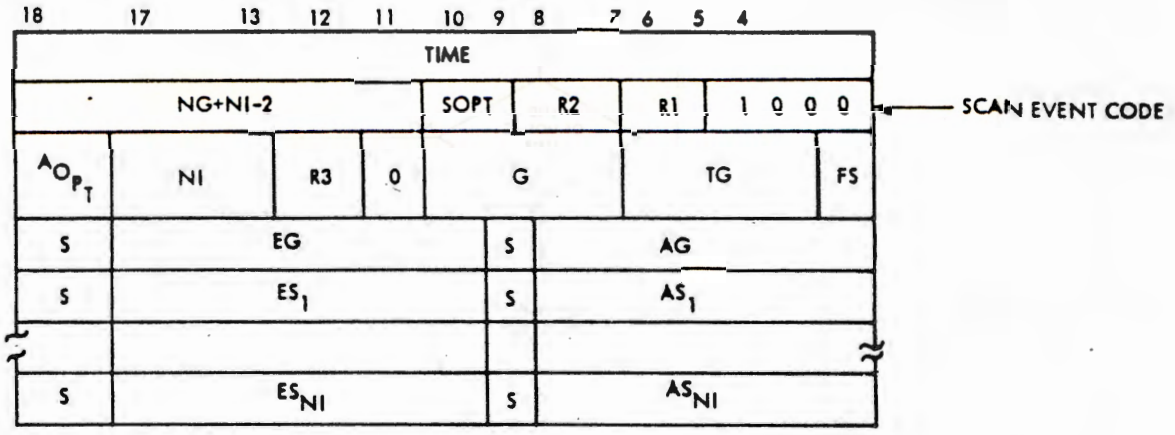


Figure 3-44. Integrated Science Scan — Special Case



R1 = EG/AG SLEW RATE 00 = HI RATE, 01 = MED RATE

R2 = ES/AS SLEW RATE 10 = LO RATE, 11 = VERY LO RATE

R3 = EF/AG SLEW RATE 0 = HI RATE, 1 = LO RATE

G = 48 (TI-1-AOPT)/TG-1

EG = NO. OF PLATFORM ELEVATION STEPS BETWEEN GENERAL SCIENCE (≤ 255)

AG = NO. OF PLATFORM AZIMUTH STEPS BETWEEN GENERAL SCIENCE POSITIONS (≤ 255)

ES_n = NO. OF PLATFORM ELEVATION STEPS BETWEEN THE PRECEDING GENERAL SCIENCE POSITION AND THE nTH IMAGING POSITION

AS_n = NO. OF PLATFORM AZIMUTH STEPS BETWEEN THE PRECEDING GENERAL SCIENCE POSITION AND THE nTH IMAGING POSITION

S = SLEW POLARITY: 0 = POSITIVE: 1 NEGATIVE

NG = TOTAL NUMBER OF GENERAL SCIENCE POSITIONS

TG = NO. OF EVEN SECONDS OR ISS FRAMES BETWEEN CONSECUTIVE UNINTERRUPTED GENERAL SCIENCE POSITIONS

FS = TIME BASE FOR TG: 0 = EVEN SECONDS, 1 = ISS FRAMES

SOPT = EG/AG SLEW DELAY OPTION:

00 = 0 SECONDS DELAY AFTER FRAME START
01 = 4 " " " " "
10 = 8 " " " " "
11 = 12 " " " " "

NI = TOTAL NUMBER OF IMAGING POSITIONS

TI = NO. OF 48-SECOND FRAMES BETWEEN IMAGING POSITIONS (≤ 16)

AOPT = ES_n/AS_nSLEW ADVANCE OPTION (0 = NO ADVANCE:
1 = ADVANCE SLEW 48 SECONDS)

(*) NEGATIVE STEPS ARE IN 2'S - COMPLIMENT

Figure 3-45. Science Scan Pseudo-Event

109

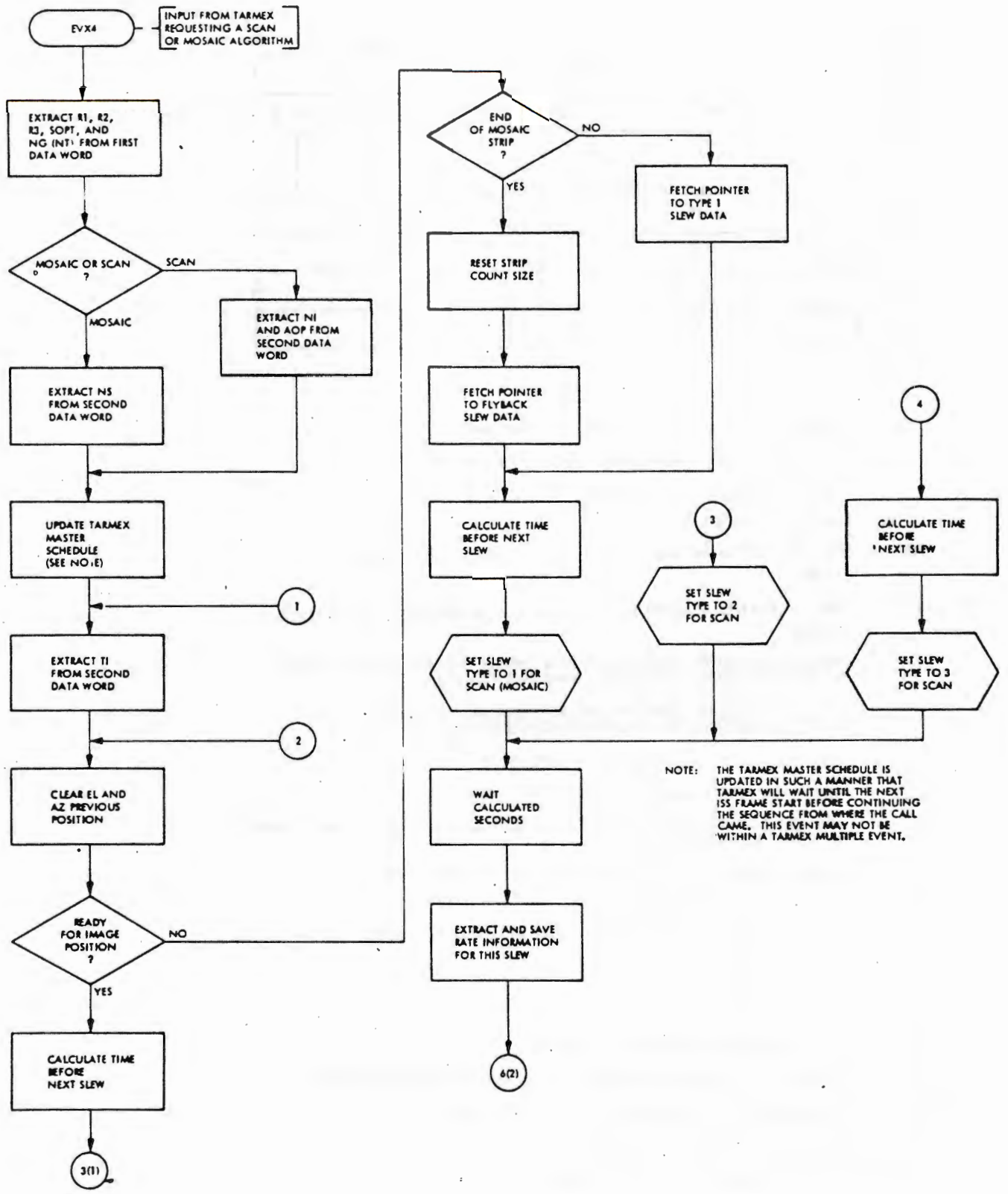


Figure 3-46. Flow Diagram of the Routine SCNPLT (Sheet 1 of 3)

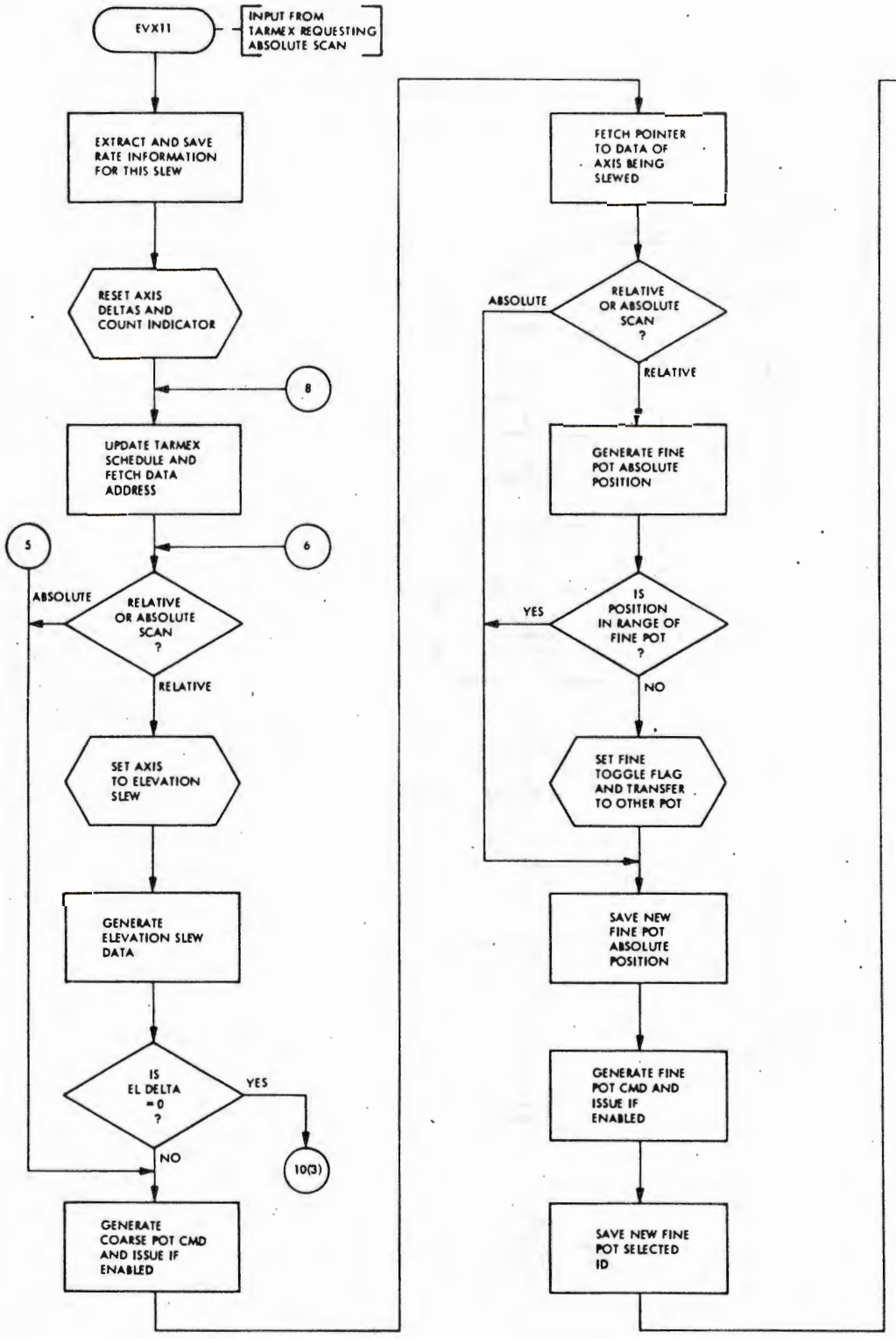


Figure 3-46. Flow Diagram of the Routine SCNPLT (Sheet 2 of 3)

IID

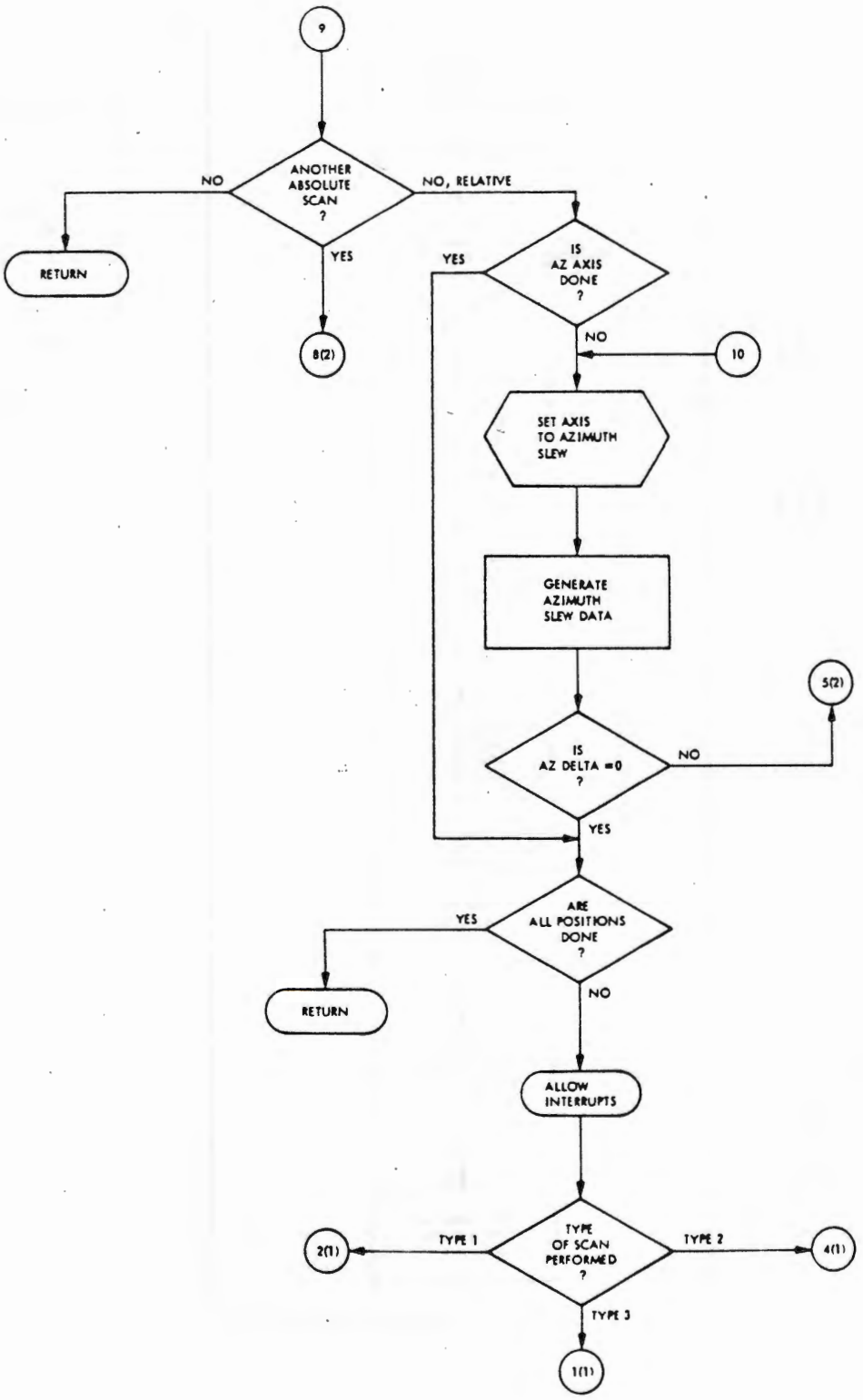


Figure 3-46. Flow Diagram of the Routine SCNPLT (Sheet 3 of 3)

3.2.2.2.6.1 Description.

3.2.2.2.6.1.1 Initialization. This routine is entered at several different times during the execution of a commanded turn maneuver. The first entry (TRNSTA) must occur before a turn start, and serves the purpose of checking the condition of the AACS and resetting certain indicators. The condition of the AACS must be that proper star and sun acquisitions be in effect, and that no "omen" power code has been received. If either of these conditions exist the maneuver routine will be terminated. In addition, if an "omen" power code has been received, or if the other processor has had a tolerance detector trip without this processor experiencing one, the abort sequence of Table 3 will be executed. The abort sequence is not executed for a loss of acquisition because the normal CCS response to an acquisition loss will accomplish approximately the same thing. Once an omen or loss of acquisition has been received by the CCS, ground intervention is required to remove the indication from the CCS.

3.2.2.2.6.1.2 Tandem Command Check. After the initial AACS status check and routine initialization, the maneuver execution may proceed. If the maneuver is to be performed using the tandem check capability, entry for each tandem command will be at either the master entry point (TRNMSA) or the slave entry point (TRNSLV). Either entry causes the AACS status and the other processor's tolerance detector to be rechecked. Next, the routine prepares for the tandem check process by setting up for a maximum of eighteen tests for a tandem compare. The slave processor will then output the tandem command to the master processor for comparison. Then each processor will reduce a tandem test-count word by one by shifting the word left one bit (doubling) and verifying that the count word has not been shifted to zero. The routine now forces the sequence support routine to stay in the multiple-event mode and overlays the pointer to the command being checked with a pointer

112

Table 3-3. Maneuver Abort Sequence

1.	CC1DR	Bay 2 Htr Off
2.	CC16AP	DSS On
3.	CC16C40XX or NOP*	DSS record 7.2 k/bs or no command*
4.	SC06BB160100	FDS 1200 b/s, TCM telemetry mode
5.	CC7ML08	Turn abort
6.	CC7ML04 } AC7MDP6	All axes inertial
7.	CC7MD6 }	
8.	Disable rollback	
9.	Wait 6 minutes	
10.	CC7SS1	Sun sensor search enable
11.	Wait 1 second	
12.	CC7CT09	Start Canopus search

*Dependent on mission phase.

Change #6
7/26/82

113

to the check sequence. Both processors now wait for the next check-time determined by the overlayed multiple event block currently active in the sequence support routine. At this point in time, the master is waiting to compare its command with that transferred from the slave, and the slave is waiting for the master to send a command for it to compare.

If the master receives and agrees with the command transferred from the slave within eighteen check-times, it will output the command, first as a non-execute command to the slave processor, and then as an execute command to the AACS or other user. The slave processor will then compare the command it received from the master with the command it originally sent, and if they agree, it will allow the sequence to proceed. If either processor disagrees with the other, the abort sequence of Table 3-3 is executed.

After each processor has finished processing the tandem command, the routine will restore the multiple-event pointer for the tandem sequence in the sequence support routine and return to that routine.

3.2.2.2.6.1.3 Turn Stop Check. Because the AACS determines when a commanded turn should end, the CCS can only verify that the end occurred within a time window. The maneuver sequence must therefore open and close that window.

The window is opened by a transfer to the TRNWOP entry point. This entry point merely checks an indicator word which should be all zeros if the turn-complete power code has not been received (should not have been at this time). The window is closed by a transfer to the TRNWCL entry point. At this time the indicator word should be non-zero, and if it is the maneuver is allowed to proceed. If the indicator is wrong at either check, the abort sequence of Table 3-3 will be executed.

The flow diagram for this routine is illustrated in Figure 3-47.

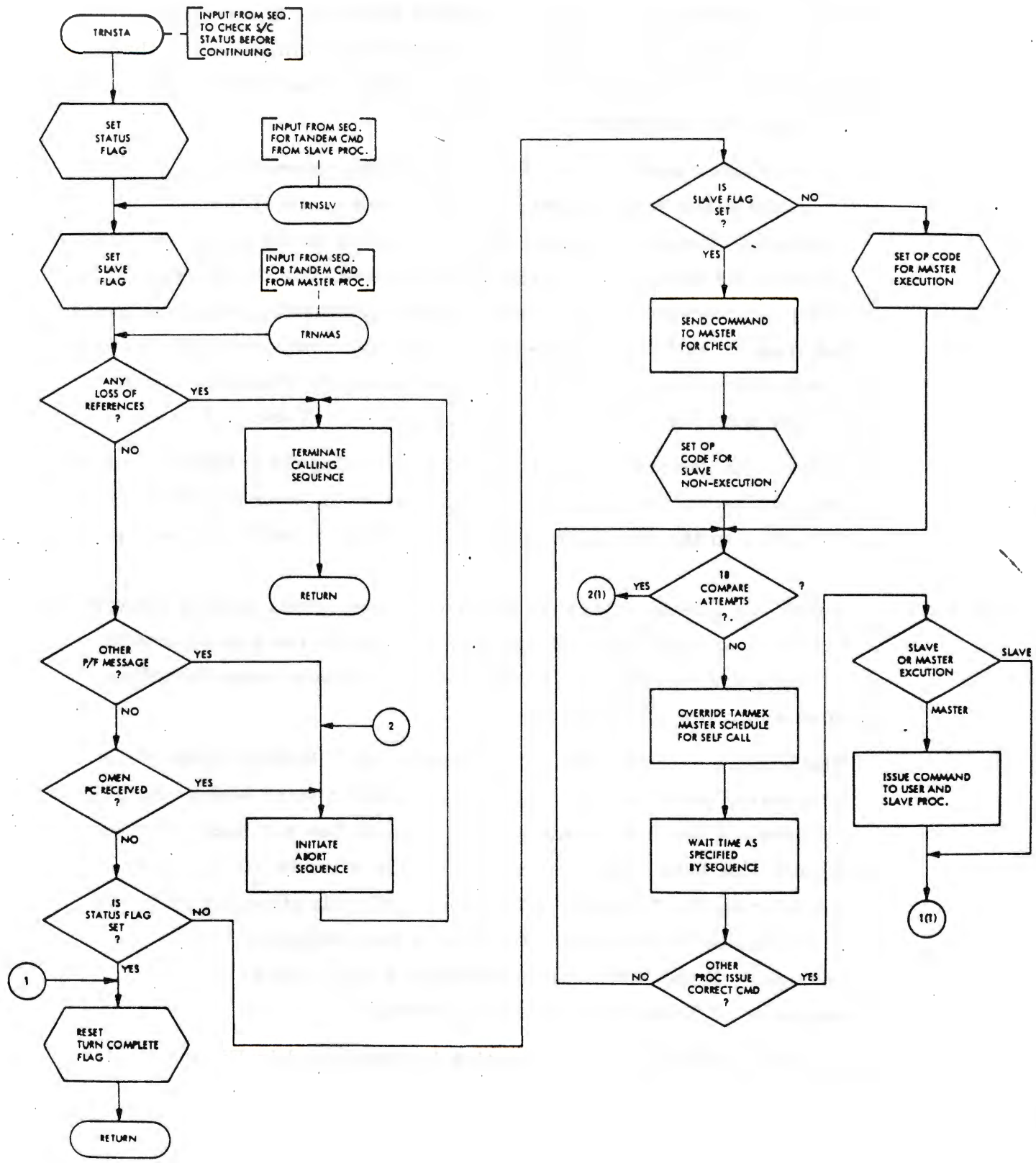


Figure 3-47. Flow Diagram of the Routine TRNSUP (Sheet 1 of 2)

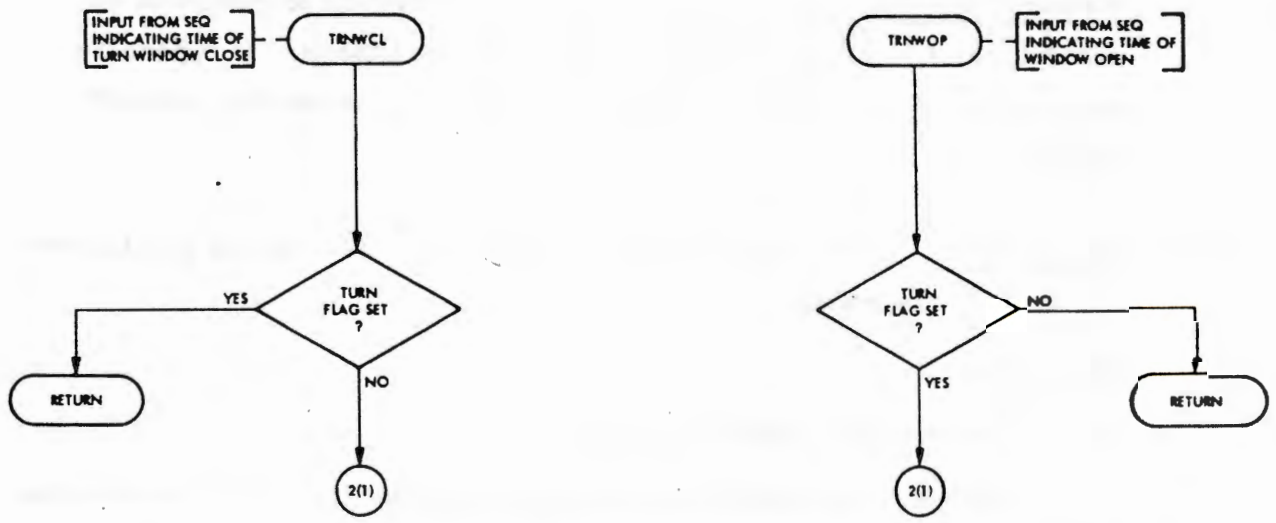


Figure 3-47. Flow Diagram of the Routine TRNSUP (Sheet 2 of 2)

3.2.2.2.6.2 Constraints.

(1) This routine must be entered from a multiple-event pseudo-event from the sequence support routine.

(2) The specified time spacing between commands in that multiple-event block must be in the centisecond time-base.

(3) All commands to be checked in the tandem mode before execution must be assembled in the multiple-event block with Bit 1=0. Additionally, DC commands must have Bit 2=1 also.

(4) Prior to the actual commands or turn duration to be checked, this routine must be entered at (transferred to) TRNSTA to initialize the routine and verify AACS status.

3.2.2.3 Error Recovery. This is the third main function of the Intermediate processing section. The ERROR routine is called whenever an abnormal CCS condition occurs (hardware or software). The others, CMDLOS, IRSPWR, PWRCHK, and RFLOSS are called when abnormal spacecraft conditions occur.

3.2.2.3.1 Error (ERROR). The error routine is provided to respond to anomalous CCS hardware and software conditions. It typically puts the CCS in a known, quiescent state and waits for ground action.

3.2.2.3.1.1 Description. The error routine is entered when one of the following conditions exist:

Hardware

1) A low voltage condition exists.
2) A primary command sync signal has been received before the previous one was processed.
3) An attempt to write into protected memory without a memory-protect override has occurred.
4) An execute of an execute instruction has been attempted.
5) The processor bit generator has reached an illegal state.

Software

1) The primary output unit has been unavailable for 14 seconds or longer.
2) The self-test subroutine has not executed properly.
3) A secondary command sync signal has been received before the previous one was processed.
4) The sequencing support routine (TARMEX) has been asked to activate more time/event tables than it can handle.
5) The output buffer has overflowed.
6) During launch a processor is counting fast relative to the other processor and the FDS.

Most errors cause the routine to be entered at the primary entry point called ERROR. The routine will then sarnple and store the condition of the four flip-flops associated with a primary command error, "internal" error, power fail, or other-processor power

fail. A check is then made of the error entry counter to determine if this error is the first (initial power on). If it is the first error, the primary error buffer is selected to store the following data:

1) Condition of hardware error flip-flops
2) Status of interrupt register 1
3) Status of interrupt register 2
4) Absolute hour count
5) Power Code active indicator
6) Self-test active indicator
7) Output unit exclusive-use indicator
8) Status of mask register 1
9) Status of mask register 2

The routine also checks to determine if the rollback feature is enabled. If it is, the associated time/event table is flagged to be reactivated when the Power Recovery routine provides for it.

The CCS "reset" subroutine is then executed which does the following:

1) Clamps other processor and disables output units.
2) Terminates the following activities if in progress:
   a) Command decoding
   b) Memory dump
   c) Sequence activity (except rollback)
   d) FDS/AACS memory load
   e) Power code processing (momentarily)
   f) DSS tape positioning
3) Clears the following:
   a) Other-processor data
   b) Output unit not-available time-counters (2)

c) Output buffer

d) TARMEX time and block schedules

e) FDS/AACS memory load pointer

f) Power code processing indicators

g) DSS tape direction/active indicator

h) Power low-voltage enable

4) Resets output and telemetry buffer pointers.

5) Maximizes TARMEX time and elapsed-time counters.

6) Resets TARMEX link-stack pointers.

7) Disables interrupts and unmasks the following interrupts:

a) Error

b) DSS tape inputs

c) Power codes

d) Internal interrupt

e) Checksum

f) Command decoding

g) Demand Read

h) 1 PPH

i) RFS failure inputs

j) Power inverter switch and MIRIS supply failure

k) Self-test

After the "reset" function is completed, this routine will initialize the output units such that their data registers will indicate that they are available for commanding (see Paragraph 3.2.3.1).

Then, if the output units have been initialized and the routine is enabled for a response to a tolerance detector trip or a power low-voltage input, it will transfer control to the power recovery routine if one or both of these conditions exist.

If neither of those conditions exist or the routine is not enabled to respond to the power low voltage inputs, then this routine will disable rollback.

If a primary or secondary command error, M/T/E error, output unit time-out error or self test error occurs, the output unit initialization that is done after the RESET function described above will be eliminated.

Figure 3-48 illustrates the flow diagram for this routine.

3.2.2.3.1.2 Self-Test. The basic purpose of the self-test subroutine is to introduce a hardware/software self-test capability into the CCS processors so that no single failure will be able to cause both processors to fail. The self-test subroutine is initiated by one of the special IEX instructions. If the hardware/software test is completed successfully, the Reset Stop function (processor unclamp) automatically occurs. If an error occurs at any time during the test, the test is immediately terminated and no Reset Stop function occurs. An additional feature of the test operation is that the Primary Command Sync interrupt (Interrupt #16) can be reset by the RSI1 instruction only during a successfully operating test. Once the test has terminated, either successfully or unsuccessfully, the RSI1 instruction will not reset the Primary Command Sync interrupt even if bit 18 of the operand is a one. A test latch and test counter have been added to the hardware to control the self-test. The latch is normally in the reset condition and the counter is normally set to its maximum value of 127. The test counter is used to control the hardware checks made during the self-test as described below. The latch is used to enable the hardware checks and to enable the RSI1 Primary Command Sync interrupt reset during the test and the automatic Reset Stop if the self-test terminates successfully. The Primary Command Sync interrupt (#16) can be reset by an RSI1 instruction only if the test latch is set. When the self-test is initiated by the IEX2-Bit 18 instruction, the test latch is set and the test counter is set to zero. The self-test proceeds as described below.

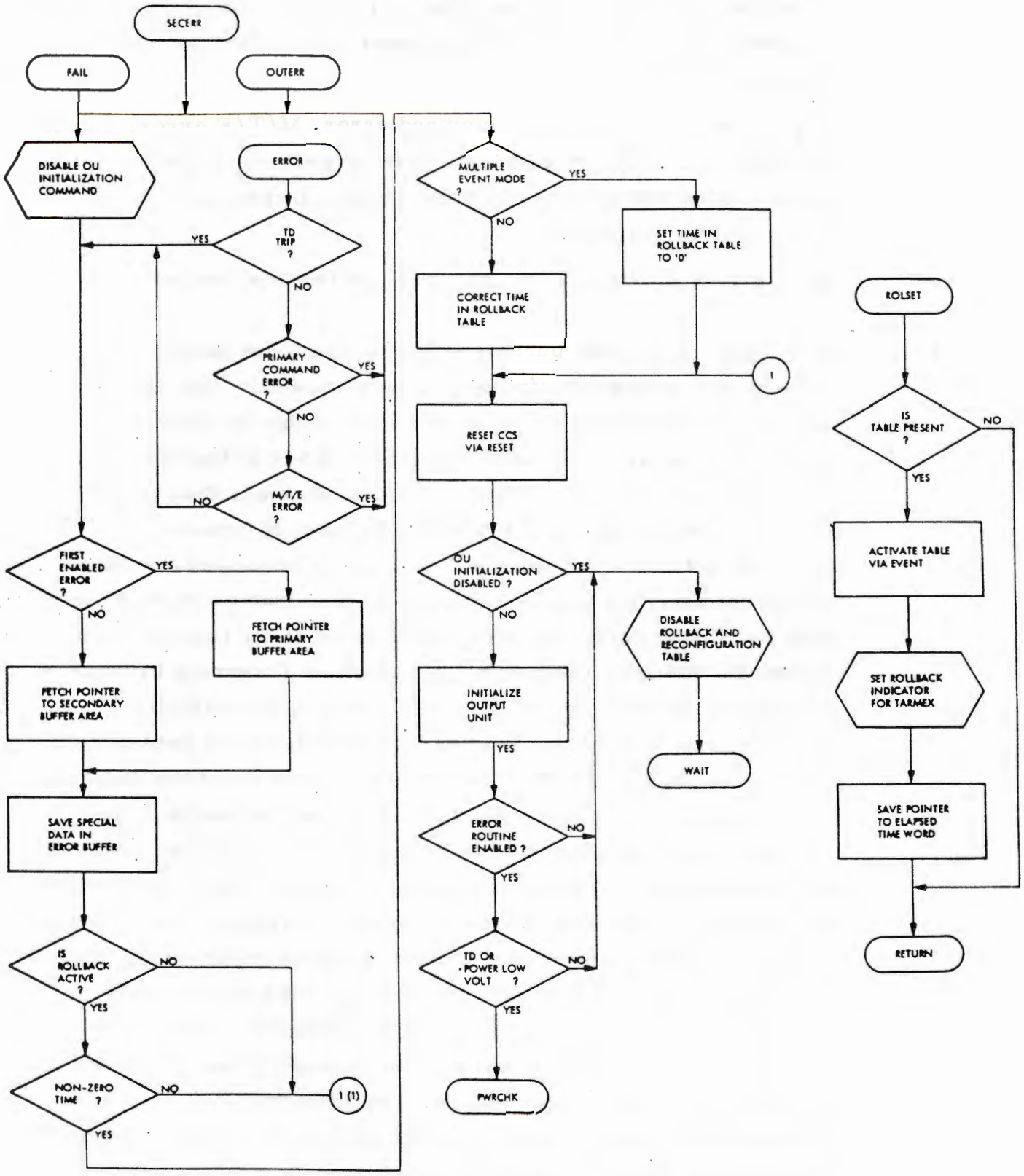


Figure 3-48. Flow Diagram of the Routine Error (Sheet 1 of 2)

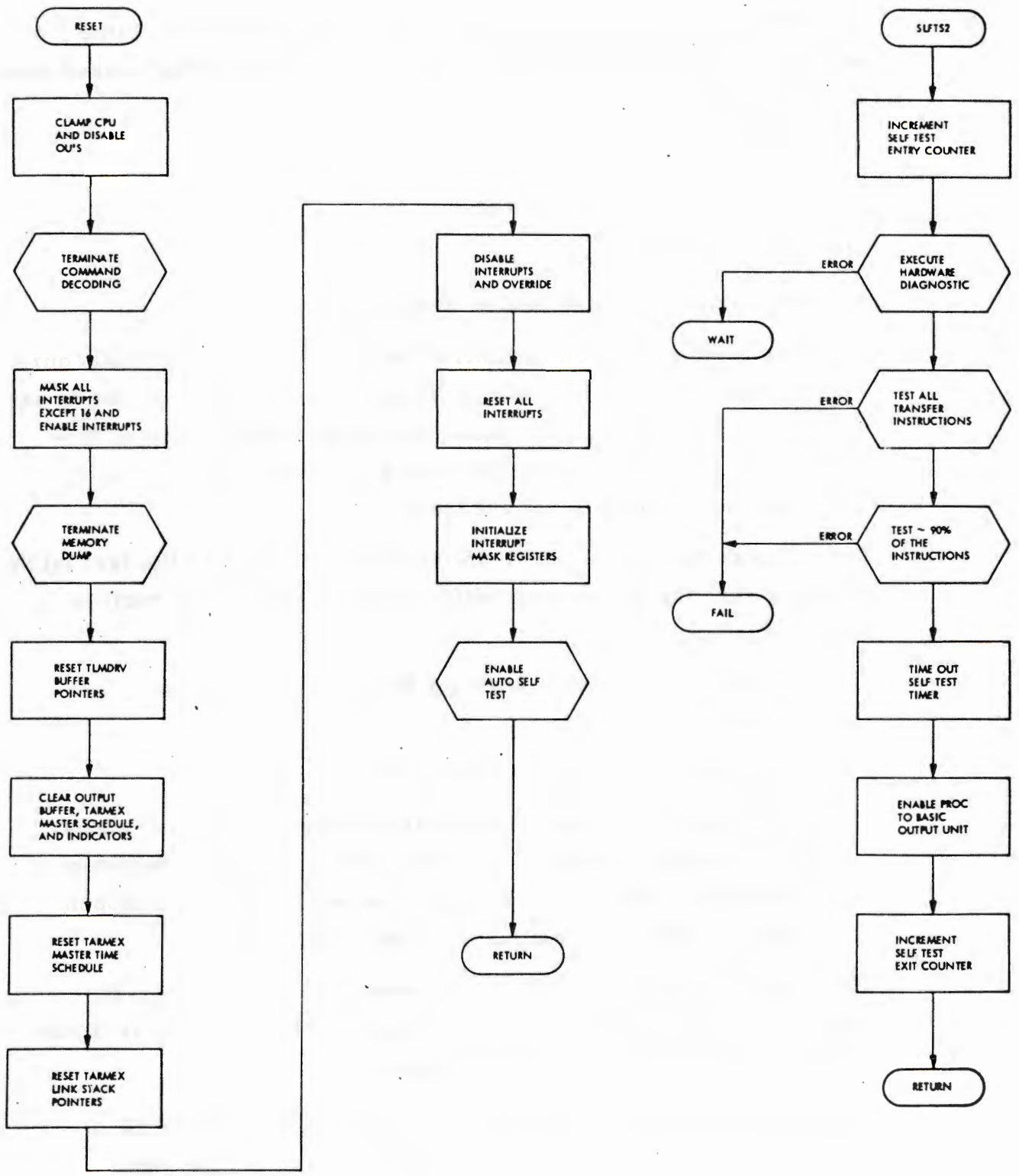


Figure 3-48. Flow Diagram of the Routine Error (Sheet 2 of 2)

122

The hardware/software self-test is initiated during Bit-Time 7 of an IEX2 instruction with Bit 18 set to one. The actions caused are the following:

1) Set the test latch.

2) Set the test counter to zero.

3) Inhibit interrupts.

4) Turn interrupts 2-8 enable override off.

The test counter is used to control the hardware checks made during the self-test while the test latch is set. The test counter is set to zero when the test is initiated and incremented by one at Bit-Pulse-Last of every instruction unless the test counter value is 127, in which case it is not changed.

For all test counter values 1-15 inclusive, as long as the test latch remains set, the following modifications to normal instruction processing occur:

1) The WAIT instruction does <u>not</u> set the inactive state indicator.

2) <u>No</u> operand read occurs during Bit-Time D.

3) <u>All</u> immediate (15-bit-time) instructions have 22-bit-times. This includes bit-times 1-6 which circulate the Z-register 6 extra bit-times. (As a result, unusual things can happen such as TRA 1256 becomes in effect TRA 3412.)

The special hardware checks and constraints made during Bit-Time 1-18 while the test latch is set and the test counter is in the range 1 to 14 inclusive are as follows:

Test Counter Value	Location	Check and Constraint
-	41_8	(Initiate test with IEX 4002)
1	42_8	NOP 0 (000000_8) Instruction
2	43_8	WAIT 7777 (777777_8) Instruction
3	44_8	WAIT 7777 (777777_8) Instruction

Test Counter Value	Location	Check and Constraint
4	45_8	WAIT 7777 (777777_8) Instruction
5	46_8	WAIT 7777 (777777_8) Instruction
6	47_8	WAIT 7777 (777777_8) Instruction
7	50_8	WAIT 7777 (777777_8) Instruction
8	51_8	WAIT 7777 (777777_8) Instruction
9	52_8	WAIT 7777 (777777_8) Instruction
10	53_8	WAIT 7777 (777777_8) Instruction
11	54_8	WAIT 7777 (777777_8) Instruction
12	55_8	WAIT 7777 (777777_8) Instruction
13	--	Special hardware check for a total
14	--	of 44 bit-times; memory address register set to 0000_8; program counter set to 0055_8.
15	55_8	WAIT 7777 (777777_8) Instruction

The locations indicated are not required by the hardware except that the location of the instruction which will be executed after count 14 will always be 55_8 if the test latch is set.

The hardware/software self-test is terminated when the test latch is reset. If the test is terminated before the test counter reaches 127, the counter continues to be incremented by one at each Bit-Pulse-Last until a count of 127 is reached, but no hardware checks are made once the test latch is reset.

The self-test is terminated unsuccessfully if any of the following conditions occur:

1) A constraint violation is detected by the hardware during test counter values 1-14 inclusive.
2) The instruction location in the memory address register at Bit-Time B is greater than 511.

3) An error condition (error interrupt) occurs.

The self-test is terminated successfully if the test latch is still set when the test counter reaches 127. In this case, the following actions occur during the instruction following that during which the test counter was incremented from 126 to 127:

1) A Reset Stop function occurs at Bit-Time A. (This removes the processor clamp.)

2) The test latch is reset at Bit-Time B.

3.2.2.3.2 Command Loss (CMDLOS). The purpose of the command loss routine is to provide a means for the VGR spacecraft to automatically correct a failure to receive ground commands. A spacecraft failure will be assumed by this routine whenever a preset number of hours have elapsed since the last received valid command.

Since the algorithm must execute during a period when the spacecraft cannot receive and/or decode a command, it is designed to provide complete reconfiguration of the spacecraft's antenna - receiver-command detector chain.

Furthermore, if loss of command ability is due to a false lock condition (a receiver locked up on a downlink spur), the algorithm must also reconfigure the downlink elements (S and X band exciters and XMTR's, and USO). To cover all possibilities and be effective for multiple failures, all "uplink" combinations are selected for each "downlink" combination. Passes through the algorithm are made alternately using primary then secondary power relays where they exist.

3.2.2.3.2.1 Description. Whenever a specified number of hours have elapsed since the last valid command was received by the CCS, this routine will assume a spacecraft failure and attempt to correct for that failure by systematically switching redundant elements until a valid command is received by the CCS. The execution of the CMDLOS routine in Spacecraft 32 has been changed from that in Spacecraft 31 due to the failure of receiver 1 on Spacecraft 32.

At the start of the routine the initial configuration commands listed in Table 3-4 are issued. Additionally, on Spacecraft 32 the most significant bit of the AUTCTR location is incremented by one to provide a CCS telemetry indication that the CMDLOS routine has been activated.

After the initial configuration commands, the routine will wait six hours on Spacecraft 31 or 12 hours on Spacecraft 32 and then check to determine if any commands have been received.

If not, the routine issues 2ER to select the low gain antenna on Spacecraft 31, or 3HRP to select CDU B on Spacecraft 32. After an additional six hours (S/C 31) or 12 hours (S/C 32) without command reception, the following commands are issued:

2HR	X-BAND TWT LOW POWER
2DR	S-BAND XMTR LOW POWER
1TR	BAY 1 HEATER ON
2KP	S-BAND XMTR POWER ON
2GP	X-BAND TWTA POWER ON

Then the routine issues the first command from the downlink table (Table 3-5A), waits six minutes on Spacecraft 31 or 15 minutes on Spacecraft 32 then sends each of the commands listed in the "uplink" table (Table 3-5B) on six minute centers on Spacecraft 31 or 15 minute centers on Spacecraft 32. When the routine completes Table 3-5, it issues the next command from Table 3-5A and cycles again through Table 3-5B. This sequence is repeated until all of the commands in Table 3-5A are issued. The routine then selects the secondary power matrix command option and the other PWR command decoder and repeats the entire sequence of commands (Tables 3-5A and 3-5B). Additionally, on Spacecraft 32, the routine is structured slightly different in each CCS processor such that all four combinations of PWR relays and decoders are utilized.

On spacecraft 31 the cycling through the downlink and uplink tables and power matrix options continues until a valid command is received by the CCS, whereupon the CMDLOS routine will terminate.

Table 3-4. Initial CMDLOS Configuration Commands

Spacecraft 31		Spacecraft 32	
2P	TWO-WAY NON-COHERENT ON	2P	TWO-WAY NON-COHERENT ON
2NR	X-BAND RANGING OFF	2AR	S-BAND RANGING OFF
2AR	S-BAND RANGING OFF	7SS1	SUN SEARCH
2FRP	RECEIVER 2 SELECT		
3HRP	CDU B SELECT		
7SS1	SUN SEARCH		

Table 3-5A. Downlink Configuration

Spacecraft 31	Spacecraft 32
1. 2BP S-BAND EXCITER 1 SELECT AND 3A28311 TMU END CIRCUIT COMMAND	1. 2BP S-BAND EXCITER 1 SELECT
2. 2CRP S-BAND XMTR 2 SELECT	2. 2CRP S-BAND XMTR 2 SELECT
3. 2BRP S-BAND EXCITER 2 SELECT	3. 2BRP S-BAND EXCITER 2 SELECT
4. 2CP S-BAND XMTR 1 SELECT	4. 2CP S-BAND XMTR 1 SELECT
5. 2GRP X-BAND POWER OFF AND 1T BAY 1 HEATER ON	5. 2GRP X-BAND POWER OFF AND 1T BAY 1 HEATER ON
6. 2KRP S-BAND XMTR OFF AND 2MRP S-BAND EXCITER OFF	6. 2KRP S-BAND XMTR OFF AND 2MRP S-BAND EXCITER OFF AND 2E LGA SELECT
7. 2MP S-BAND EXCITER ON AND 2KP S-BAND XMTR ON	7. 2MP S-BAND EXCITER ON AND 2KP S-BAND XMTR ON
8. 1TR BAY 1 HEATER OFF AND 2GP X-BAND POWER ON AND 2QR USO OFF	8. 1TR BAY 1 HEATER OFF AND 2GP X-BAND POWER ON AND 2QR USO OFF
9. 2Q USO ON	9. 2E HGA SELECT

Table 3-5B. Uplink Configuration

Spacecraft 31	Spacecraft 32
1. 3HRP CDU B SELECT	1. 2NR X-BAND RANGING OFF
2. 2FRP RCVR 2 SELECT	2. 3HRP CDU B SELECT
3. 3HP CDU A SELECT	3. 3A32311 TMU STATE COMMAND
4. 2E HGA SELECT	4. 3A43330 TMU STATE COMMAND
5. XX DUMMY DC COMMAND	5. 3HP CDU A SELECT
6. 3HRP CDU B SELECT	6. XX DUMMY DC COMMAND
7. 2FP RCVR 1 SELECT	7. 2Q USO ON
8. 3HP CDU A SELECT	
9. 2ER LGA SELECT	
10. XX DUMMY DC COMMAND	
11. XX DUMMY DC COMMAND	

On Spacecraft 32, after the second time through the downlink and uplink tables, the routine will terminate and re-enable the timer to re-activate the routine. The re-activation timer will be set to double the number of hours of the previous cycle. Figures 3-49A and 3-49B are the flow diagrams which illustrate the operation of this routine on each spacecraft.

3.2.2.3.2.2 Constraints. When the command loss routine is active it is normally inhibited from being started again until it is finished the first time. Other software (sequence, error, etc.,) must not be allowed to remove the restart inhibit before an active command loss routine is finished.

3.2.2.3.3 MIRIS Auto Switch (IRSPWR). This routine responds to a change in the MIRIS level input (level 6, 2). If a change occurs, the routine will disable re-entry, wait one minute, and then increment the master automatic counter which is incremented any time one of the automatic routines is entered. It then checks to determine the state of the level input. If the level is set (1), the routine will issue the following two commands:

39FR MIRIS STANDBY SUPPLY A OFF
39B MIRIS STANDBY SUPPLY B ON

The routine will then exit without re-enabling itself.
If the level is not set, the routine will re-enable itself and return.
Figure 3-50 illustrates the flow diagram for this routine.

3.2.2.3.4 Power Recovery (PWRCHK). The purpose of the power recovery routine is to provide a means for the VGR spacecraft to automatically configure itself to a safe, low-power mode following a power subsystem under-voltage condition, a main-to-standby inverter switch, or a CCS Tolerance detector trip.

3.2.2.3.4.1 Description. As far as the CCS is concerned, there are three indicators of an abnormal power condition:

(1) The undervoltage signal input from PWR to CCS.
(2) The main-to-standby inverter switch signal input from PWR to CCS.

29

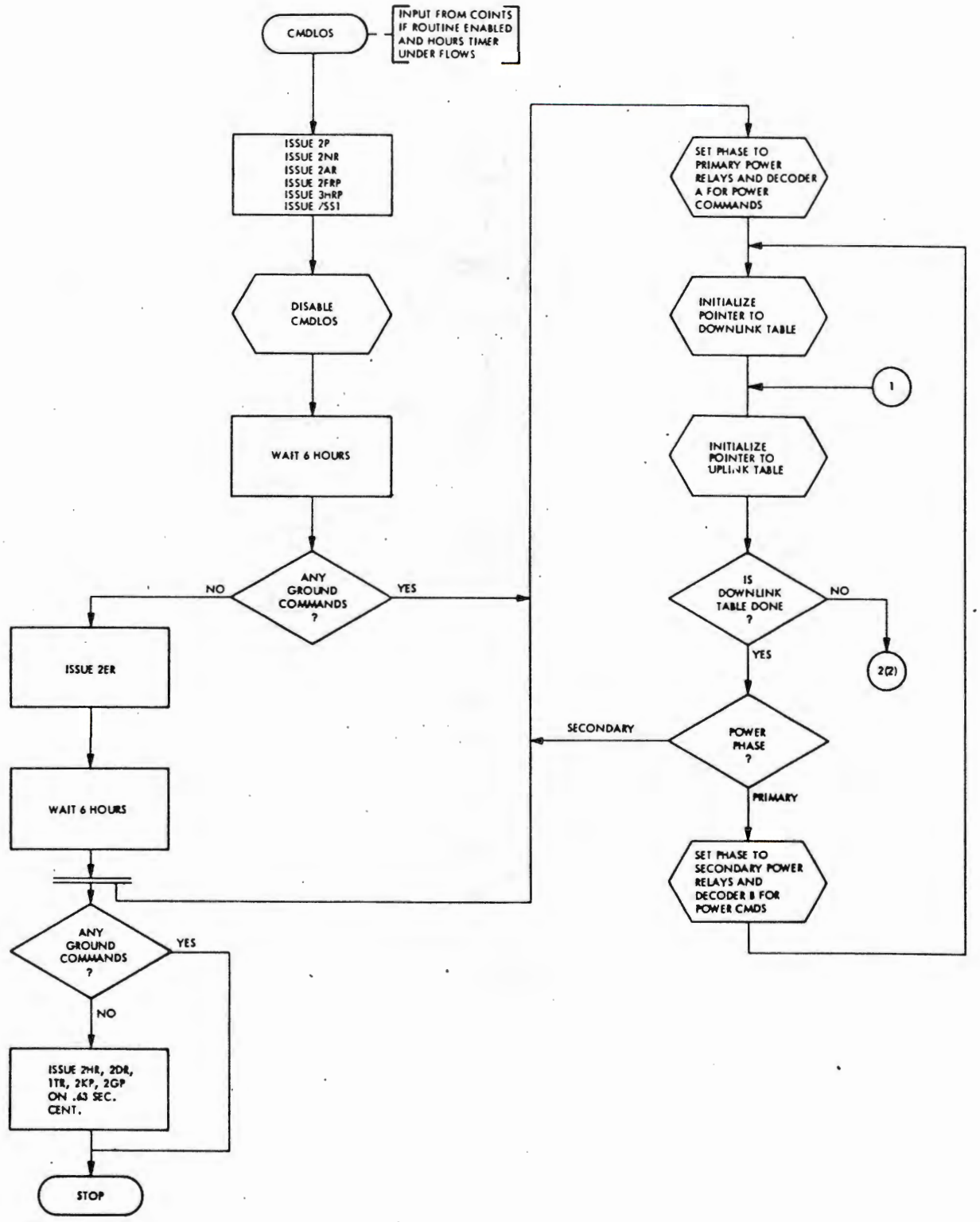


Figure 3-49A. Flow Diagram of the Routine CMDLOS for Spacecraft 31
(Sheet 1 of 2)

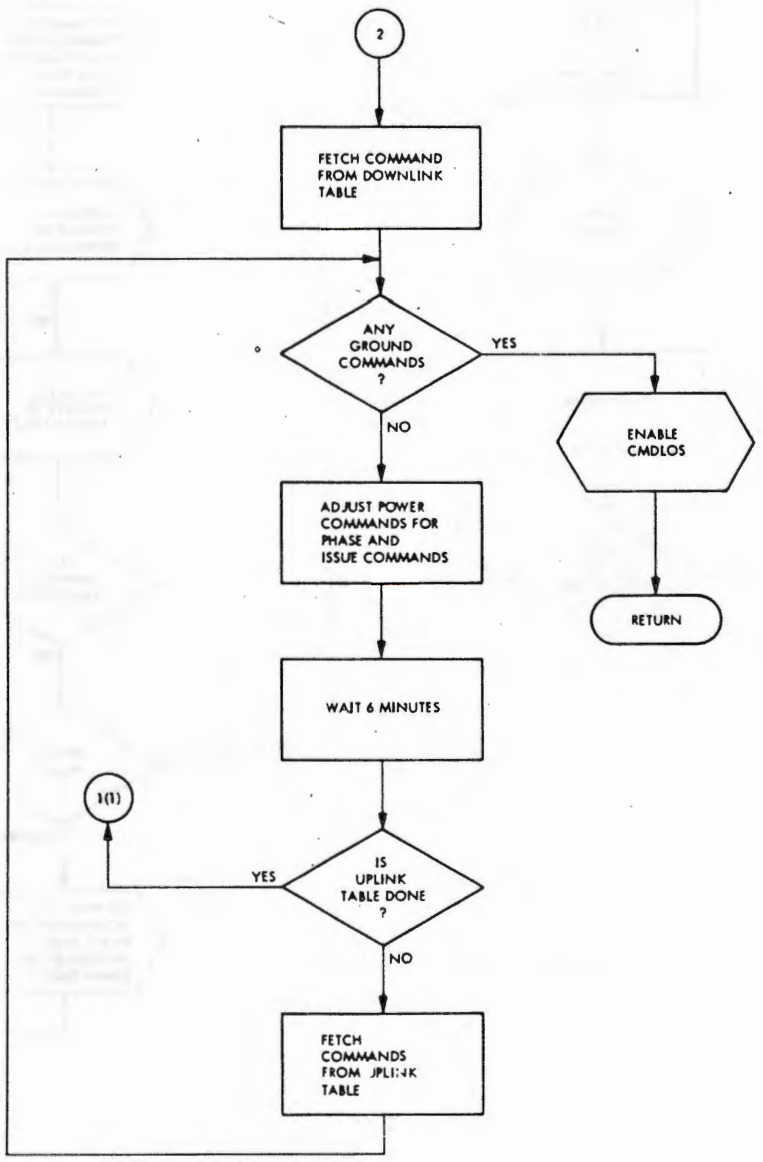


Figure 3-49A. Flow Diagram of the Routine CMDLOS for Spacecraft 31 (Sheet 2 of 2)

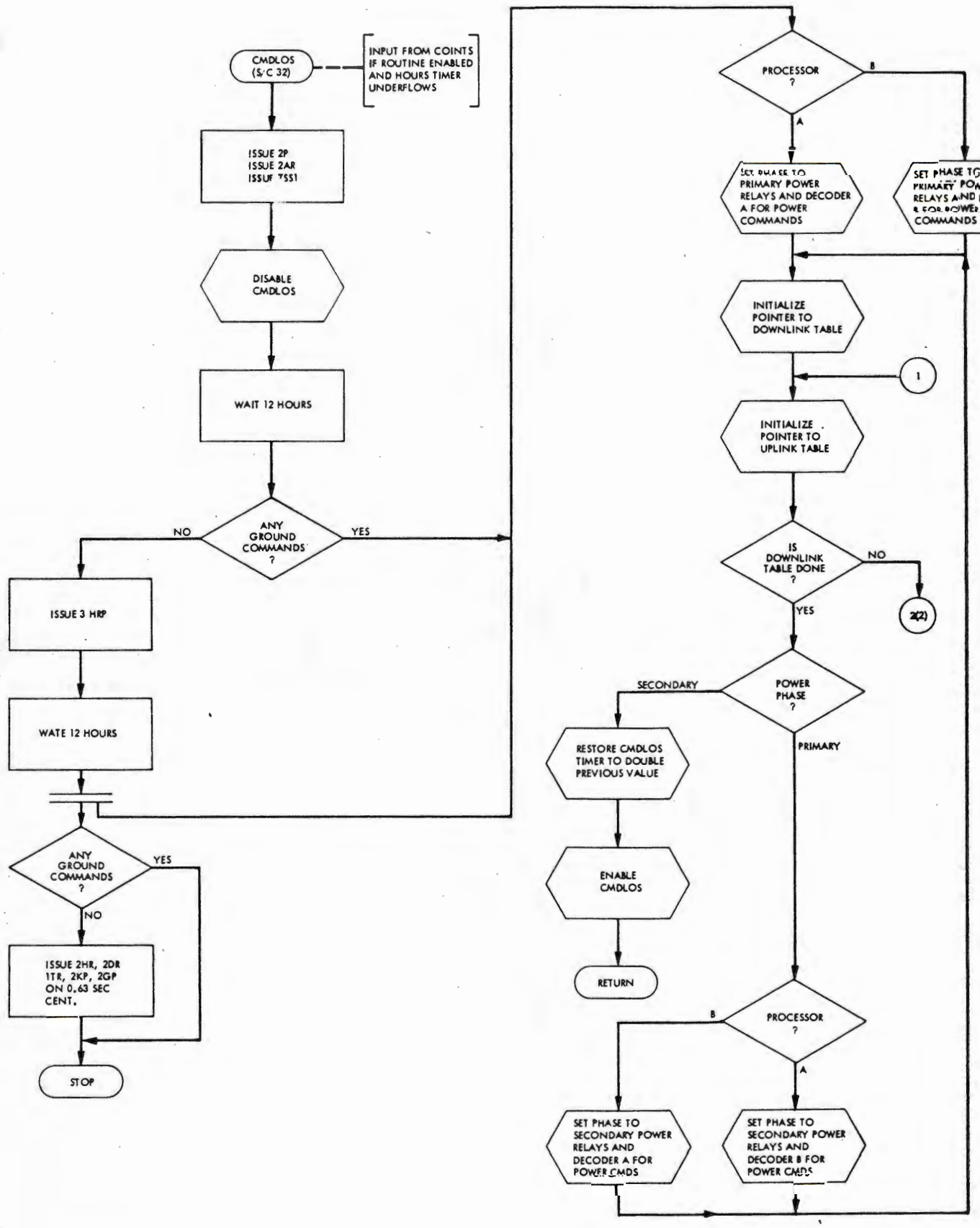


Figure 3-49B. Flow Diagram of the Routine CMDLOS for Spacecraft 32 (Sheet 1 of 2)

13D

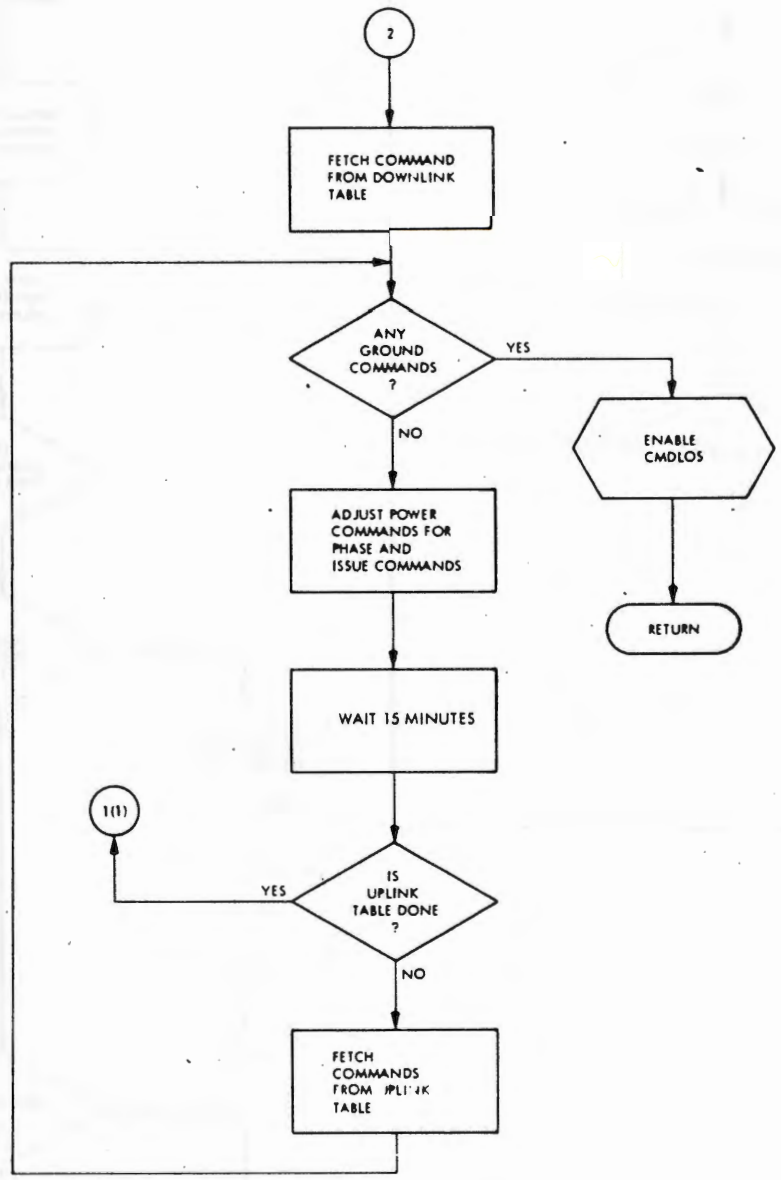


Figure 3-49B. Flow Diagram of the Routine CMDLOS for Spacecraft 32 (Sheet 2 of 2)

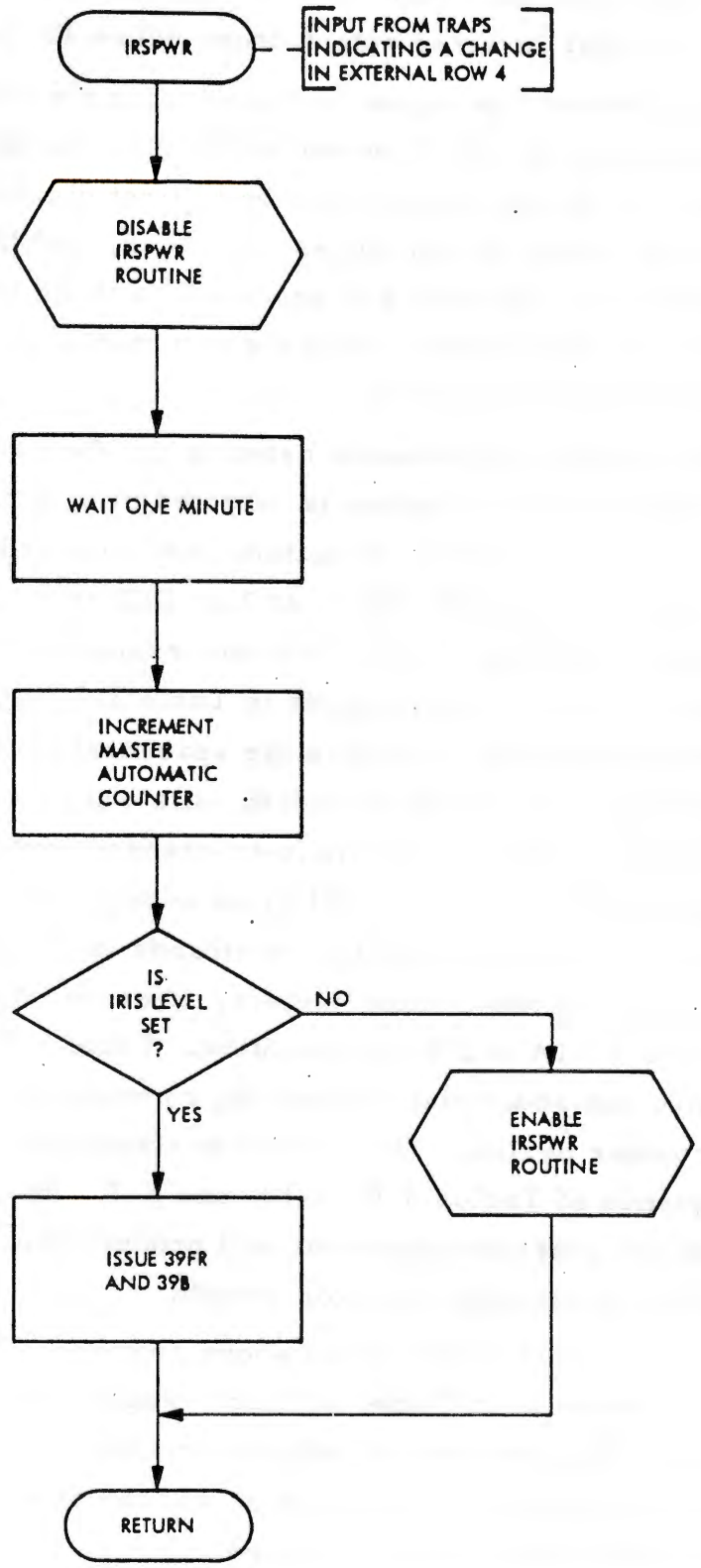


Figure 3-50. Flow Diagram of the Routine IRSPWR

3) The CCS tolerance detector circuit which senses when the 2.4 Khz inverter output drops below 45.3 volts (approx.).

If an undervoltage signal is received by the CCS, the power subsystem has turned off power to the AC2 and DC2 busses. The response by this routine will be to first disable entry into this routine, AACS Power Codes, RF Loss, and MIRIS Power. The routine will then wait 4-5 seconds and if the power low voltage condition still exists, issue the commands in Table 3-6 on 200 millisecond centers.

The next set of commands depends on whether or not the power recovery override option is selected or a main-to-standby inverter switch has occurred. If neither has occurred, an overload condition associated with either AC2 or DC2 is assumed and the commands in Tables 3-9 and 3-10 are executed on 200 millisecond centers, then the commands in Table 3-7 on one-second centers. These commands configure the spacecraft to a minimum power level and also select redundant units where possible to help eliminate the overload. If a main-to-standby inverter switch has also occurred (always preceded by an undervoltage trip), an inverter failure is assumed and the commands in Tables 3-7 and 3-8 are executed on one-second centers, after which the commands in Tables 3-11A and B are executed. Tabel 11B has been added to provide additional configuring commands for the planetary encounter period. This series of commands replaces the sequence of Tables 3-9, 3-10, and 3-7. An undervoltage trip with the override option set will produce the same response as a main-to-standby inverter switch.

Finally, after either of the above sequences has been completed, the commands in Table 3-12 are executed on 100 millisecond centers. This sequence of subsystem initializing commands includes reenabling entry to this routine and the other automatics disabled at initial entry into this routine.

Two special options are available early in the mission. A "power up" option provides a means to automatically configure the spacecraft to a data transmitting configuration shortly after spacecraft turn-on. This affords a work around method to acquire engineering telemetry in the event that the spacecraft is not responding to

Table 3-6. Command List 'A1'

(200 Millisecond Centers)

2AR	S-BAND RANGING OFF
2GRP	X-BAND XMTR OFF
2KRP	S-BAND XMTR OFF
2MRP	S-BAND EXC. OFF
2NR	X-BAND RANGING OFF
2 PR	TWO-WAY NON-COHERENT OFF
3BRP	TMU OFF
22BRP	PRA/PWS ANTENNA DEPLOY OFF

Table 3-7. Command List 'B1'

(1 Second Centers)

36JP	ISS NA ELEC REPL HTR ON
36H	ISS WA ELEC REPL HTR ON
36DRP	ISS NA OFF
36BR	ISS WA OFF
IT	BAY 1 HEATER ON
4A	UNDERVOLTAGE TRIP RESET
3 GRP	MDS TMU SELECT (B)
3 BP	TMU ON
2BRP	S-BAND EXCITER 2 SELECT
2CP, 2CRP, 2LP, or 2LRP	XMTR SELECT (PROGRAMMABLE)
2DR	S-BAND XMTR LOW POWER*
2JRP	X-BAND EXCITER 2 SELECT
2MP	S-BAND EXCITER POWER ON
2KP	S-BAND XMTR POWER ON

*S/C 32 has the capability to disable this command from the BML3 sequence.

136

Table 3-8. Command List 'C' for PWRCHK Routine

(a) S/C 31

INITIATE COMMAND LIST 'B1'
WAIT 14 SECONDS
6 EVENTS ON 1 SECOND CENTERS
NOP
NOP
NOP
NOP
NOP
NOP

(b) S/C 32

INITIATE COMMAND LIST 'B1'
WAIT 14 SECONDS
4 EVENTS ON 1 SECOND CENTERS
4K RTG 1 DIODE BYPASS ON
4L RTG 2 DIODE BYPASS ON
4M RTG 3 DIODE BYPASS ON
INITIATE COMMAND LIST 'D2'

ground commands. The events issued are shown in the flow diagram of Figure 3-51. This option and its events are cleared out by executing Launch Hold/Reset. A second option, the "launch" option permits a rapid restart of the launch sequence via the roll-back capability. The events issued (after those in Table 3-6) are contained in Tables 3-11 and 3-12. The "launch" option is enabled by the Launch Hold/Reset routine and is eventually disabled by the launch sequence.

A CCS tolerance detector (TD) trip indicates that the 2.4 Khz power voltage has dropped below 45.3 volts (approx.). It is possible that the CCS TD will activate without a corresponding under-voltage trip or inverter switch. This can occur because the CSS TD is more sensitive to transients on the 2.4 Khz power buss than

is the undervoltage sensor in the PWR subsystem. Because a transient as sensed by the CCS TD could potentially disrupt other subsystems on the spacecraft, the commands in Tables 3-11 and 3-12 will be executed for this condition.

It is possible that an RFS TWT could short out or fail in a high power mode and cause the CCS TD to trip without an associated undervoltage trip and inverter switch. If this occurs, and the associated TWT high-voltage interrupt activates, the CCS must switch to the redundant TWT before the RFS restores the failed TWT's high voltage or else re-entry to this routine will continually occur each time the high-voltage is restored and the failure (TD trip) recurs. This is prevented by eliminating the five minute delay associated with the TWT failure check in the RF loss routine. The commands associated with Tables 3-11 and 3-12 are then issued which include entry to the RF Loss routine.

Table 3-9. Command List 'A2'

(200 Millisecond Centers)

16 ARP	DSS OFF
22AR	PRA OFF
23AR	PWS OFF
25AR	LECP OFF
25BR	LECP STEPPER MOTOR OFF
27AR	PPS OFF
32AR	PLS OFF
34AR	UVS OFF
35AR	MAG OFF
35BR	MAG STANDBY OFF
35CR	MAG IBLFM FWD FLIPPER OFF
35DR	MAG IBLFM REV FLIPPER OFF
35ER	MAG OBLFM FWD FLIPPER OFF
35FR	MAG OBLFM REV FLIPPER OFF
36DRP	ISS NA OFF
36BR	ISS WA OFF
39AR	MIRIS OFF
39CR	MIRIS FLASHOFF HTR OFF

Each time a power undervoltage signal is detected by the CCS, this routine will increment a count word which will keep a running count of undervoltage trips. In addition, a TD trip counter is incremented each time a CCS TD trip, only, occurs. Also, as with all "automatics," the master automatic counter is incremented. This counter is telemetered from the CCS each hour.

120

Table 3-10. Command List 'B2'

(200 Millisecond Centers)

39D	MIRIS REPL HTR ON
36H	ISSWA ELEC REPL HTR ON
36JP	ISSNA ELEC REPL HTR ON
36G	ISSWA VID REPL HTR ON
36FP	ISSNA VID REPL HTR ON
34B	UVS REPL HTR ON
32C	PLS REPL HTR ON
32B	PLS SUPP HTR ON
27B	PPS REPL HTR ON
25C	LECP REPL HTR ON
25D	LECP TELE HTR ON
25E	LECP SUPP HTR ON
7SHP	AACS AZ ACT HTR ON
1D	BAY 2 HEATER ON

Each time a CCS TD trip occurs, normal CCS operation (sequences) will typically terminate and the CCS error routine will be activated. Details of the error routine are discussed in Paragraph 3.2.2.7.

Figure 3-51 illustrates the flow diagram for this routine.

3.2.2.3.5 RF Power Loss (RFLOSS). The purpose of the RF Power Loss routine is to provide a means for the Voyager spacecraft to automatically recover from an S- or X-band exciter or XMTR failures.

3.2.2.3.5.1 Description. The CCS monitors four input signals from the RFS associated with a failure of the S and X-band exciters and XMTR's. Whenever one or more of the above signals is input to the CCS, the RF Power Loss routine will be activated.

Table 3-11A. Command List 'D2' for PWRCHK Routine
(Immediate)

CHECK AND ENABLE RF POWER LOSS AUTOMATIC			
INITIATE SPECIAL ROLLBACK TABLE			
16C40XX or NOP*			
NOP or INITIATE COMMAND LIST 'D3' *			
INITIATE COMMAND LIST 'ISS' (Table 3-13)			
5 EVENTS ON 1 ISS FRAME CENTERS			
RESET CCS HARDWARE CLOCK			
SC25AH 1777	S/C 31 or	SC25AH 1677	S/C 32
SC25AH 2617		SC25AH 3017	
SC25AH 5200		SC25AH 5200	
SC25AH 6072		SC25AH 6032	

*Dependent on mission phase

Table 3-11B. Command List 'D3; for PWRCHK Routine

7 MINUTE DELAY	
2 HR	X-BAND TWT LOW POWER
2 MINUTE DELAY	
1 TR	BAY 1 HEATER OFF
2 MINUTE DELAY	
2GP	X-BAND XMTR ON
2 MINUTE DELAY	
SC06BB050090	GS-3
NO TIME DELAY	
START PLATFORM NEUTRAL SEQUENCE	
9 MINUTE DELAY	
PLATPO HEADER	
XXX	RESERVED FOR OPTIMUM
XXX	PLATFORM POSITIONS DURING ENCOUNTER

Change #5
25 Feb 82

1F1

Table 3-12. Command List 'D1' for PWRCHK Routine
(100 Millisecond Centers)

4A UNDERVOLTAGE TRIP RESET
6E0B, 6E0B END OF FDS MEMORY LOAD
SC21AG 0
NOP (USED TO LINK PWRCHK TO TRNSUP ABORT TABLE)
ISSUE POWER FAIL MESSAGE TO OTHER PROCESSOR
CHECK AND ENABLE IRSPWR AUTOMATIC
ENABLE CMDLOS AUTOMATIC
ENABLE $\overline{\text{IDET}}$ ALGORITHM
ENABLE CA AND $\overline{\text{CA}}$ ALGORITHM
ENABLE $\overline{\text{IDET}}$ OR $\overline{\text{ADET}}$ ALGORITHM
RESTART LOSS OF REFERENCE ROUTINE IF ACTIVE
* NOP OR START 'DRIFT-STOP'
* NOP OR START FCP SWAP TABLE OR RESTART BML-4 SEQUENCE (S/C 32 ONLY)
INITIATE HEARTBEAT CHECK ALGORITHM
WAIT 10 MINUTES
ENABLE POWER LOW VOLTAGE
*Dependent on mission phase

Table 3-13. Command List 'ISS' for PWRCHK Routine
(1 Second Centers)

6 EVENTS ON 1 SECOND CENTERS	
CC36BRP	ISS NA OFF
CC36BR	ISS WA OFF
CC36H	ISS WA ELEC REPL. HTR ON
CC36JP	ISS NA ELEC REPL. HTR ON
CC36G	ISS WA VID REPL. HTR ON
CC36FP	ISS NA VID REPL. HTR ON

Table 3-12, Change #5
25 Feb 82

Table 3-14. Command List 'BML3' for PWRCHK Routine - S/C 32, Processor A only

DELETED

Table 3-15. Command List 'Drift-Stop' for PWRCHK Routine - S/C 31 only (Immediate)

AC7MDP6	ALL AXIS INERTIAL	
AC7VCD3, 000000	STOP	DRIFT TURN
AC7VCD1, 000000	STOP	DRIFT TURN
AC7VCD2, 000000	STOP	DRIFT TURN

S/C 32 only (Immediate)

AC7MDP6	ALL AXIS INERTIAL
AACS MEM LOAD (BASE ADDR = 7014, 3 WORDS)	ZERO PITCH, YAW, AND ROLL DRIFT-TURNS
AC7PAR 3743, 701462	

Change #4
16 July 1981

103

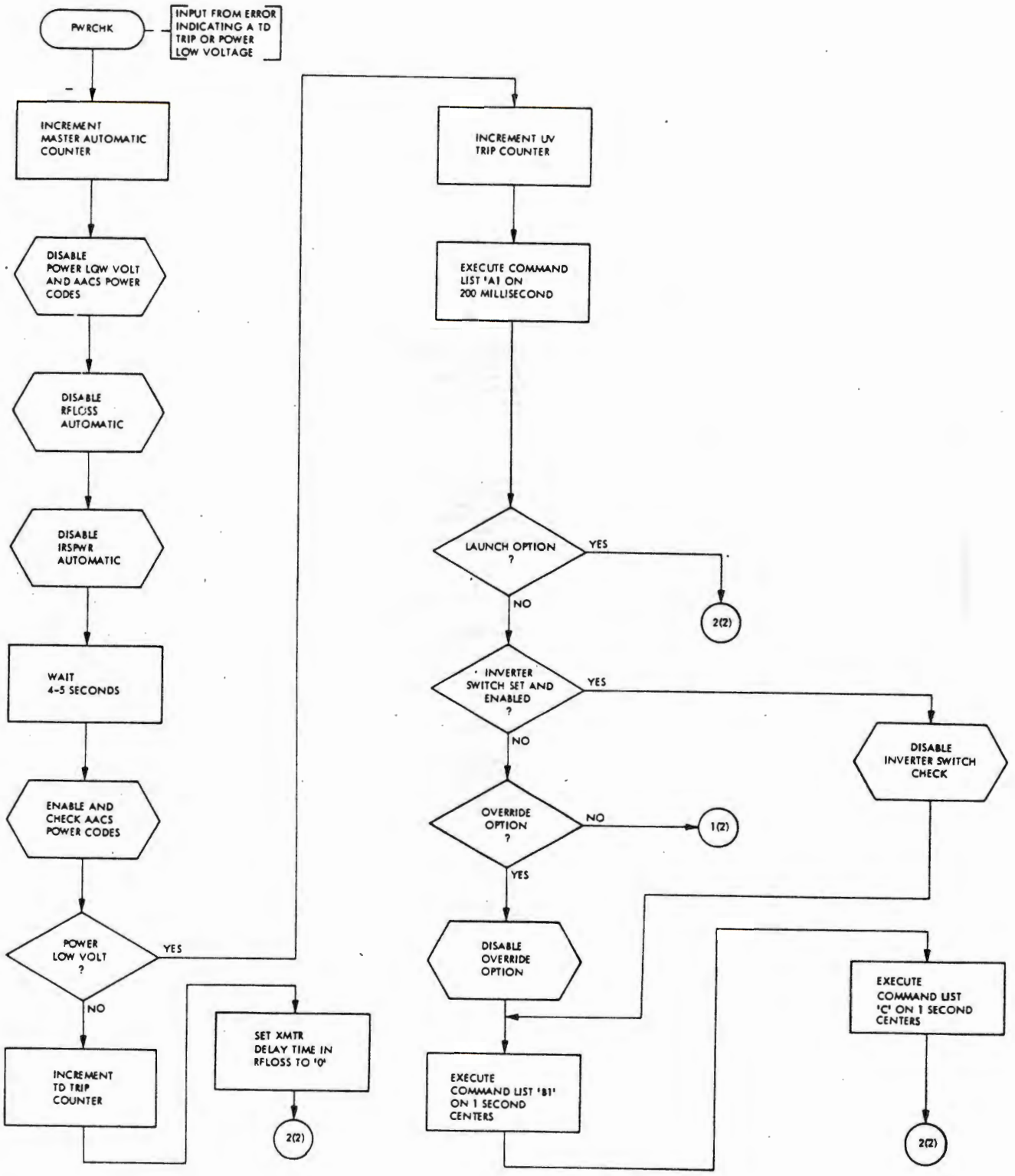


Figure 3-51. Flow Diagram of the Routine PWRCHK (Sheet 1 of 2)

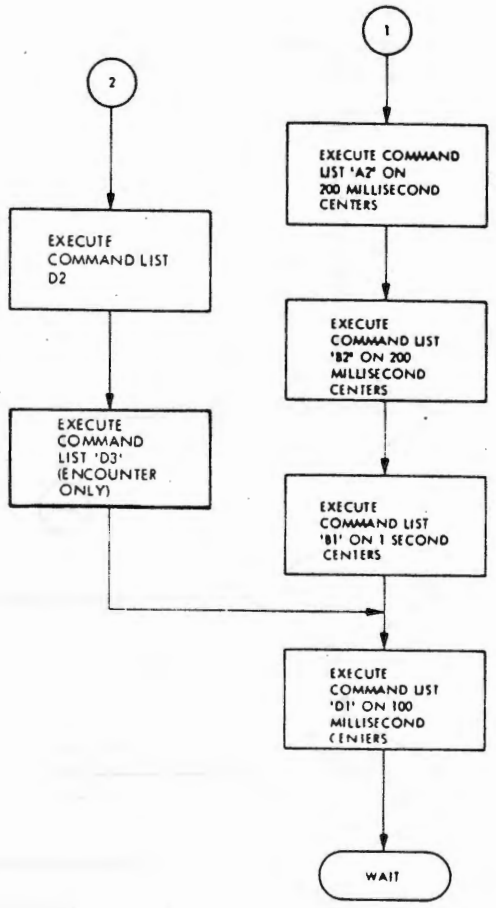


Figure 3-51. Flow Diagram of the Routine PWRCHK (Sheet 2 of 2)

When activated the routine will first disable re-entry, increment the master automatic counter, and then wait five seconds to allow the exciters to stabilize (in case they were just powered up or switched). After the five second delay, the RF Loss counter is incremented and the S-band exciter input is checked. If the S-band exciter failure signal is present, the routine will issue DC2P to place the RFS in the two-way non-coherent mode to cause a switch to the auxiliary oscillator (or USO if on). After one second the routine will increment the RF Loss automatic counter again and recheck the S-band exciter input.

If the S-band exciter failure is still present, the routine will disable any re-entry into the S-band exciter failure subroutine, and issue a CC2BRP to switch the S-band exciter. The routine will then start the five second delay to increment the RF Loss automatic timer and check for other RFS failures. Also, one second after CC2BRP, the S-band exciter level will be checked again, and if the failure is still present, DC2QR will be issued to turn off the USO.

If issuing the DC2P command corrected the failure, the routine will immediately check for other RFS failures. If it did not, then the other RFS inputs will be checked by re-entering the routine after the five second delay. After the S-band exciter failure analysis and response is completed, the X-band exciter is checked.

If the failure signal is present, the routine will disable future X-band exciter checks, and issue CC2JRP to switch exciters. If the X-band exciter failure signal is still present one second after issuing the above commands, this routine will issue CC2BRP to switch the S-band exciter. The routine will also disable future S-band exciter checks. Note: This exciter and MDS output may have already been selected as a result of previous processing of an S-band exciter failure signal.

After responding to the X-band exciter failure signal in the manner described above, the routine next checks the S and X-band XMTR

failure signals. However, a five minute delay is provide before checking the XMTR inputs to allow for XMTR stabilization in the event that this routine was entered as a result of power turn-on or a XMTR switch.

If the S-band XMTR failure signal is present after the five minute delay, the routine will issue commands to PWR to switch XMTR's* and set the XMTR to low power if it is the first S-band XMTR failure, and then proceed with the X-band TWTA check. If not the first, then the XMTR has already been switched, and the routine will instead switch S-band exciters if they have not been switched, before. After the S-band exciter switch, future S-band exciter checks are inhibited and the routine continues by re-entering at the five second delay point (beginning). If the S-band exiter has already been switched, the routine will inhibit future S-band XMTR checks and issue CC2KRP to turn off the S-band power. The routine will then continue and check the X-band TWTA input.

If the X-band TWTA failure signal is present, the routine will respond in one of two ways. During Saturn FE1, 2, NE, and PE, the routine will issue CC2LP to switch to the redundant X-band XMTR. If the failure signal is still present after switching X-band XMTRs, then the X-band exciters will be switched. If switching X-band exciters still does not correct the failure, then the X-band power will be turned off, and the bay 1 heater turned on (CC2GRP and CC1T).

During cruise and Saturn OB, the switch to the redundant X-band XMTR and exciter is inhibited. If an X-band transmitter failure occurs, the X-band power will be turned off, the bay 1 heater turned on, and the S-band power will be turned ON.

---

* Note: For S/C 32, the presence of BML-4 requires special treatment of which S-band XMTR should be selected. When the S-band is powered on with the SSA selected (CC2CRP), RFLOSS is configured to select the TWTA (CC2CP). When the S-band is powered off with the SSA selected, RFLOSS is configured to select the SSA (CC2CRP). Reference ECR 37424.

Change #6
7/26/82

157

After the RF loss routine is done processing all the failure inputs, it will restore the five minute delay prior to the XMTR checks (in case it was eliminated by the Power Recovery routine), restore re-entry to the routine, and return to previous processing.

After exiting the RF Loss routine, the CCS will only re-enter the routine if there is a change in the condition of any of the four RFS signals, or the Power Recovery routine is executed.

Figure 3-52 illustrates the flow diagram of this routine.

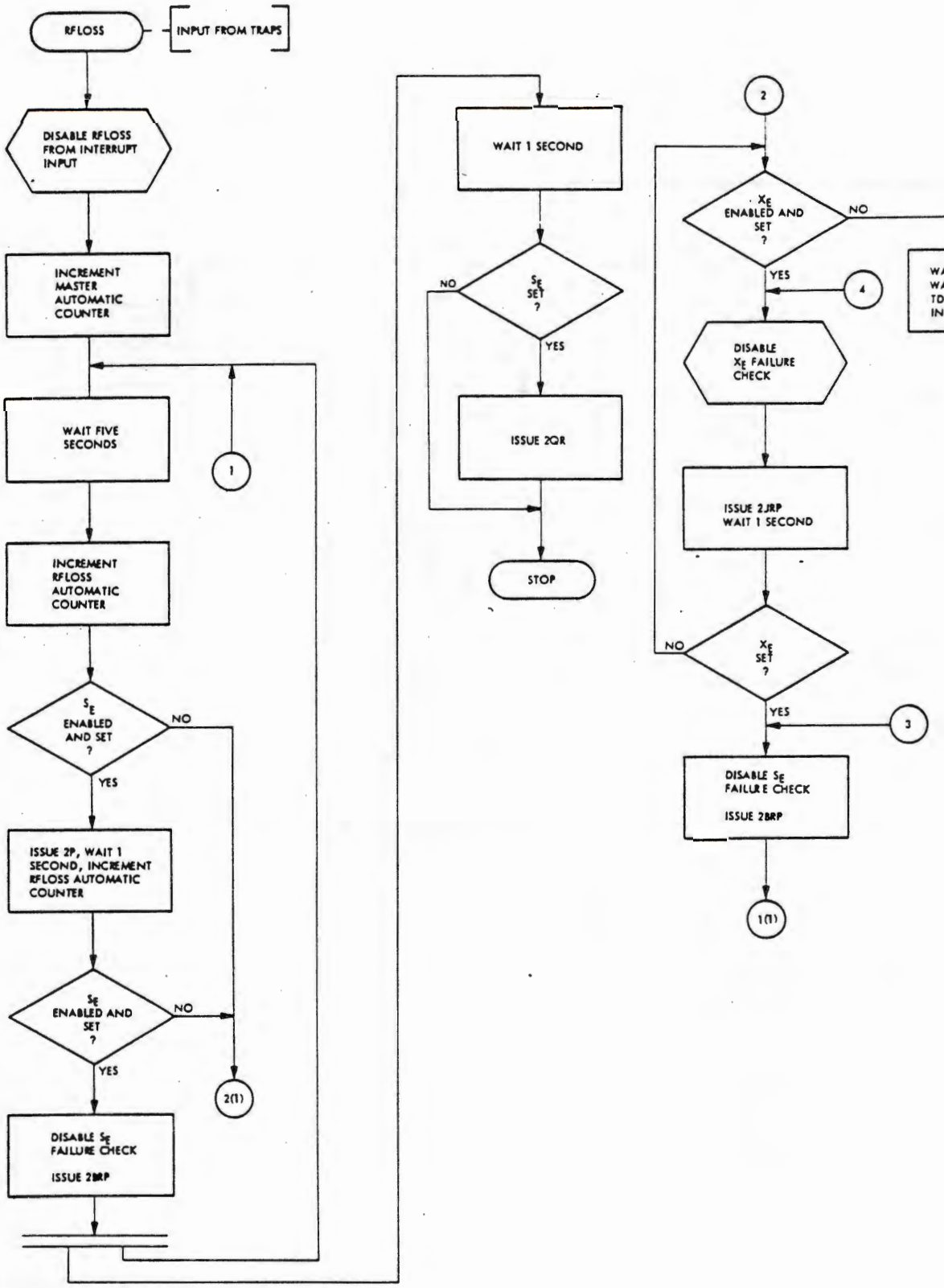


Figure 3-52. Flow Diagram of RF LOSS Routine (Sheet 1 of 2)

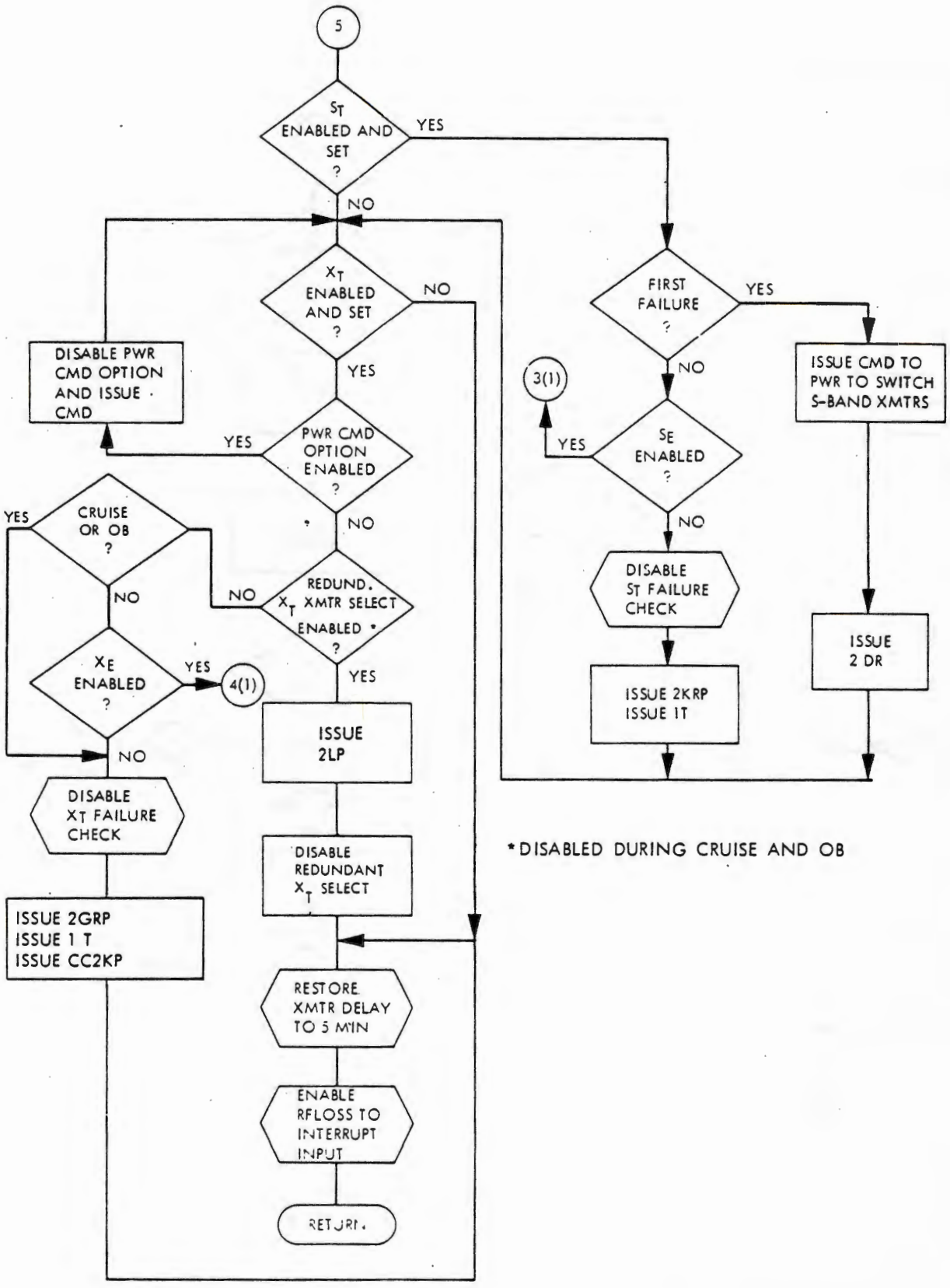


Figure 3-52. Flow Diagram of RFLOSS Routine (Sheet 2 of 2)

3.2.3 Output. This is Section (4) of Figure 2. All communication to the other spacecraft subsystems occurs in this section. Any commands or telemetry that are generated by the input or intermediate processing sections are buffered and issued by this section when the output units become available. The event output function handles CC's, DC's, FDS and AACS memory loads, and interprocessor communication, while the telemetry function handles telemetry, checksums, and memory readouts.

3.2.3.1 Event Output (OUTDRV). This routine is responsible for accepting, storing, and outputting CC and DC commands. Commands are generated by the normal operation of other routines within the CCS, or by the receipt of ground commands.

3.2.3.1.1 Description. The event output routine performs necessary functions associated with outputting commands: (1) accepts and stores commands from other routines or the ground; (2) outputs commands on a first-come, first-served basis when both output units are available; and (3) configures the output units as required to output the command.

The event output routine in each half of the CCS is designed to accommodate a "parallel" outputting scheme wherein commands out of the CCS may be output out of each half at approximately the same time. In order to prevent conflicts between each half outputting commands through an output unit at the same time, (particularly sets of commands to the AACS or FDS that must not be interrupted), the following control between the processors and output units has been provided for:

1) When a processor has a command or a set of commands to output through an output unit, it will first gain control of both units before attempting to output a command through its primary output unit. Controlling both output units will prevent the other processor from issuing commands through its primary output unit at the same time.

121

2) Although a command is output from the CCS from only one output unit at a time, the other output unit (secondary) will also receive the command but without the execute bit set. The non-executed command will appear in the CCS telemetry as well as the executed command if the telemetry register is available at the time.

3) Processor A will have output Unit 1 as its prime output unit, and Processor B will have output Unit 2 as its prime output unit.

4) In order to prevent one processor from grabbing control of the output units in the middle of a sequence of commands being output by the other processor (when the output units are temporarily "available" between individual commands), a scheme is utilized whereby an output unit is not truly available unless its data register contains all zeros. Therefore, a processor will not attempt to control an output unit until the data register is cleared by the controlling processor.

5) If a processor is unable to output a command for any reason within ten seconds after the need to do so arises (command in output buffer), the processor will try to gain control of the output units regardless of the state of the output unit data registers. If the processor still cannot output a command after an additional two to twenty seconds, the processor will go to ERROR. The actual time depends on the nature of the failure.

6) If a processor can successfully control its primary output unit but fails three times to gain control of the other output unit, the routine will assume a failure in the other output unit and command through its primary output unit only.

7) For a processor to initially gain control of the output units, it typically sends the command to be transmitted to each output unit as a <u>non-execute</u> command. When control of each output has been verified, the processor will output the

command again, as an execute command to its primary output unit and as a non-execute command to the other output unit.

8) Whenever a processor outputs a command with Bit 1=0, it does so as a non-execute command with a read initiate interrupt to the other processor. This type of command is typically used for interprocessor communication.

When a command is generated by some routine in the CCS, it is transferred to the event output routine by first loading the word into the accumulator register and then jumping to the appropriate entry location in the event output routine. The command is then inserted into the next available location within a 40 word command buffer. If the buffer is full, control will be transferred to the error routine which causes normal operation to cease.

When a command is input to the command buffer in the event output routine, the routine will unmask the "output unit available" interrupt in preparation for outputting the command. It will also check to see if the self-test routine is currently active such that a command can be output to an output unit. If it is not, the routine will initiate a self-test so that a command(s) can be output to an output unit.

The self-test routine is a necessary part of outputting commands because of a hardware interlock between processors in the CCS. The CCS was designed to allow each processor to be inhibited by the other processor. This inhibit prevents a processor from outputting commands whenever the inhibit is in effect. This capability was included to prevent a failed processor from running wild. Each processor has the capability of overriding the inhibit from the other processor for one second. This capability was included to prevent a failed processor from applying an irrevokable inhibit on a good processor. These inhibit/override capabilities are used as follows:

Each processor is normally inhibited by the other processor. Whenever a processor has a command to output, it must override the other processor's inhibit before it can do so. The hardware

15B

is designed such that a successful self-test must be performed before the inhibit-override is applied. This is why the event output routine initiates a self-test when it has a command to output. Because the inhibit-override is good for one second, a self-test is not repeated if an inhibit-override is still in effect at the time of the next command.

FDS commands are currently constrained to occur no oftener than 60 msec. Also, PWR has requested that commands to them be at least 110 msec. apart. To effect a delay between commands for these subsystems, the event output routine will output a dummy DC command after each command to those subsystems. The dummy DC will "use up" 110 msec. of time and prevent commands to those subsystems from being too close together.

Similarly, a required delay between AACS commands will be effected by a dummy CC command which will allow for approximately 15 msec of spacing.

These dummy commands overlay the previous command in the output buffer, and the buffer pointer remains on that location until after the dummy command is issued.

Ground commands transferred to the event output routine will be handled exactly like a preprogrammed command. That is, it will wait its turn in the command buffer and be subject to additional delays if previous buffer commands were for AACS, FDS or PWR.

Before a command is actually output by this routine to an output unit, the routine checks to see if bit-one of the command is a "zero". If it is, the "command" is really a data word to be transferred to the other processor. In this case, the routine will output the data word to the output unit and will instruct the output unit to send an "initiate read" interrupt to the other processor. The other processor will respond by retrieving the data word stored in the output unit.

When a command is output to an output unit, it is first loaded into the accumulator register, and then one of two special output instructions (INIT or READ) is executed. Execution of an output instruction causes output unit mode/control information as well as the accumulator data to be transferred to storage registers in the output unit.

Mode/control information is of the following types:

1) Which output unit (or both) is to execute the command.

2) Whether the command should be executed (sent to appropriate subsystem) or just accepted.

3) Whether the data sent to the output unit is for telemetry output only.

4) Whether the output unit should maintain communication with the same processor, or switch to the other processor, or become available for either processor.

5) Whether data already in the output unit is to be compared with new data from a processor.

6) Whether a "demand read" interrupt (not used) or an "initiate read" interrupt (interprocessor communication) is to be sent to the other processor.

FDS and AACS block loads are controlled by the AACS and FDS memory load routine (Paragraph 3.2.3.2). The event output routine will determine that a memory load is required and transfer to the AACS and FDS memory load routine.

Figure 3-53 illustrates the flow diagram for this routine.

3.2.3.1.2 Constraints. All commands to FDS and PWR will be followed by a dummy DC command to guarantee at least 110 msec. spacing between commands. All AACS commands will be followed by a dummy CC command to guarantee at least 15 msec. spacing between commands.

155

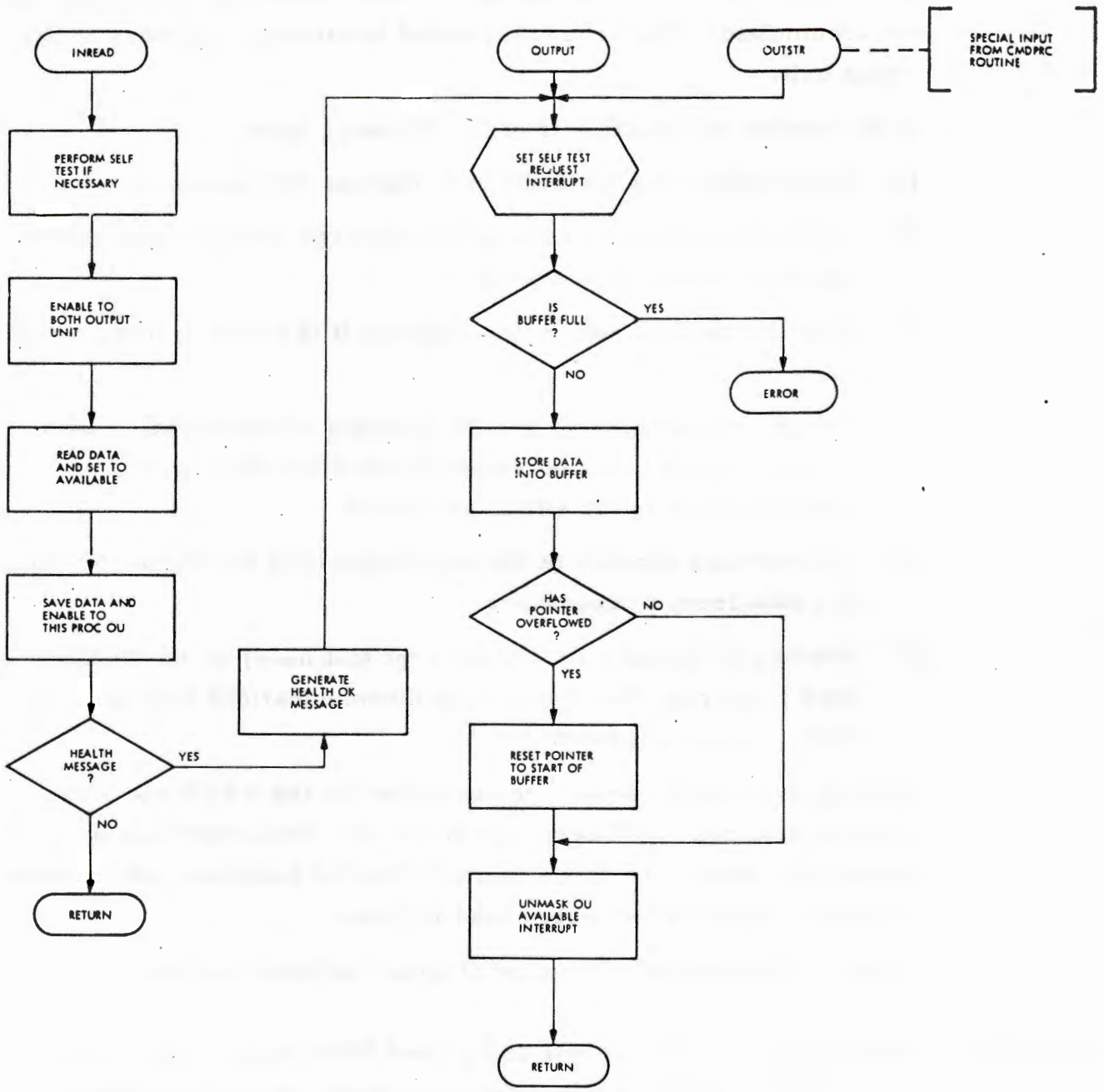


Figure 3-53. Flow Diagram of the Routine OUTDRV (Sheet 1 of 3)

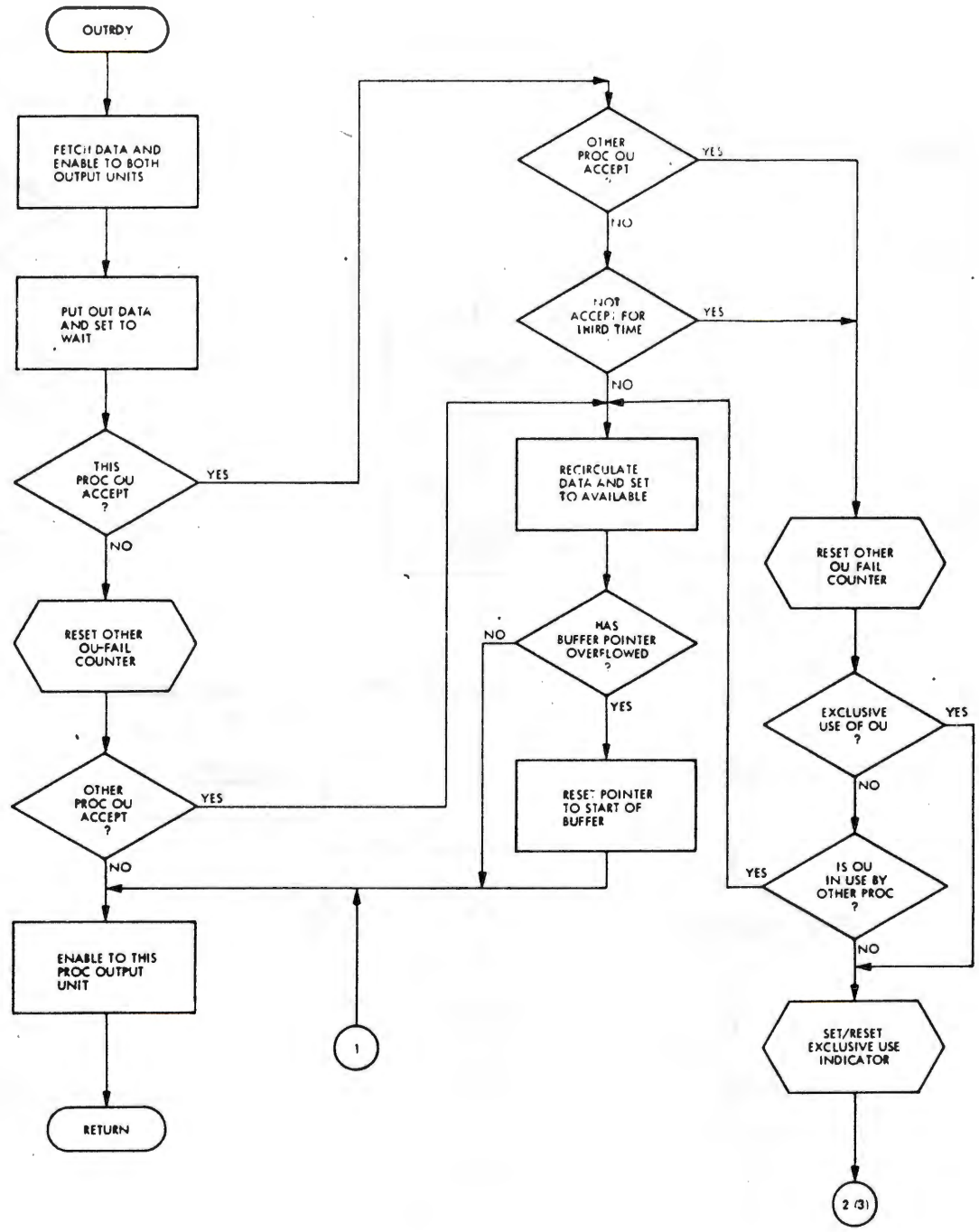


Figure 3-53. Flow Diagram of the Routine OUTDRV (Sheet 2 of 3)

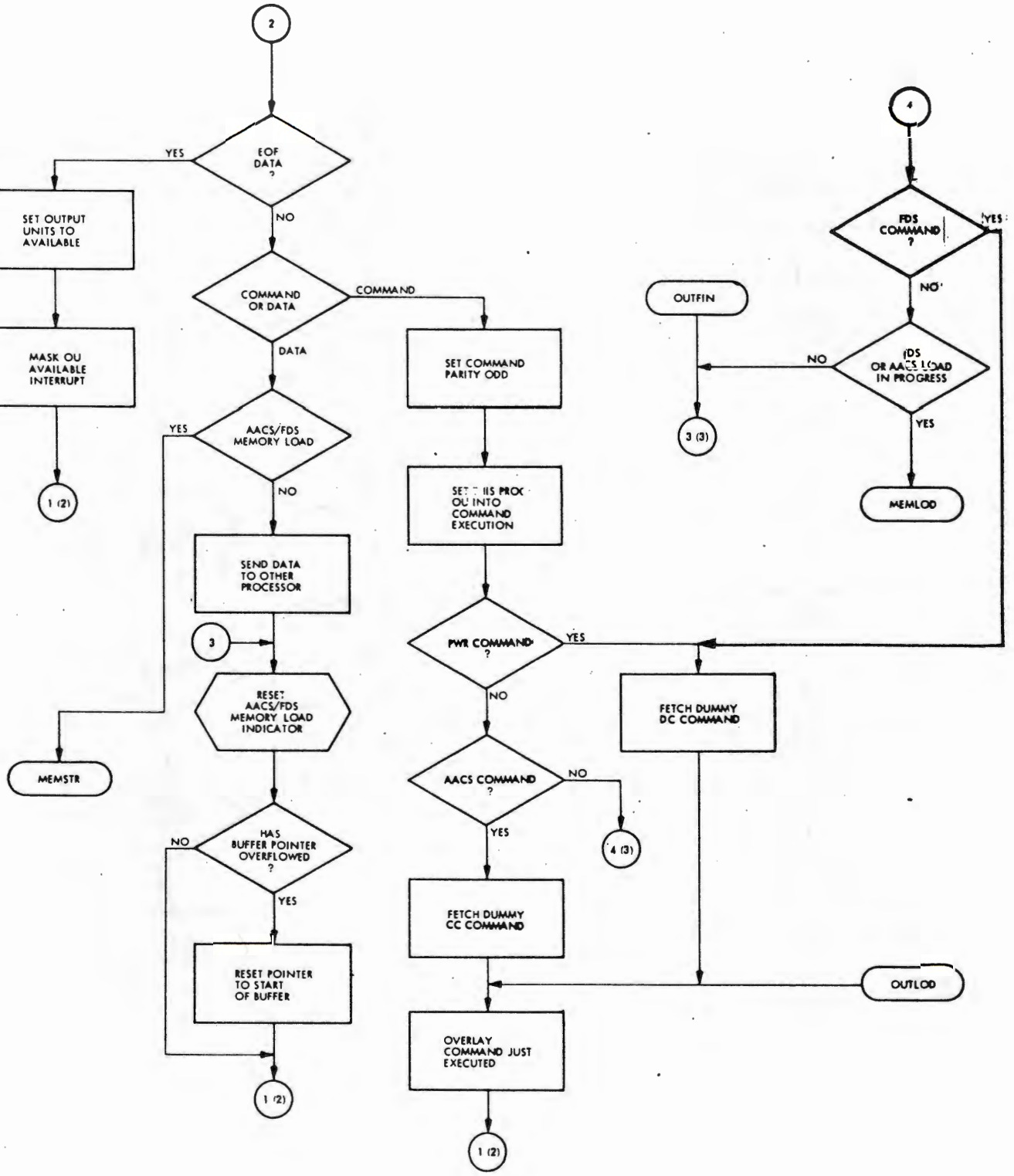


Figure 3-53. Flow Diagram of the Routine OUTDRV (Sheet 3 of 3)

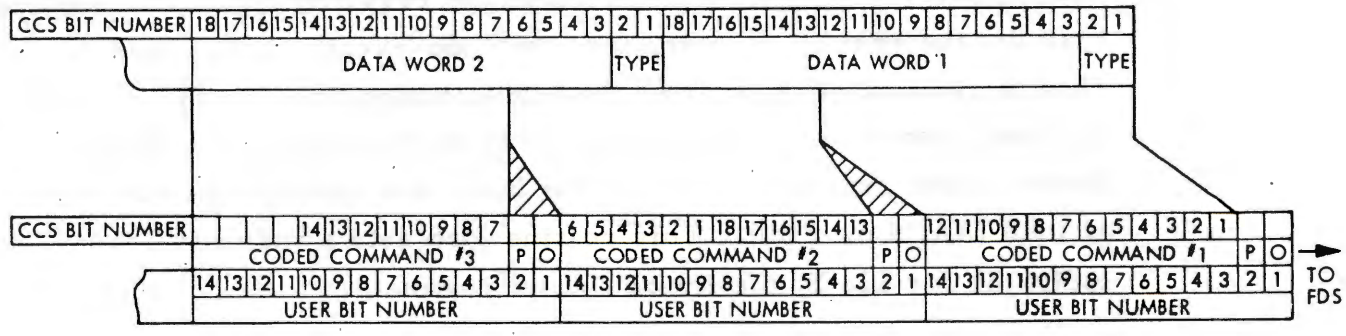
3.2.3.2 AACS and FDS Memory Load (MEMLOD). This routine processes memory loads to AACS and FDS by formatting the memory load data into the required coded commands.

3.2.3.2.1 Description. This routine is entered from the Event Output routine after that routine determines the requirement for a memory load (by the special memory load "pseudo-event" in the buffer). Upon activation, this routine first determines the size of the load, commands the output units to be available so that the Event Output routine can output the load, and determines the type of load (AACS or FDS). This routine also sets a memory load indicator for use by the Event Output routine.

If the load is for FDS, the routine will generate the FDS load "initiation" command and transfer it to the Event Output routine where it overlays the previous command in the output buffer. All commands associated with the memory load will be output from this same location in the output buffer. In effect, the Event Output routine is locked onto this location in the buffer until the load is completed. After the Event Output routine has issued the command and its associated dummy, it returns to this routine for the next load command (as long as the memory load indicator is set). This routine then continues generating load commands from the stored CCS data in accordance with Figure 3-54. When the last data command has been sent, this routine generates an all-ones "end of load" command.

If the load is for AACS, the routine must differentiate between address words and data words. The routine assumes that the first word of an AACS load is an address word with the six least significant bits indicating the number of words in the load before the next address word. (See Figure 3-55.) The routine then generates each AACS load command in accordance with Figure 3-56, sending each one in turn to the Event Output routine for outputting and generation of the required CC dummy command.

159



NOTES: 1) USER BIT #1 RECEIVED FIRST BY USER SUBSYSTEM
2) P REPRESENTS ODD PARITY ON USER BITS 3-14
3) USER BITS 1 AND 2 ARE GENERATED BY CCS AND NOT INCLUDED IN THE CCS BLOCK LOAD

Figure 3-54. FDS Command Word to CC Conversion

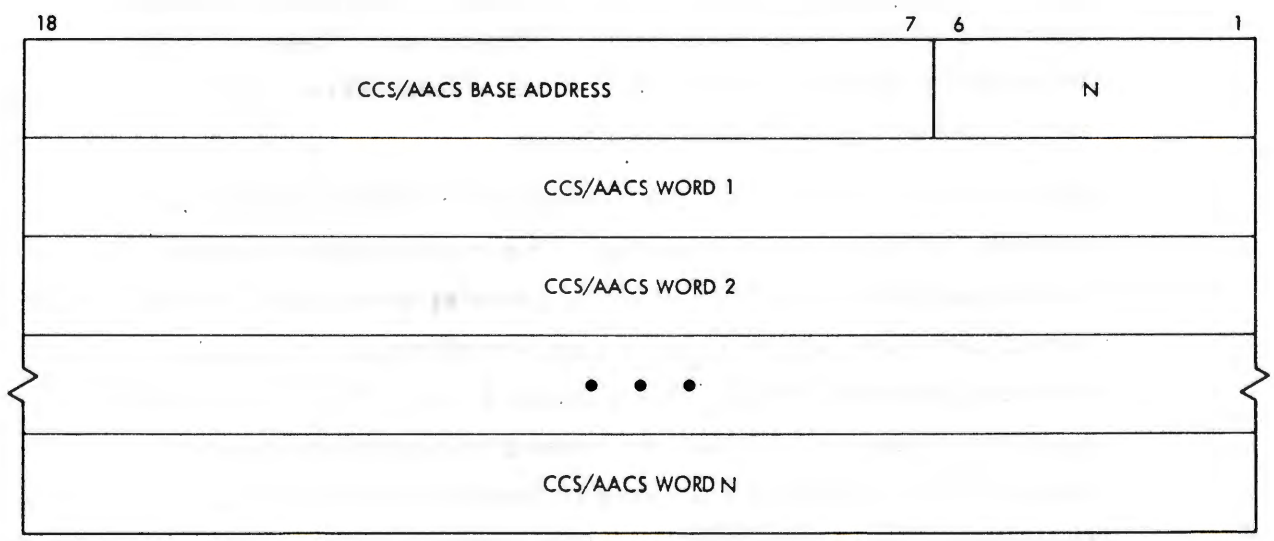
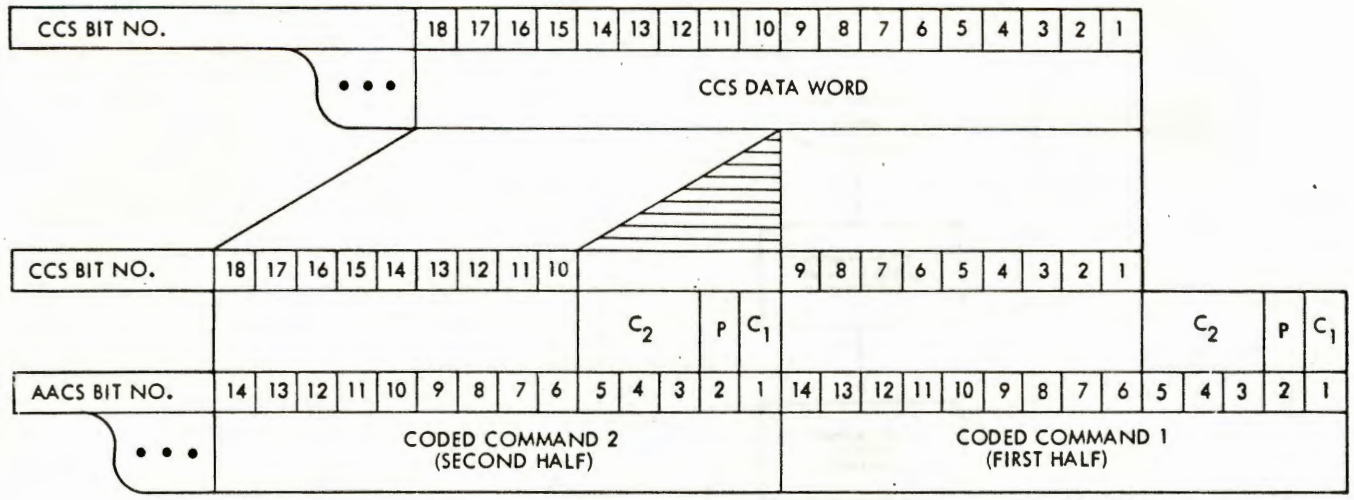


Figure 3-55. CCS/AACS Memory Load Format (N ≤ 35)



NOTES: (1) USER BIT 1 RECEIVED FIRST BY USER SUBSYSTEM
(2) C_1 AND C_2 ARE CODE BITS GENERATED BY CCS AND NOT INCLUDED IN THE CCS BLOCK LOAD
(3) P REPRESENTS ODD PARITY ON USER BITS 1-14

Figure 3-56. CCS Command Word to AACS CC Conversion

After an AACS or FDS memory load is completed, this routine returns to the Event Output routine to clear the memory load indicator and position the output buffer pointer.

Figure 3-57 illustrates the flow diagram for this routine.

3.2.3.3 Telemetry Generation (TLMDRV). The Telemetry Generation routine is responsible for accepting, storing, and outputting processor telemetry words. These telemetry words are generated by normal or abnormal operation of other CCS routines. The processor-generated telemetry words are shown in Figure 3-58. Each telemetry word has a unique 6-bit telemetry identification field.

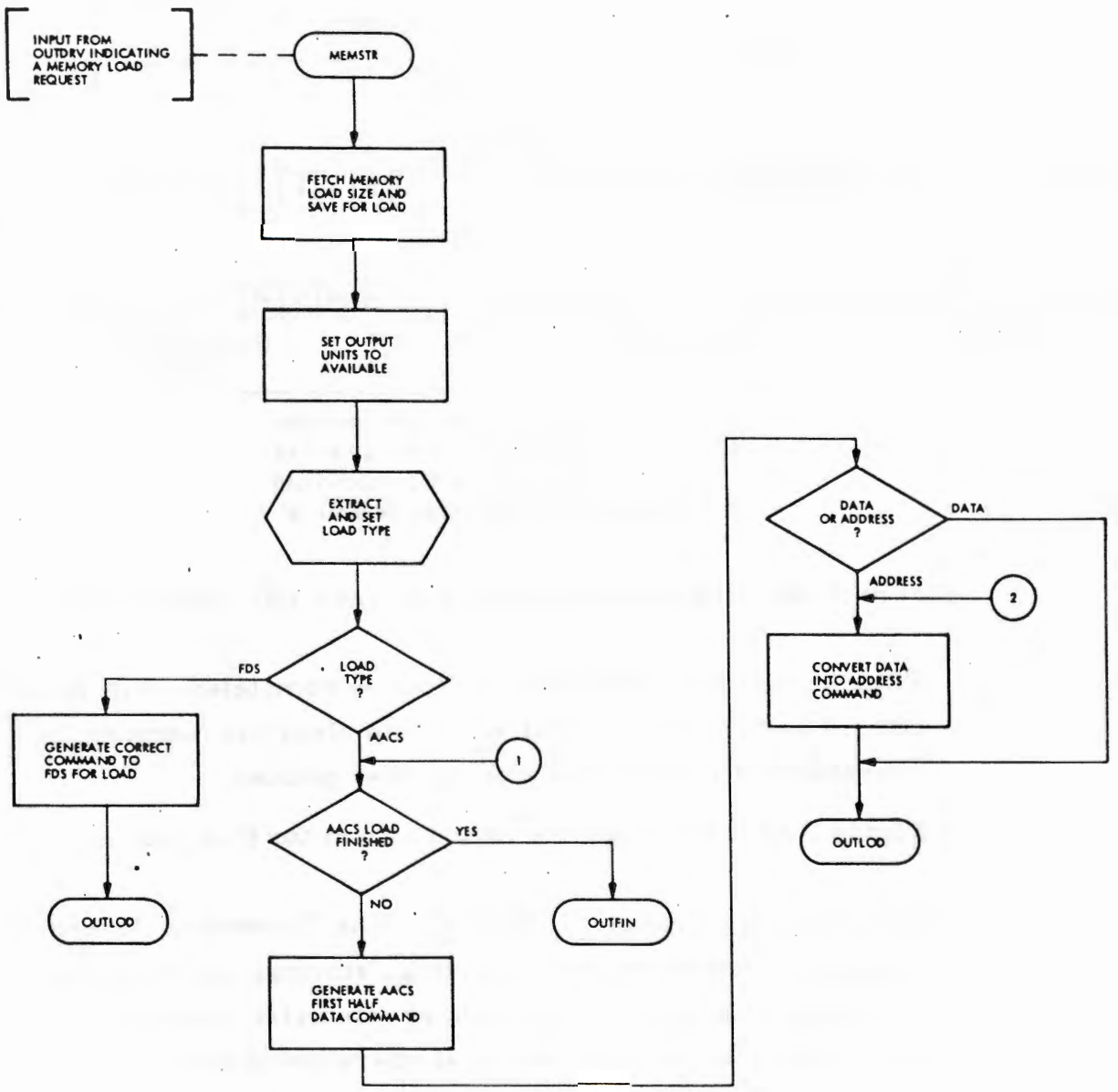


Figure 3-57. Flow Diagram of the Routine MEMLOD (Sheet 1 of 2)

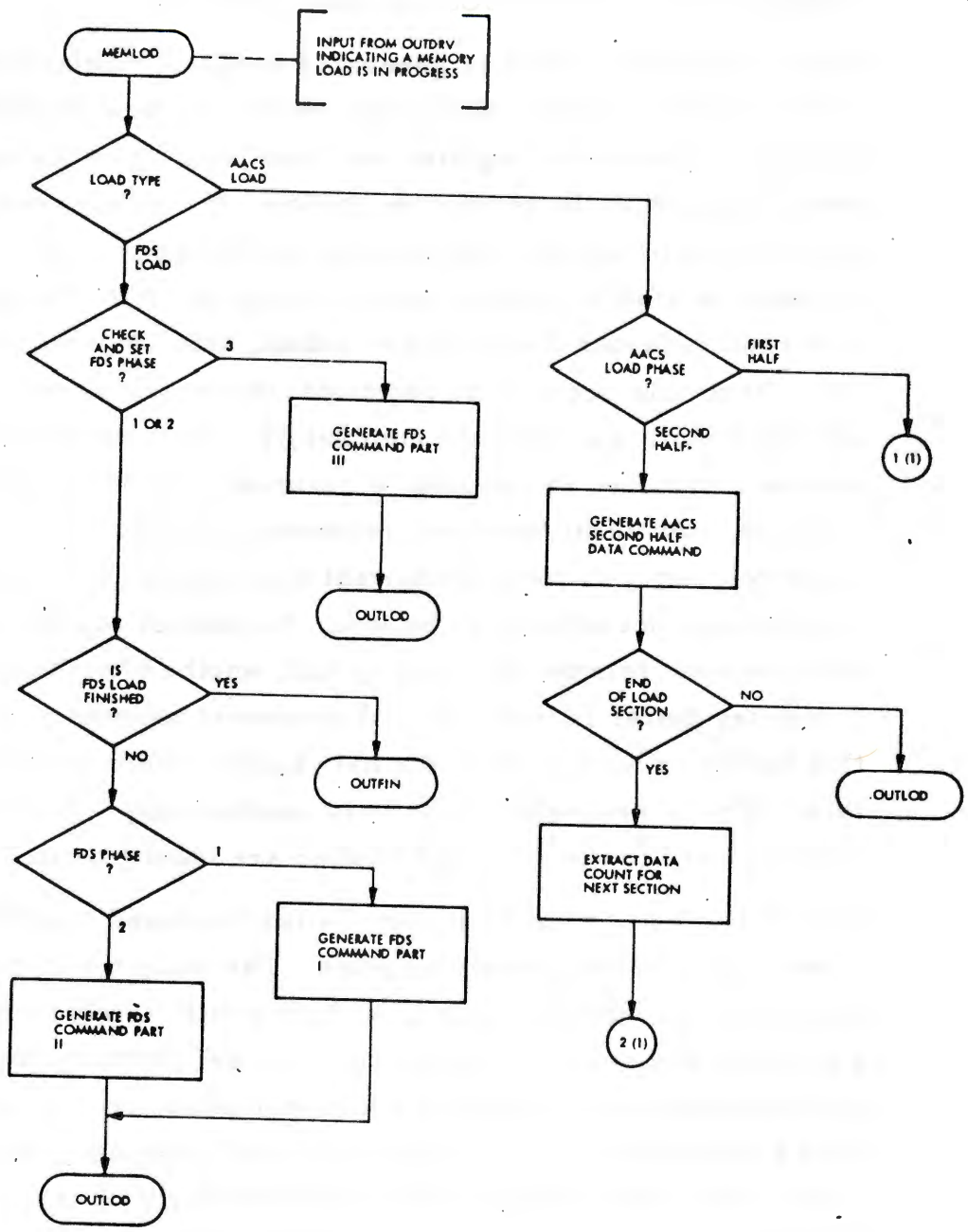
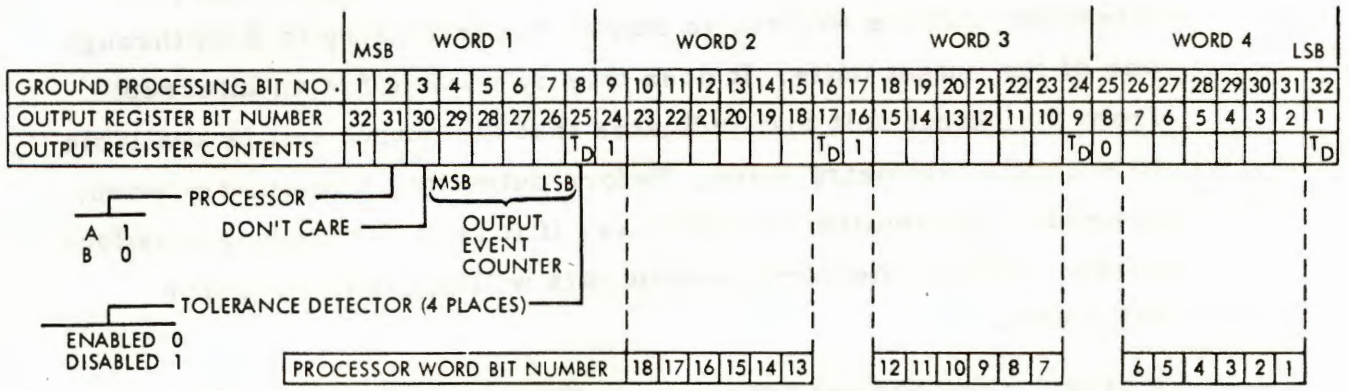


Figure 3-57. Flow Diagram of the Routine MEMLOD (Sheet 2 of 2)

3.2.3.3.1 Description. The Telemetry Generation routine performs functions associated with telemetry output: (1) accepts and stores telemetry words from other routines (or itself); (2) outputs telemetry on a first-in first-out basis when an output unit is available; (3) controls CCS memory readouts.

When a telemetry word is generated by some routine in the CCS, it is transferred to the telemetry routine by first loading the word into the accumulator register and then jumping to the appropriate entry location in the telemetry routine. If the telemetry word is to be treated as "normal" telemetry, it is stored in the normal 14-word telemetry buffer, only. If telemetry words are input to this routine faster than they are output, this 14-word buffer may fill. When this occurs, no additional telemetry words will be accepted until it is emptied completely. After being emptied, normal telemetry processing is resumed. While it is being emptied, additional "normal" telemetry will be lost. Because some telemetry is more important than others, it is highly desirable to not allow it to be lost. To protect against this when the "normal" telemetry buffer is full, another "critical" 10-word telemetry buffer is utilized. All important telemetry is stored in this buffer as well as the "normal" buffer. Telemetry stored in this buffer is accessed by memory readout only. Telemetry words routed to the "critical" buffer are identified in Figure 3-58.

Once full, the normal telemetry buffer is closed to additional telemetry until completely emptied. The reason for this is that each time the buffer is filled, a "buffer full" telemetry word is generated and stored in the critical buffer. If the normal buffer is not emptied before accepting additional telemetry, it might keep filling repeatedly, with many "buffer full" telemetry words being generated. This could fill the critical buffer with these words, to the exclusion of other, more important telemetry. The "buffer full" telemetry word is generated after a full buffer has been emptied and restored to normal operation. This telemetry word contains a count of the number of telemetry words lost while the buffer was being emptied.



NOTES:

1. THE PROCESSOR WORD MAY CONTAIN ONE OF SEVERAL DATA STATEMENTS INCLUDING THE FOLLOWING:

PROCESSOR WORD BIT NUMBER	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
BASE COMMAND ① ③	T	ID	E	C	CMD ID								0	0	0	0	0	1
BLOCK COMMAND ① ③	R	ID	E	C	CB	SEQ	SEQ NO.					BLK ID	0	0	0	0	1	0
CCS MEMORY DUMP ⑤	SECTION START ADDRESS												0	0	0	1	0	0
	CCS WORD 1																	
	CCS WORD 2																	
	CCS WORD 3																	
	⋮																	
	CCS WORD 8																	
CHECKSUM	12 MSB OF CHECKSUM WORD												0	0	0	1	1	1
CHECKSUM	12 LSB OF CHECKSUM WORD												0	0	1	0	0	0
MASTER AUTOMATIC ENTRY COUNT ②	ENTRY COUNT												0	0	1	1	1	0
UPLINK COMMAND COUNT ②	COMMAND COUNT												0	0	1	0	1	1
CC & DC EVENT COUNT ②	EVENT COUNT												0	0	1	1	0	1
TELEMETRY BUFFER FULL ④	WORDS LOST COUNT												1	0	0	1	1	0
CDU LOCK CHANGE COUNT ②	LOCK CHANGE COUNT												1	0	0	1	0	1
DTR TIC COUNT	TIC COUNT												0	1	1	1	1	1

2. SYMBOLS

T	SET TO 1	IF COMMAND CAUSED TERMINATION
R	SET TO 1	IF COMMAND WAS REJECTED
C	SET TO 1	IF BASE COMMAND WAS CORRECTED
CB	SET TO 1	IF BLOCK COMMAND WAS CORRECTED
ID	SET TO 1	IF PROCESSOR OR S/C ID WAS INCORRECT
E	SET TO 1	IF COMMAND START HAD ONE ERROR
SEQ	SET TO 1	IF COMMAND BLOCK WAS OUT-OF-SEQUENCE
CMD ID	BITS 18-11 OF ACCUMULATOR WORD	
BLK ID	BITS 8-7 OF ACCUMULATOR WORD	
SEQ NO.	BITS 4-1 (SEQ NO.) OF ACCUMULATOR	

① ISSUED 10 BIT TIMES AFTER LAST BIT OF COMMAND UNLESS THE T OR R BIT IS SET THEN IT IS ISSUED AFTER THE LAST BIT OF THE COMMAND

② ISSUED EVERY CCS HOUR PULSE

③ INCLUDED IN CRITICAL TELEMETRY BUFFER IF COMMAND CAUSED TERMINATION OR WAS REJECTED

④ INCLUDED IN CRITICAL TELEMETRY BUFFER

⑤ MEMORY IS READ OUT IN 8-WORD SECTIONS. THE THREE LSBs OF THE SECTION START ADDRESS ARE ALWAYS ZERO

3. TOLERANCE DETECTOR STATUS:

IF WORD IS FROM OU1, TOLERANCE DETECTOR STATUS IN PWR SUPPLY B IS INDICATED
IF WORD IS FROM OU2, TOLERANCE DETECTOR STATUS IN PWR SUPPLY A IS INDICATED

4. FOR DATA TRANSMISSION PROCESSOR WORD HAS HIGHEST PRIORITY

Figure 3-58. CCS Processor Telemetry Word

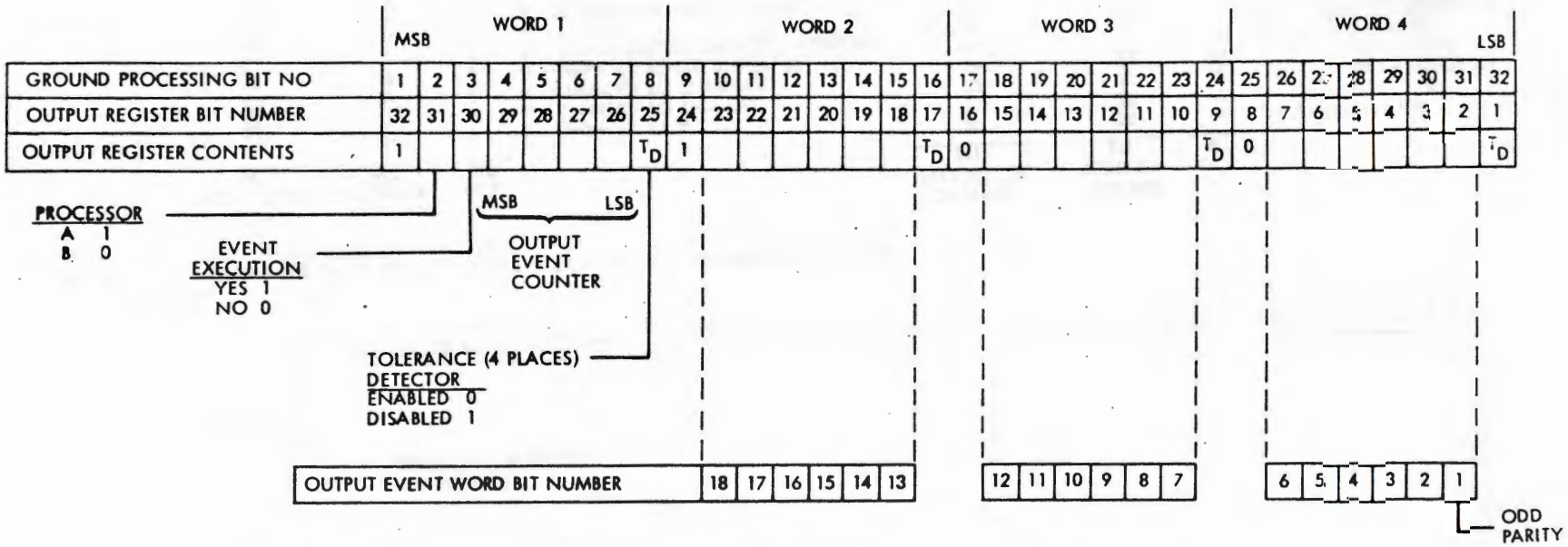
When the normal telemetry buffer contains telemetry data, the telemetry routine will try to output this telemetry to FDS through one of the output units. It does this by waiting for an interrupt from the output unit that indicates that the output unit is available to accept a telemetry word. Before outputting a telemetry word, however, the routine checks to see if it is in the memory readout mode. If it is, memory readout data will pre-empt regular telemetry.

Whenever the normal telemetry buffer is empty, the CCS output units will output event telemetry or status telemetry in that order. Figures 3-59 and 3-60 illustrate those telemetry words.

This routine controls a memory readout of the memory locations within the range of addresses specified by the data contained in the CCS accumulator register at the time the memory readout is started. A memory readout is started by transferring control (jumping) to a fixed memory readout start address in the telemetry routine (MEMDMP).

Prior to entering the routine, the accumulator register in the CCS must be loaded with the nine most-significant bits of the starting address and the number of eight-word blocks to be read out, as shown in Figure 3-61. Because only nine bits of address information is included in the address field, the telemetry routine "fills in" the missing three least significant bits. These three bits define eight contiguous memory locations beginning at the starting address as follows:

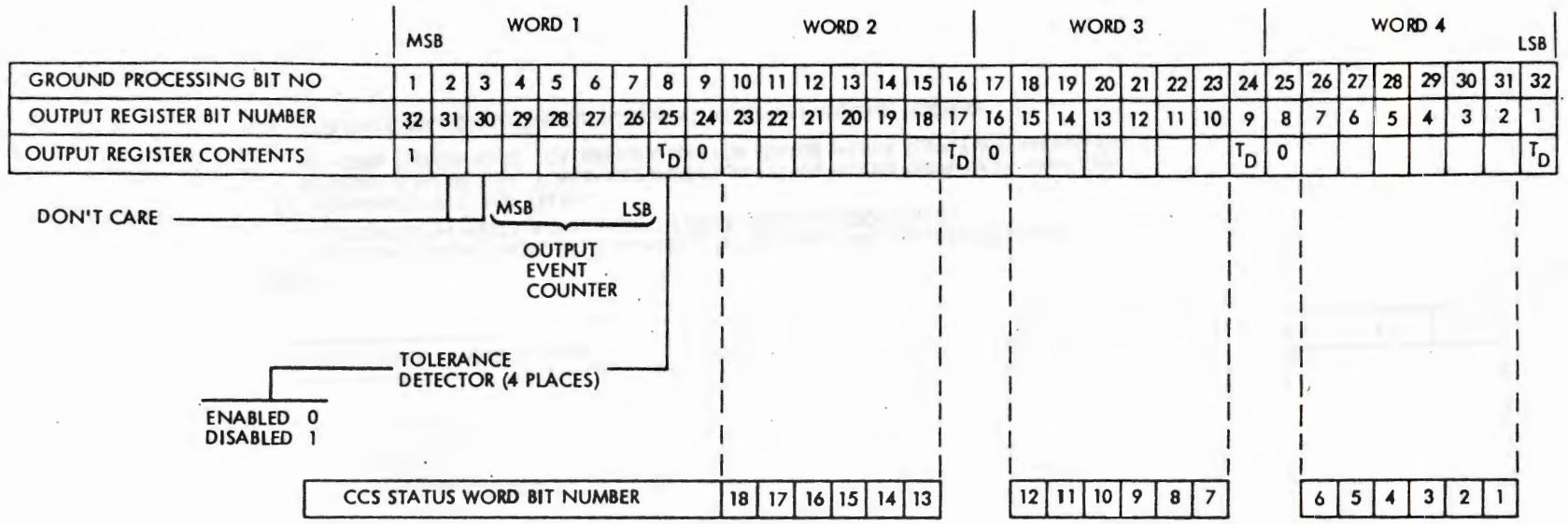
START ADDR. FILL-INS	
(Location 1)	XXXXXXXXX000
(Location 2)	001
(Location 3)	010
(.)	.
(.)	.
(.)	.
(Location 8)	XXXXXXXXX111



NOTES:

1. THE OUTPUT EVENT WORD IS EQUIVALENT TO THE 18-BIT ACCUMULATOR WORDS ILLUSTRATED IN MJS77-3-290, EXCEPT FOR LSB WHICH IS ODD PARITY
2. TOLERANCE DETECTOR STATUS:
   IF WORD IS FROM OU1, TOLERANCE DETECTOR STATUS IN PWR SUPPLY B IS INDICATED
   IF WORD IS FROM OU2, TOLERANCE DETECTOR STATUS IN PWR SUPPLY A IS INDICATED
3. FOR DATA TRANSMISSION OUTPUT EVENT WORD HAS SECOND PRIORITY

Figure 3-59. CCS Output Event Telemetry Word



NOTES:

1. THE STATUS WORD BITS DESCRIBE THE FOLLOWING CONDITIONS AND THE TITLE IS DENOTED BY A "1" STATE.

GROUND PROCESSOR BIT	OUTPUT REGISTER BIT	CCS BIT		GROUND PROCESSOR BIT	OUTPUT REGISTER BIT	CCS BIT	
21	12	9	B - ENABLED BY A	10	23	18	A - INTERNAL ERROR
22	11	8	B - ENABLED TO THIS OU	11	22	17	A - POWER FAILURE
23	10	7	B - ENABLED TO OTHER OU	12	21	16	A - COMMAND ERROR (P)
26	7	6	A - MEMORY PROTECT ACCESS	13	20	15	B - INTERNAL ERROR
27	6	5	A - 2-8 INTERRUPT OVERRIDE ENABLED	14	19	14	B - POWER FAILURE
28	5	4	A - ACTIVE	15	18	13	B - COMMAND ERROR (P)
29	4	3	B - MEMORY PROTECT ACCESS	18	15	12	A - ENABLED BY B
30	3	2	B - 2-8 INTERRUPT OVERRIDE ENABLED	19	14	11	A - ENABLED TO THIS OU
31	2	1	B - ACTIVE	20	13	10	A - ENABLED TO OTHER OU

2. TOLERANCE DETECTOR STATUS:

   IF WORD IS FROM OU1, TOLERANCE DETECTOR STATUS IN PWR SUPPLY B IS INDICATED
   IF WORD IS FROM OU2, TOLERANCE DETECTOR STATUS IN PWR SUPPLY A IS INDICATED

3. FOR DATA TRANSMISSION STATUS WORD HAS LOWEST PRIORITY

Figure 3-60. CCS Status Telemetry Word

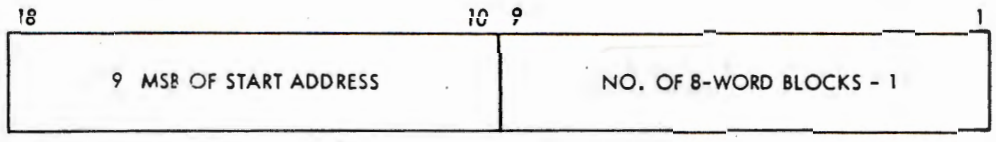


Figure 3-61. Memory Readout Data Word

The telemetry routine does not presently control the FDS or any other subsystem that might be affected by CCS readouts. Any mode changes and the like required of other subsystems must be controlled by other software within the CCS or by ground command.

The flow diagram for this routine is illustrated in Figure 3-62.

3.2.3.3.2 Constraints

1) Memory readouts must be specified in blocks of eight contiguous memory locations.

2) During busy periods, some telemetry may be lost if the normal telemetry buffer fills.

3) Memory readouts at bit-rates above 1 Kbps may have status telemetry words mixed with memory readout words due to the CCS's inability to output memory words that fast.

3.2.3.4 Checksum (CHKSUM). The CCS checksum routine is used as an aid in verifying that the contents of the CCS memory are as desired. The checksum routine is typically used at the end of a memory load period and is initiated by the conditional execute block which is typically the last block of a CCS update. This routine sums memory words in blocks of eight and transfers the calculated sum to the TLMDRV routine for output to the FDS.

19

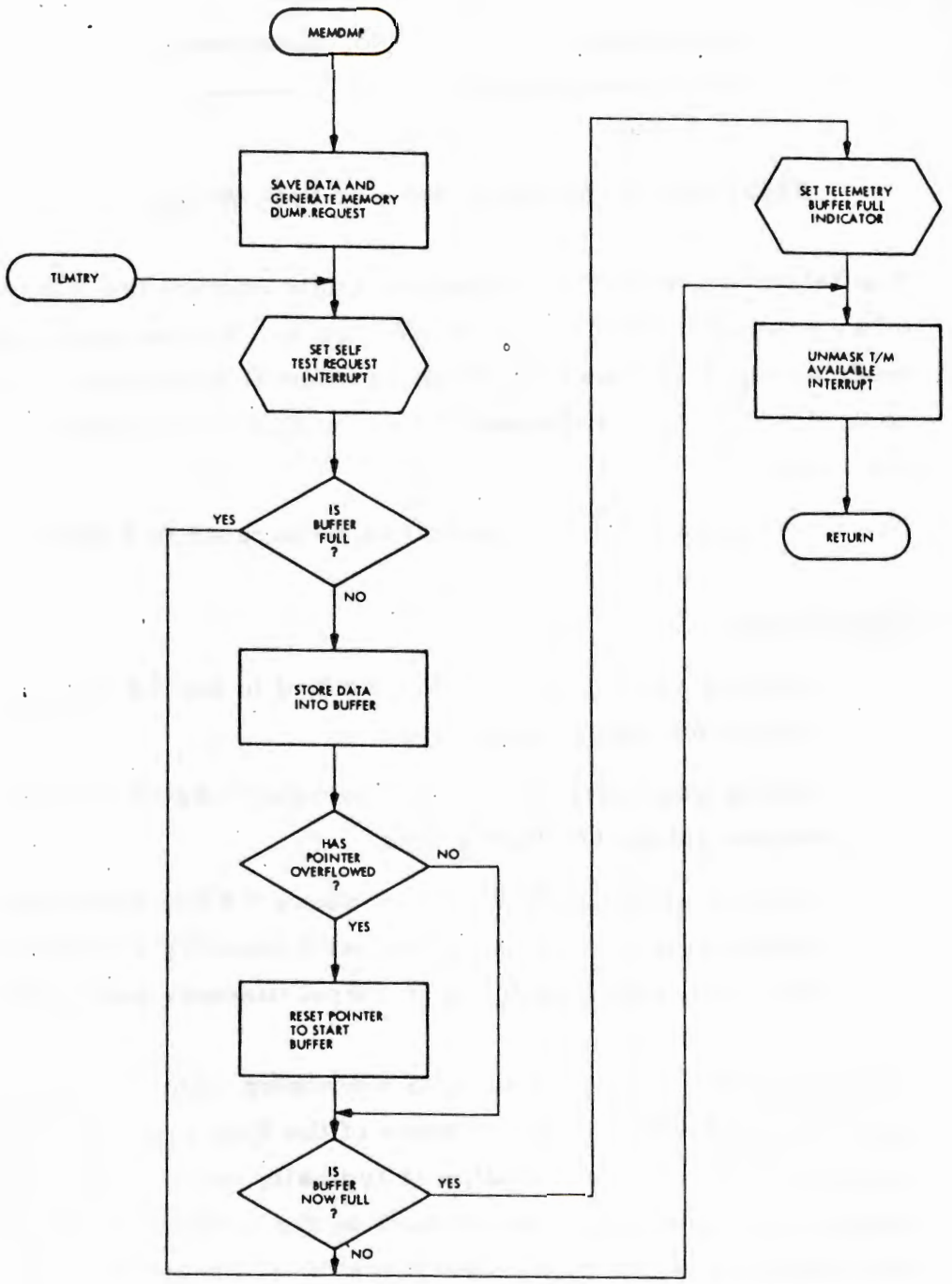


Figure 3-62. Flow Diagram of the Routine TLMDRV (Sheet 1 of 2)

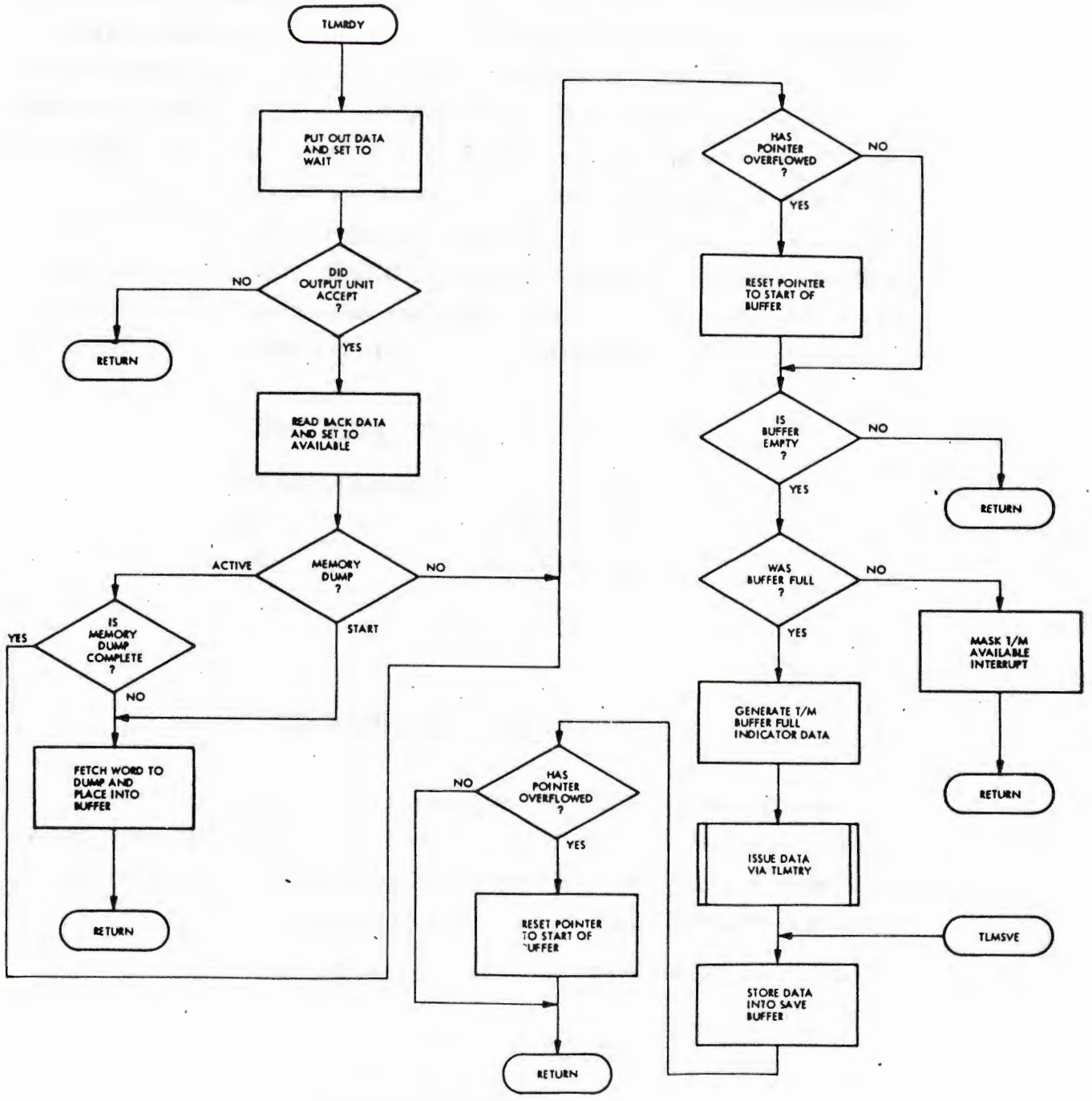


Figure 3-62. Flow Diagram of the Routine TLMDRV (Sheet 2 of 2)

471

3.2.3.4.1 Description. This routine forms the arithmetic sum of the data in the memory locations within the range of addresses specified by the data contained in the CCS accumulator register at the time the checksum routine is started. The checksum routine is started by transferring control (jumping) to a fixed checksum start address in the CCS memory (CHKSUM). Prior to entering the routine, the accumulator register in the CCS must be loaded with the nine most significant bits of the starting address and the number of eight-word blocks to be summed, minus one, as shown in Figure 3-63. Because only nine bits of address information is included in the address field, the checksum routine "fills in" the missing three least significant bits. These three bits define eight contiguous memory locations beginning at the starting address as follows:

START ADDR. FILL-INS

(Location 1)	XXXXXXXXX000
(Location 2)	001
(Location 3)	010
(.)	.
(.)	.
(.)	.
(Location 8)	XXXXXXXXX111

At the completion of the summing process, the eighteen least significant bits are split into two telemetry words with identifiers, and transferred to the TELEMETRY routine for output to FDS. Figure 3-64 illustrates these telemetry words.

Figure 3-65 illustrates the flow diagram for this routine.

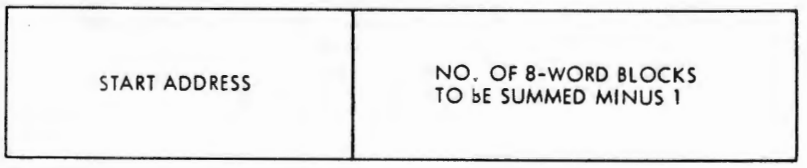


Figure 3-63. Accumulator Data for Checksum Start and Stop

172

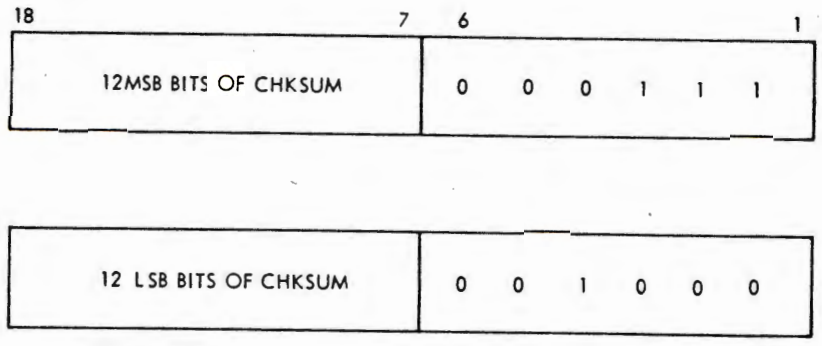


Figure 3-64. Telemetry Checksum Words

3.2.3.4.2 Constraints. Checksums must be specified in blocks of eight contiguous memory locations. Care must be exercised in selection of the eight-word blocks such that memory locations not adequately simulated on the ground are not included in the summing operation, or correct checksum verification cannot be guaranteed.

3.2.4 Processor Constants

3.2.4.1 Global Constants (GLBCNT). GLBCNT is a collection of data words which is assembled as a convenience to the coding of other routines. The data words include arithmetic constants, bit masks, identifier codes, and dummy (assembler) data definitions.

The format used for labeling these data words is as follows·

1) FMXX – in general, this label format is used for an identifier code, typically for merging with data which will be output via telemetry.

2) BTXXYY – this label format is used for data words which have zeroes in all bit positions other than those from the

173

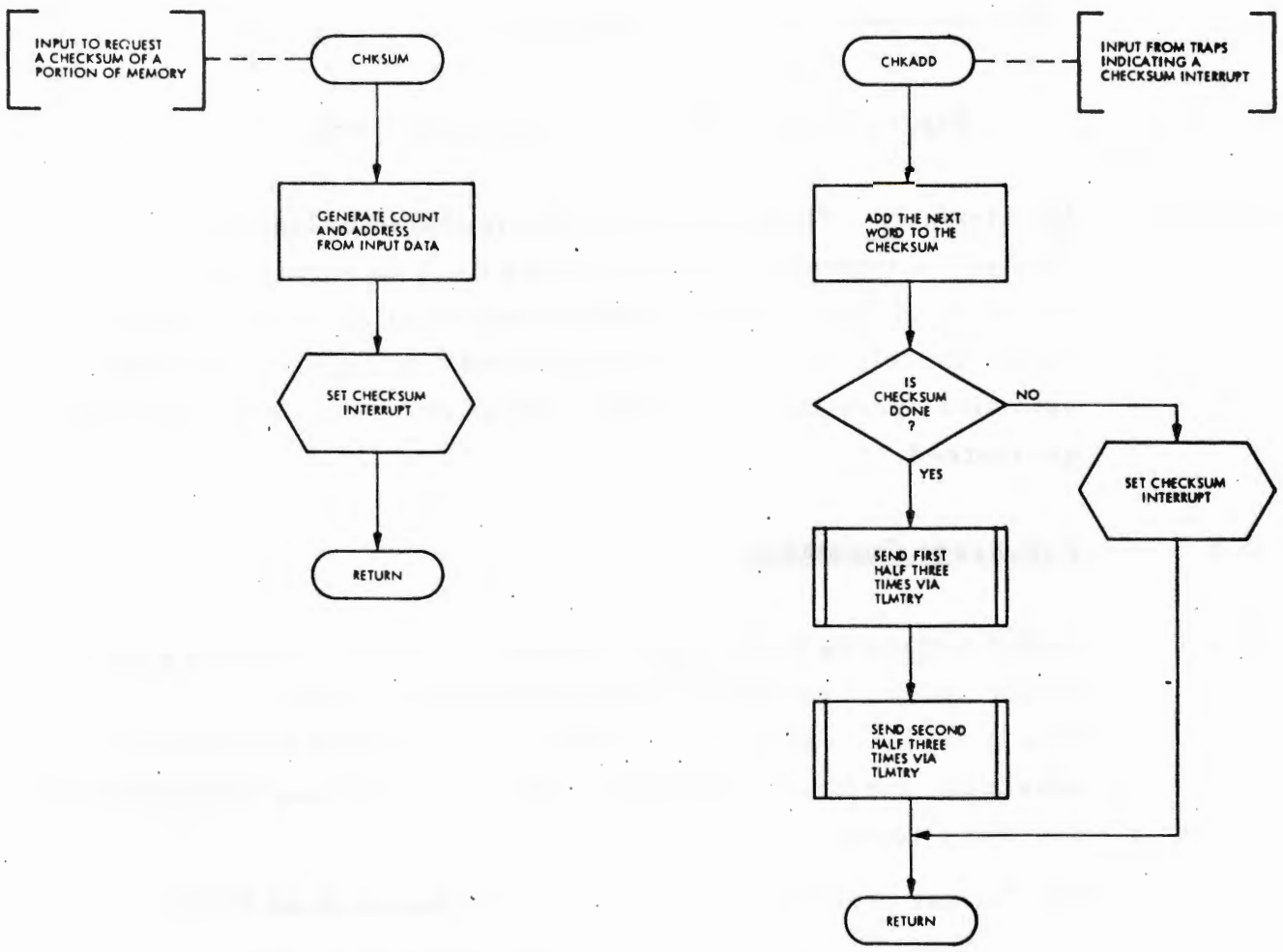


Figure 3-65. Flow Diagram of the Routine CHKSUM

high order XX to the low order YY positions in the 18-bit word. For example, BT1615 is the label assigned to the pattern $(140000)_8$.

3) BNXXYY – this label format is used for data words which have ones in all bit positions outside the range from XX through YY. For example, BN1615 is the label assigned to the pattern $(637777)_8$.

This routine is loaded into the write-protected portion of the CCS memory.

3.2.4.2 Processor Usage Dependent Constants (PARMXY). Those constants that control the functional operation of the CCS software have been placed in specific assembly decks. With this organization, the main load module of the CCS software is constant regardless of the processor or spacecraft in which it is used. The role which the main load module executes is controlled by the parameters in the small assembly decks.

One control deck is required for each processor of each spacecraft. The control decks are labeled PARMXY, where X is the identification number (2, 3) of the Voyager spacecraft, and Y is the identification letter (A, B) of the CCS processor to which the control deck is assigned. Thus, the PARM2A deck is assigned to CPU A of Voyager 1.

3.2.5 Variables (VARABL). The memory locations which are used for data storage by the various CCS routines are collected together and assembled as the VARABL deck. This assembly is loaded into the read/write (unprotected) portion of the CCS memory.

In addition to the "working storage" locations for the CCS routines, VARABL contains time/event regions for the support of CCS routines. The time/event regions are located in this routine for convenience of reprogramming events, via ground command, at different points in the mission.

175

Two labeling conventions were adopted in the coding of the VARABL routine:

1) RG<u>XX</u> – this label identifies a storage location used as a general register. RG12 identifies general register 12.

2) XXXWYY – this label identifies a storage location used by a particular routine. The <u>XXX</u> field is filled with the first three letters of the name given to the assembly deck for the routine. For example, CHKW2 is a storage location used by the checksum (CHKSUM) routine.

APPENDIX A

OPTIONAL ROUTINES

APPENDIX A

OPTIONAL ROUTINES

The routines described in this appendix will not be resident in the CCS memory at launch but will be available if needed during flight. The associated listings for these routines can be found in Appendix A of Volume II.

A-1. DTR AND HEATER POWER SHARE PATCH (DRHSHR)

This is an optional routine to be loaded when it has been determined that the 5 watts which are required for DTR tape motion change (except ready or power off) are not available in the power profile. The routine shall turn off selected heaters (CC21CR and CC36CRP) before allowing the OUTDRV routine to issue a CC16AP and any command to the DSS except READY mode. The routine shall also turn off these same heaters when the DTR is within two TICS of BOT or EOT and is in the record or play mode. Ten seconds after issuing the last occurrence of the heater off commands, the routine shall then turn on the same heaters (CC36CP and CC21C).

The routine is to be loaded in an unused position of read-write memory. When verified that it is properly loaded, the following locations must also be altered, in order, as follows: load location OUTLOD-3 in the routine OUTDRV with TRNZ OUD135 unless the optional routine DTRCCB is loaded then instead, load location DTRCCB+1 in the routine DTRCCB with TRNZ OUD135; load location TSTCAL+6 in the routine COINTS with TRA COI137, and load location OUT3+1 in the routine OUTDRV with TRA OUD118.

When this routine is no longer required, restore the following locations to their original value, in order, as follows: load loaction OUT3+1 in OUTDRV, location TSTCAL+6 in COINTS, location DTRCCB+1 in DTRCCB or location OUTLOD-3 in OUTDRV.

The flow diagram for this routine is shown in Figure A-1.

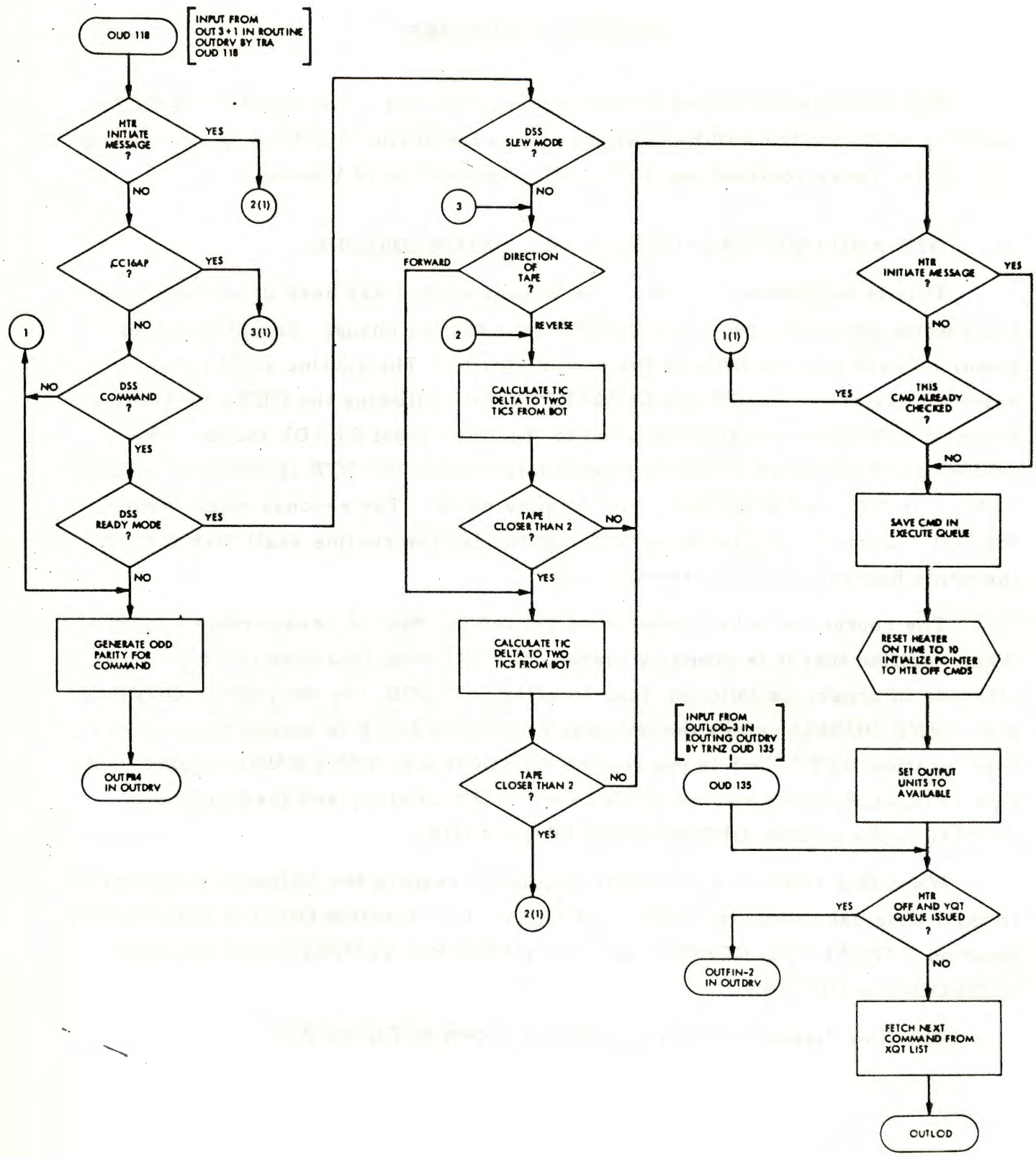
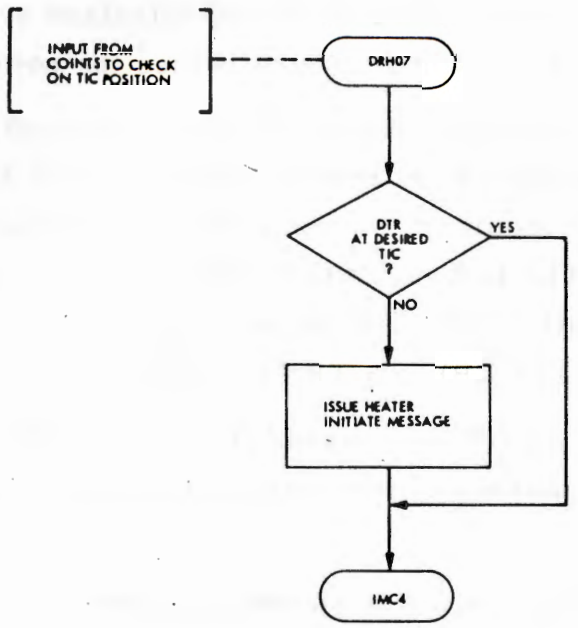


Figure A-1. Flow Diagram of the Optional Routine DRHSHR (Sheet 1 of 2)



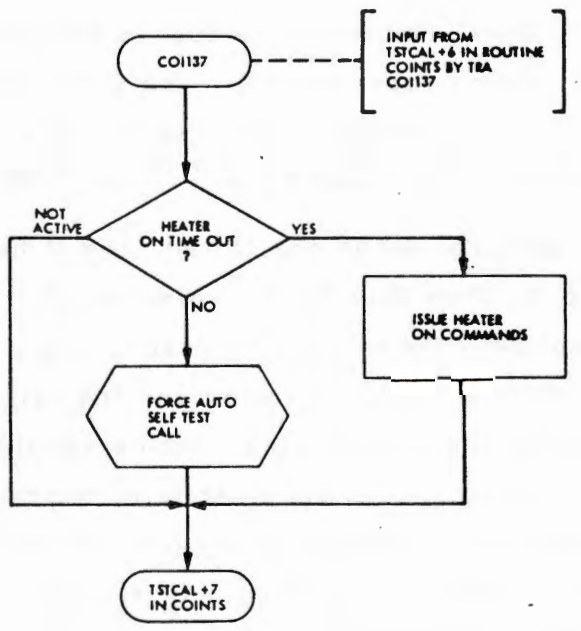


Figure A-1. Flow Diagram of the Optional Routine DRHSHR (Sheet 2 of 2)

180

A-2. DTR COMMAND INTERFACE VIA FDS BACKUP (DTRCCB)

This is an optional routine to be loaded when the primary CCS to DSS command interface fails. It utilizes the FDS to DSS interface by issuing the appropriate DSS command to FDS, which in turn issues the command to DSS.

When the CCS to DSS interface fails, a DC16XR command to isolate the failed interface should be issued. Then this routine should be loaded in an unused position of read-write memory. When verified that it is properly loaded, load the location OUTLOD-3 in the OUTDRV routine with a TRNZ DTRCCB. If the optional routine DRHSHR is also loaded, change the location DTRCCB+1 in this routine (DTRCCB) to a TRNZ OUD135.

This routine will issue <u>all</u> DSS commands first to the DSS via the failed (disabled) interface (an idiosyncrasy of the routine) then the same command to the FDS (an SC16DA).

The flow diagram for this routine is shown in Figure A-2.

A-3. EMERGENCY LOAD OF OTHER PROCESSOR VIA OUTPUT UNIT (EMLOAD)

This is an optional routine to be loaded into a processor when the need arises to load the other processor when the other processor's normal command decode function is not operating. This routine is a dedicated subroutine of TARMEX and requires a one-word patch to the OUTDRV routine.

When the other processor has failed, and it has been determined that it is safe to unclamp it, then this routine may be used in the good processor. If it is not safe to unclamp the other processor, i.e., it will issue unwanted commands, then this routine cannot be used and the failed processor will have to be assumed permanently disabled. This routine should be loaded into an unused portion of the good processor's read-write memory. After verifying that this routine is properly loaded, patch the location OUT3 in the OUTDRV routine with a TREA EMLCHK. Also, the location 24_8 should be loaded with a NOP instruction so that the DMLOAD routine in the good processor will be disabled.

NOTE

Location EML02+8 of this routine would be IEX '1002 if the failed processor normally communicated with output Unit 1 or an IEX '2002 if it communicated with output Unit 2, or a NOP if both processors normally communicated with the <u>same</u> output unit.

Prior to sending any data to the other processor, it must be unclamped by an IEX '0052. To execute an instruction in the failed processor, execute the following pseudo event in TARMEX in the good processor.

18 17 16 15 14 13 12 11 10 9 8 7	6 5 4 3 2 1
EMLINS	1 0 0 1 0 0
INSTRUCTION TO BE EXECUTED	
ACCUMULATOR DATA	

To load data in the failed processor's read-write memory, execute the following pseudo event in TARMEX in the good processor.

18 17 16 15 14 13 12 11 10 9 8 7	6 5 4 3 2 1
EMLOAD	1 0 0 1 0 0
BLOCK SIZE (N)	
CCS DATA WORD 1	
CCS DATA WORD 2	
.	
.	
.	
CCS DATA WORD N	

NOTE

Format of data words is the same as the description of the UPLINK format of CCS memory loads (Scatter Load) (See Figures 3-9 of this document.)

While this routine is sending data to the other processor, the output unit buffer is still available for commands, i.e., this data transfer has a lower priority than CC's, DC's, AC's and SC's.

When this routine is no longer required, the failed processor should be clamped by executing an IEX '0402 in the good processor, and changing locations 24_8 and OUT3, also in the good processor, to their original value.

The flow diagram for this routine is shown in Figure A-3.

A-4. AACS MEMORY LOAD VIA MAM (MEMMAM)

This is an optional routine to be loaded when it has been determined necessary to load the AACS via the MAM subassembly. The routine will allow loading of the AACS FCP memory via the MAM in the same manner as in loading the memory via the normal CCS to AACS interface. When loading the FCP via the MAM, the uplink block type shall be 12_{10} (14_8). See Figure A-4 for the new operator word associated with an AACS MAM load. The format of the block shall be identical to that of a normal AACS block load (block type 8_{10}).

This routine is to be loaded in an unused portion of read-write memory. When verified that it is properly loaded, the following locations should be loaded, in order as follows: Load location MEMLOD+1 is the routine MEMLOD with TRNZ MEM1O, and location MEMSTR+12_{10} in the routine MEMLOD with TRZ MEM12.

Before issuing any memory loads to the AACS via the MAM, make certain that the MAM commons are set, (DC77X) that the MAM is powered, that the NON-PRIME FCP is on (DC77P) and that the MAM is in the write mode (CC77CW2X).

When this routine is no longer required, restore the following locations to their original value in the following order; location MEMSTR+12_{10} in MEMLOD and location MEMLOD+1 in MEMLOD.

The flow diagram for this routine is shown in Figure A-5.

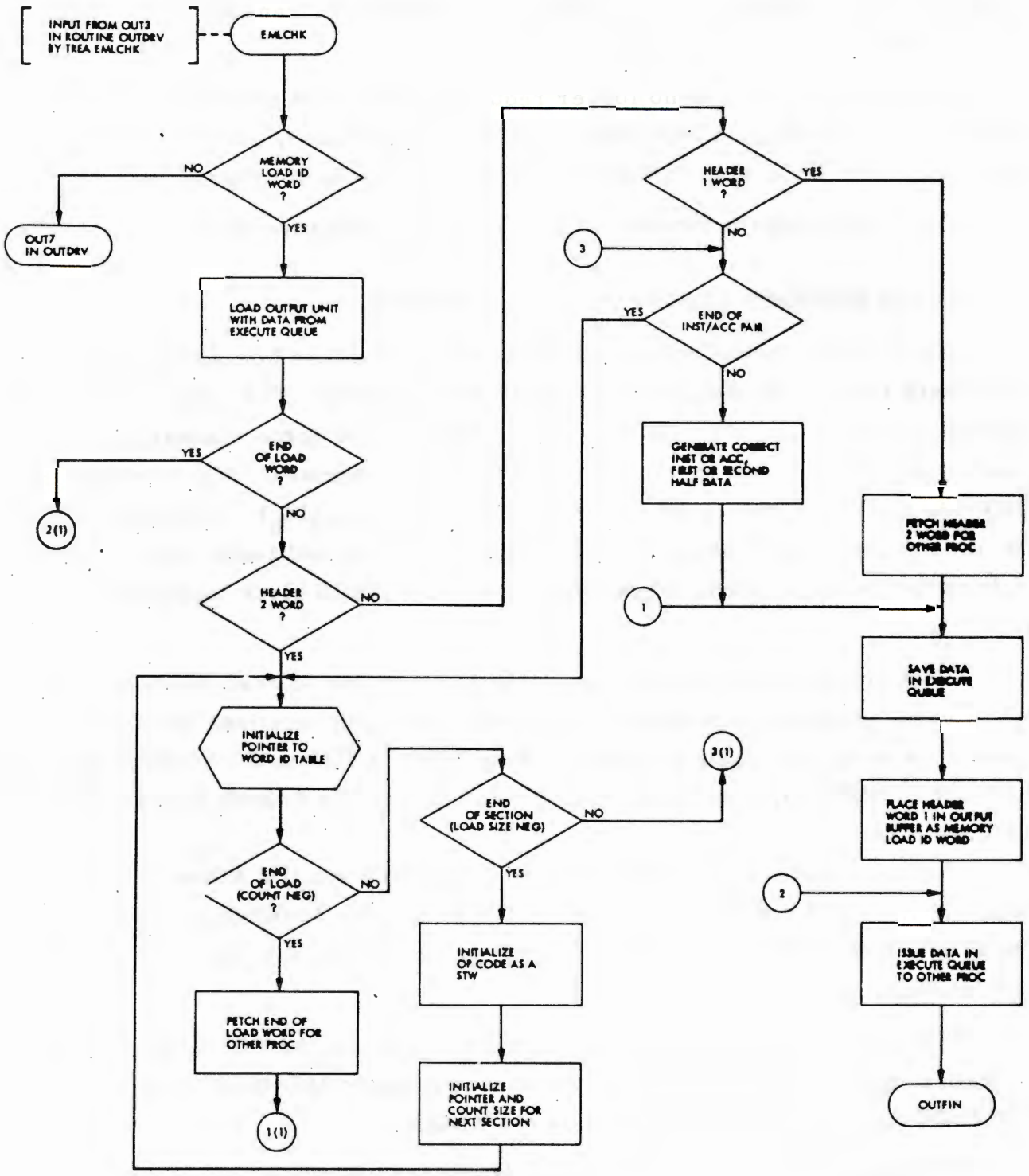


Figure A-3. Flow Diagram of the Optional Routine EMLOAD (Sheet 1 of 2)

10<

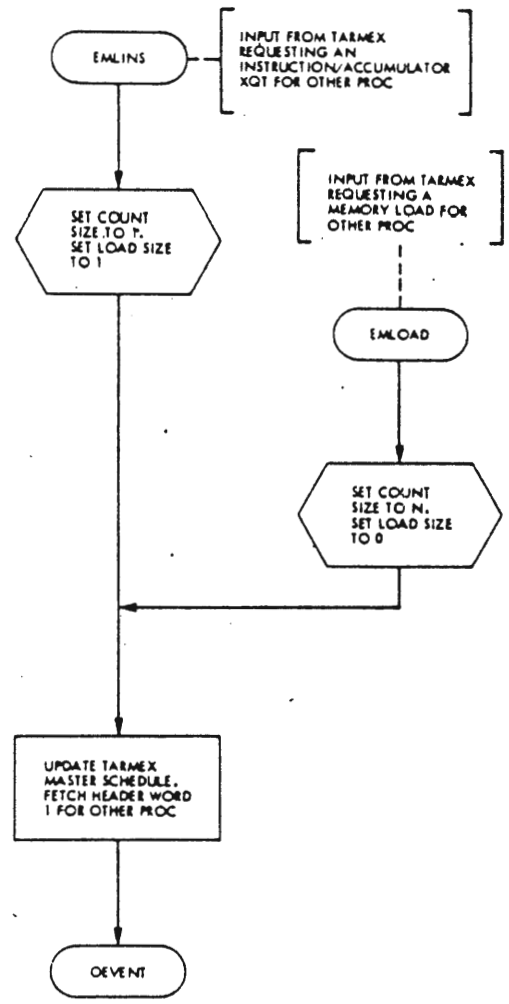


Figure A-3. Flow Diagram of the Optional Routine EMLOAD (Sheet 2 of 2)

156

	GROUND CMD BIT NUMBER	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
	CCS BIT NUMBER	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
CC OR DC	CC OR DC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1			
PC	STORE	1	MEM ADDRESS											0	1	1			
	TRANSFER	MEM ADDRESS												0	1	0			
	INTERNAL EXECUTE	OPERAND												1	0	0			
	BLOCK TYPE 0 COND. CCS EXECUTE	0	0	0	0	0	0	BLOCK SIZE						0	0	0			
	BLOCK TYPE 1 COND. EVENT EXEC	0	0	0	1														
	BLOCK TYPE 2/3 CCS LOAD	0	0	1	X														
	BLOCK TYPE 4/5 CCS LOAD	0	1	0	X														
	BLOCK TYPE 6 CCS EXECUTE	0	1	1	0														
	BLOCK TYPE 7 EVENT EXEC	0	1	1	1														
	BLOCK TYPE 8 AACS LOAD (NORM)	1	0	0	0														
	BLOCK TYPE 12 AACS LOAD (MAM)	1	1	0	0														
	BLOCK TYPE 11 FDS H/W LOAD	1	0	1	1														
	BLOCK TYPE 10/14 FDS PRI MEM LOAD	1	A	1	0														
	BLOCK TYPE 9/13 FDS SEC MEM LOAD	1	A	0	1														
	RETURN	MEM ADDRESS												1	0	1			
	STORE INDIRECT	MEM ADDRESS												1	1	0			
	EXECUTE	MEM ADDRESS												1	1	1			

A = 1 FOR MEM PROTECT OVERRIDE

X = DON'T CARE

MODIFIER

MODIFIER ID

PROCESSOR ID { 1 = PROCESSOR B; 1 = PROCESSOR A

SPACECRAFT ID { 0 = S/C 31; 1 = S/C 32

Figure A-4. CCS Operator Words

Change #1
6/11/79

ICT

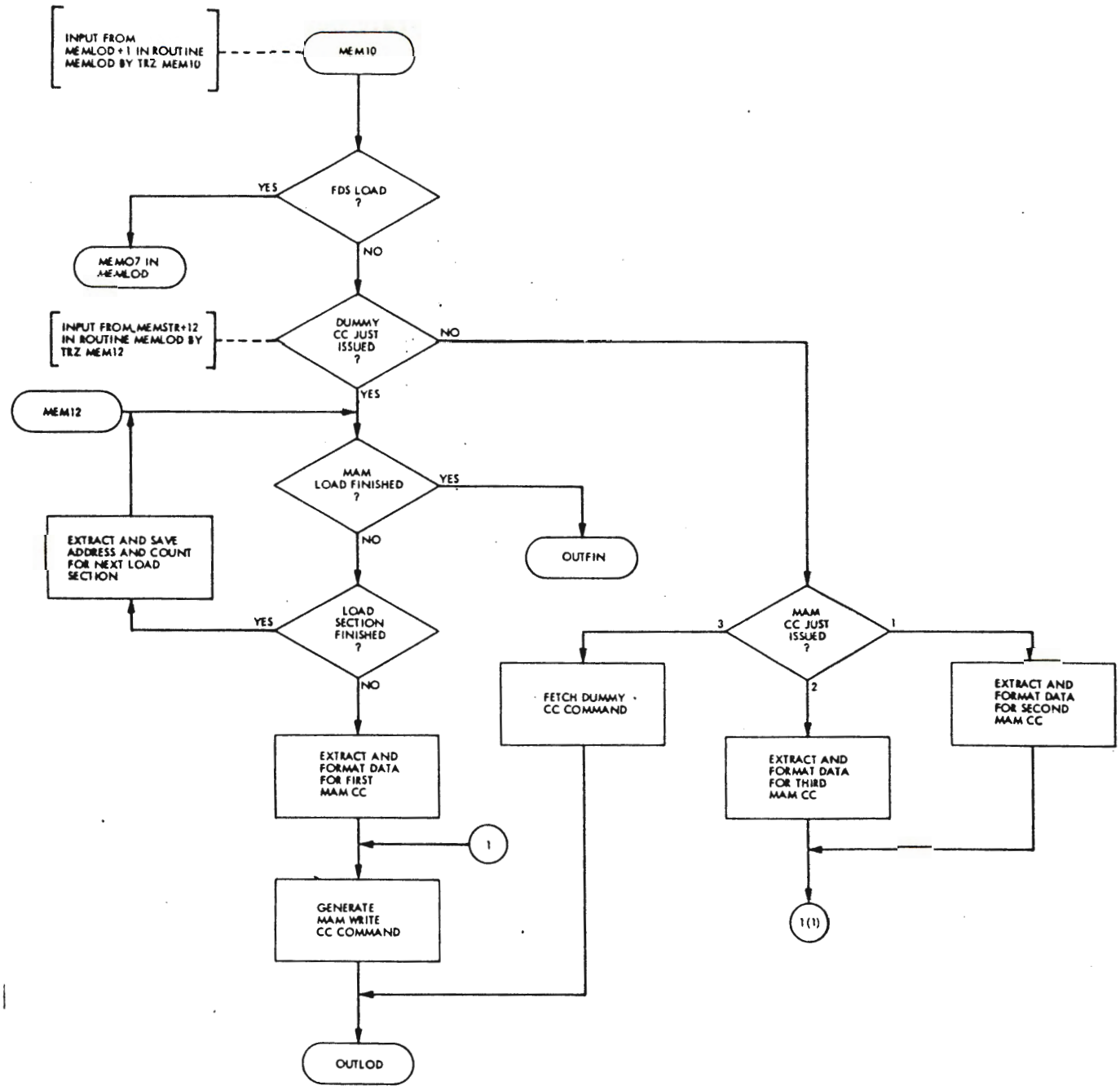


Figure A-5. Flow Diagram of the Optional Routine MEMMAM

180

A-5. DSS MOMENTUM CANCELLATION ROUTINE (DSSCAN)

This is an optional routine to be used to cancel DSS tape start-up momentum transients to the spacecraft by issuing special commands to the AACS whenever tape start-up and run-down to or from high-rate occurs. This routine also responds to tape reversals at BOT or EOT if the tape is moving in high-rate. This routine may be removed from the CCS if not needed for an extended period of time.

The commands issued to AACS by this routine shall be as follows:

(1) A CC7VC4 command shall be issued whenever the DSS is commanded to high-rate, forward direction, from low-rate or ready.

(2) A CC7VC5 command shall be issued whenever the DSS is commanded to high-rate, reverse direction, from low-rate or ready.

(3) A CC7VC4 command shall be issued whenever the DSS is commanded to low-rate or ready from high-rate, reverse direction.

(4) A CC7VC5 command shall be issued whenever the DSS is commanded to low-rate or ready from high-rate, forward direction.

(5) Two CC7VC4 commands shall be issued whenever the DSS is at high-rate, reverse direction, and a tape reversal occurs at BOT. One command shall be issued at each BOT that the CCS receives from the DSS.

(6) Two CC7VC5 commands shall be issued whenever the DSS is at high-rate, forward direction, and a tape reversal occurs at EOT. One command shall be issued at each EOT that the CCS receives from the DSS.

The routine shall also check DSS ready commands and store direction "presets" if that option is used in conjunction with tape sequences.

Additionally, start-ups and run-downs that occur after a BOT/EOT (outside the tape sensor) as a result of a low-rate or ready command shall also be taken into consideration by this routine to eliminate unnecessary commands to AACS.

This routine shall respond to DSS record and ready commands only, and shall ignor playback and slew requests, This routine shall assume that high-rate tape reversals without an intermediate ready command (except at BOT/EOT) shall never be commanded by on-board sequences or ground commands.

This routine is to be loaded in an unused portion of read-write memory. When verified that it is properly loaded, the following locations must also be altered, in order, as follows:

(1) Load Location OUTLOD-3 in the routine OUTDRV with TRNZ DSS01.

(2) Load Location DTRBET in the routine DTRPRC with STL RG3.

(3) Load Location DTR03 in the routine DTRPRC with TRA DSS07.

If the optional routine DRHSHR is also loaded, change the location DSS01+1 in this routine (DSSCAN) to TRNZ OUD135. Also change the locations as indicated in (1), (2) and (3) above.

If the optional routine DTRCCB is also loaded (after DSSCAN), change the following locations in this routine (DSSCAN) after DTRCCB is loaded:

(1) Change location DSS02-1 to TRA DTRCCB+2

(2) Change location DSS05-2 to TRZ LNKSAV

(3) Change location DSS05-1 to TRA DTRCCB+2

(4) Change location DSS02+3 to TRA DTRCCB+2

(5) Change location DSS03+1 to TRNZ DTRCCB+2

(6) Add the following code to the DTRCCB routine:

```
LNKSAV  STW   RG2       SAVE AACS CMD MOMENTARILY
        LOAD  RG1       RETRIEVE LINK IN RG1
        STW   RG3       TEMPORARILY STORE IN RG3
        LOAD  RG2       RETRIEVE AACS CMD
        TRA   OUTPUT    STORE IN OUTPUT BUFFER
        LOAD  RG3       RETRIEVE PREVIOUS RG1 LINK
        STW   RG1       STORE IN RG1
        TRA   DTRCCB+2  GO TO DTRCCB ROUTINE TO OUTPUT
                        DSS CMD THRU FDS
```

Locations DTRCCB and DTRCCB+1 can be eliminated in the DTRCCB routine. Also, location LNKSAV+7 can be eliminated if DTRCCB and DTRCCB+1 are eliminated and LNKSAV through LNKSAV+6 immediately preceed DTRCCB+2.

The flow diagram for this routine is shown in Figure A-6.

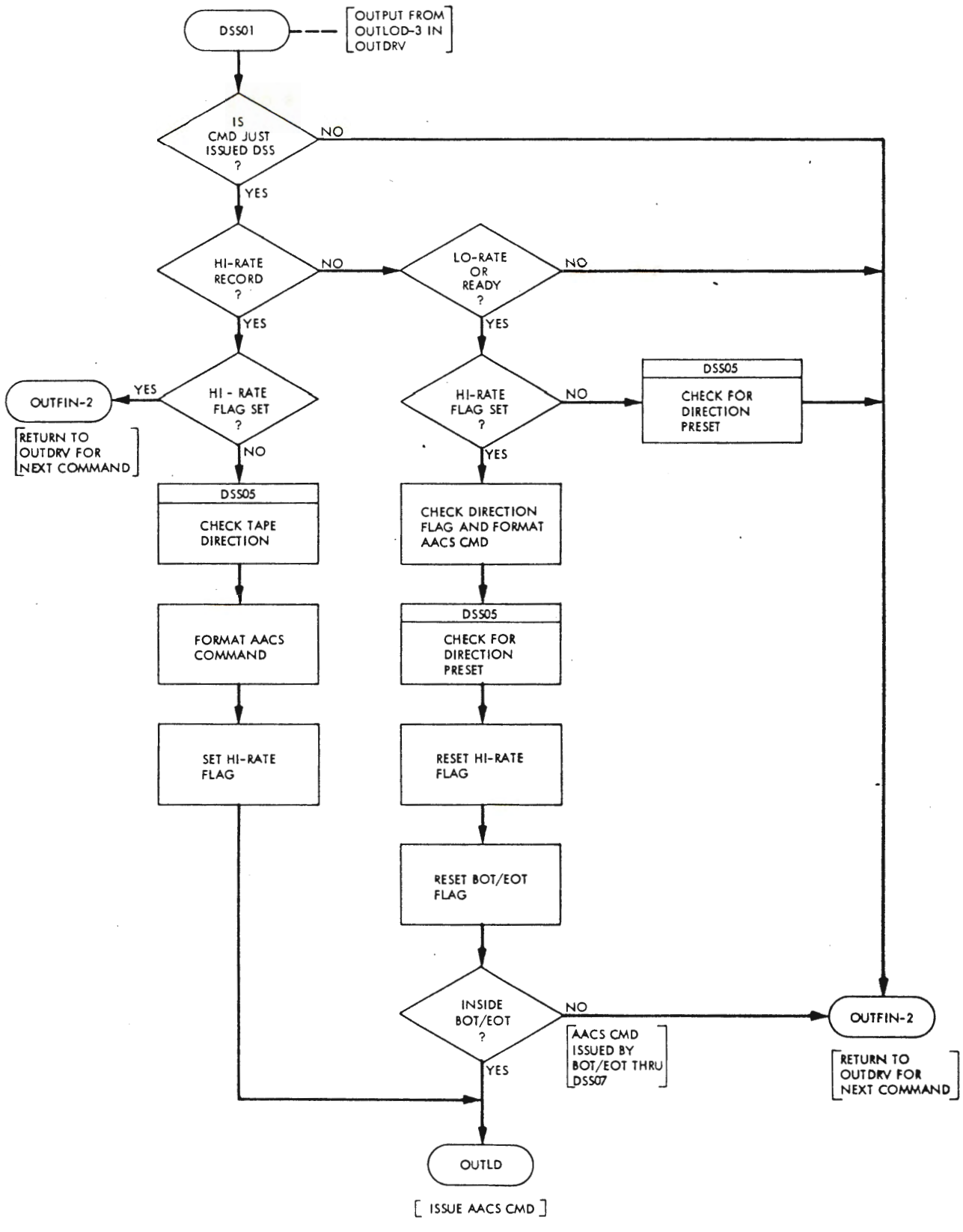


Figure A-6. Flow Diagram of the Optional Routine DSSCAN (Sheet 1 of 2)

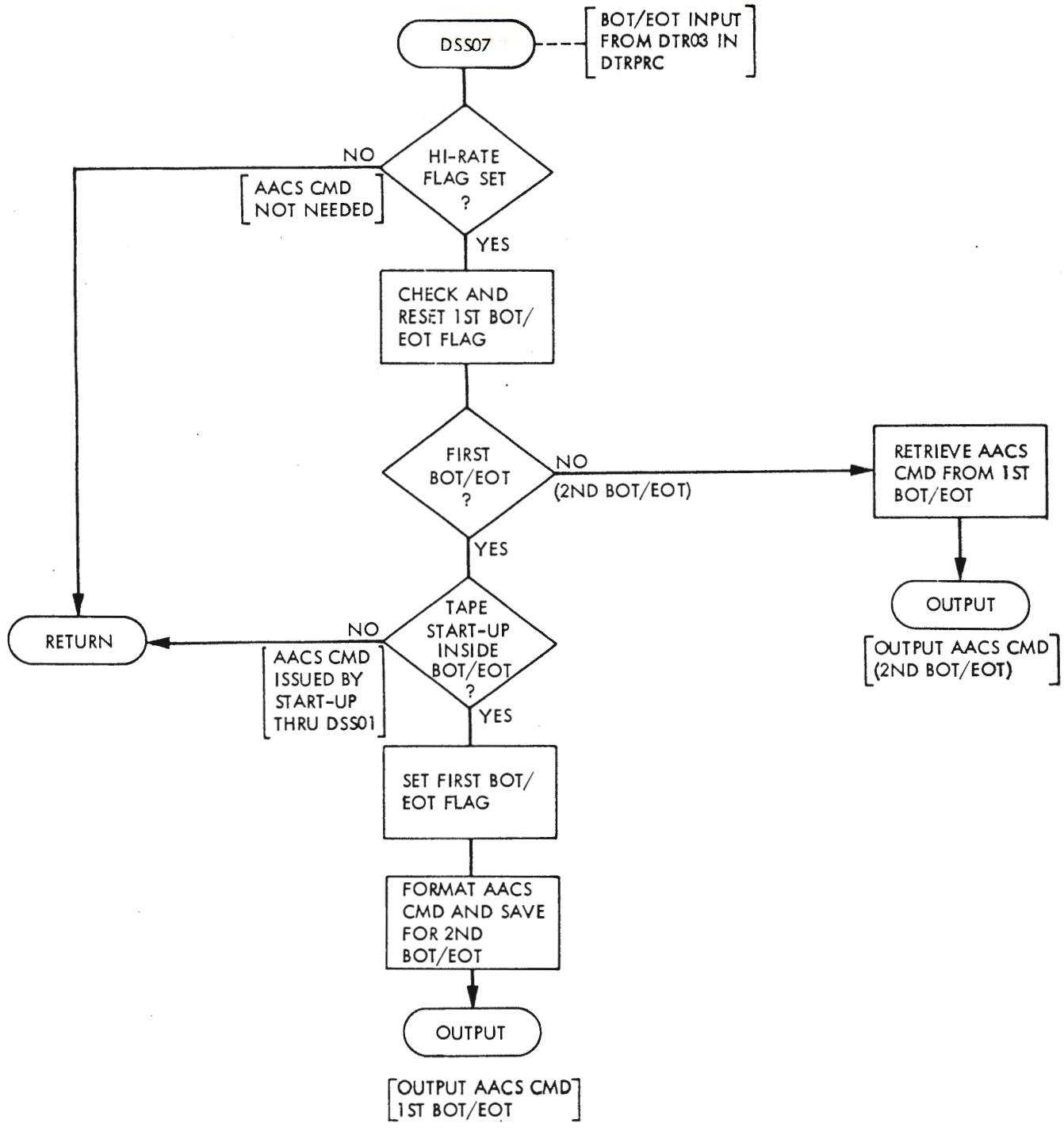


Figure A-6. Flow Diagram of the Optional Routine DSSCAN (Sheet 2 of 2)

A-6. HOUR RESYNCHRONIZATION ROUTINE (HRSYNC)

This is an optional routine that may be used to resynchronize the CCS timing circuitry to the FDS ISS frame-start signal. This routine is intended for use where a spacecraft environment exists that may cause FDS power-on-reset conditions or extra counts in the CCS timing circuitry.

The routine consists of two parts: (1) software to check for a CCS hours pulse every FDS frame, and (2) a time/event table which counts 75 FDS frames and checks an early-hour indicator to determine if the CCS timing circuitry should be reset or not.

Part 1 may be loaded into any convenient 9 contiguous memory locations in read/write memory. After loading and verification, location FABXEC+5 must be changed to TRA HR SYNC. This change to write-protect memory will cause an early-hour indicator to be set each hour at the ISS frame time immediately following a CCS hours pulse.

Part 2 may also be loaded into any convenient 10 contiguous memory locations in read/write memory. After loading and verification, this hourly cyclic may be started by sending the following command:

TRA 2600 DATA (HRCYC) 32

This command should be sent when the CCS is properly synchronized to the FDS and should be executed by the CCS between a CCS hours pulse and the next ISS frame time (an interval of 48 seconds). This hourly cyclic will check the early-hour indicator every 75 frames at the ISS frame that should immediately precede a CCS hours pulse. If the CCS hours pulse occurs before this frame then the early-hour indicator will be set just prior to the hourly check. This will result in a CCS clock reset instruction being executed that will resynchronize the CCS timing chain to the FDS. It must be noted that any already active time/event table in seconds resolution will not be corrected by this clock reset command. However, any new time event table started by a CCS hours-pulse

after the clock reset should be synchronous with the FDS. If the CCS hours-pulse occurs after the proper FDS frame start, as it should, then the early-hour indicator will be set one frame after the hourly check and reset one frame after that without causing a clock-reset to occur.

The flow diagram for this routine is shown in Figure A-7.

A-7. IN-SEQUENCE FRAME RESYNCING (RESYNC)

This is an optional routine that may be used to partially resynchronize an active CCS seconds-resolution time/event table. This routine is intended for use where a spacecraft environment exists that may cause FDS power-on-reset conditions or extra counts in the CCS timing circuitry.

This routine requires that a special resynchronization call be included one or more times in a time/event table. Upon execution of the resync call in a time/event table, this routine will delay further processing of that time-event table until the next FDS ISS frame-start signal. The intent here is to resynchronize the rest of the time/event table to FDS frame-starts in the event that a timing error occurs while the time-event table is active (before the resync execution). It must be noted that this resyncing operation will only remove timing errors of one second or greater. Errors of less than one second will not be affected. Use of the HRSYNC routine can remove the less-than-one-second error (once per hour). Also, this resyncing function will only affect the time/event table that initiates the resync. Any other seconds or hours resolution tables will not be affected.

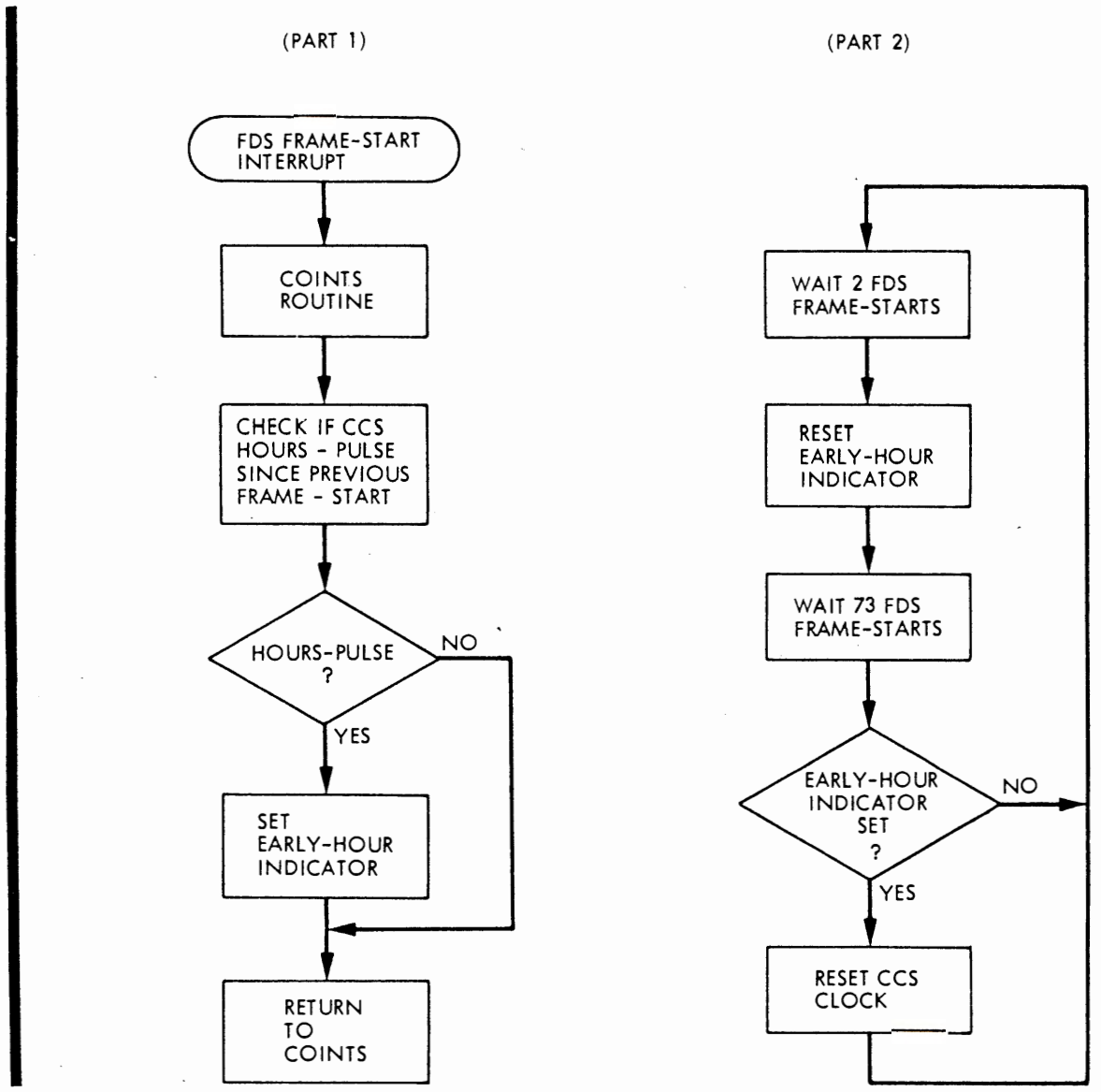
This 9-word routine may be loaded into any convenient contiguous 9 words of read/write memory. The following special resync call in a time/event table is required each time the sequence resync function is desired.

TIME
(RESYNC) 34

The flow diagram for this routine is shown in Figure A-8.

Change #2
22 Jan 1980

94



Change #2
22 Jan 1980

Figure A-7. Flow Diagram of the Optional Routine HRSYNC

IC<

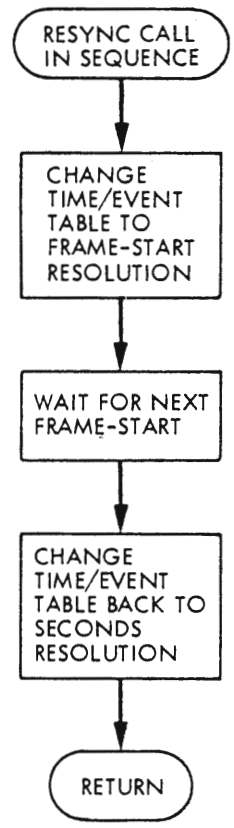


Figure A-8. Flow Diagram of the Optional Routine RESYNC

10b

A-8 LOW GAIN ANTENNA OPTION SEQUENCE

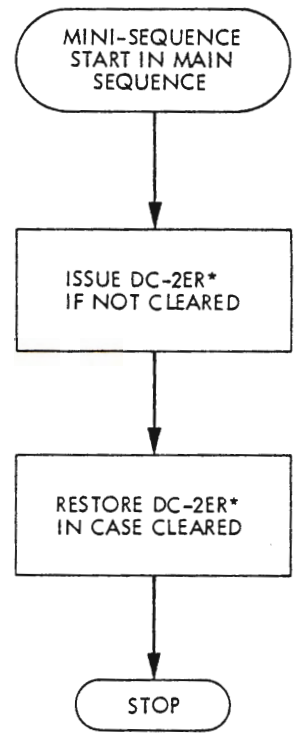
This is an optional mini-sequence that may be used to provide a switch to the low-gain antenna if a disable command is not sent before execution time. This mini-sequence typically would be used after maneuvers as a contingency in the event the maneuver did not return to earth-point.

This mini-sequence would typically be loaded into the flight-reserve memory area of CCS. Providing a fixed location for this sequence allows the disable command to be the same for each use as long as the mini-sequence remains in the same place. The sequence requires six contiguous memory locations within the CCS.

The mini-sequence is executed by a table start command from a primary sequence (maneuver). A DC-2ER command is immediately issued unless this location has previously been cleared by a ground command. The sequence then restores the DC-2ER command in case it was cleared so that the mini-sequence is enabled for its next use, whenever that may be.

During spacecraft 32 maneuvers the DC-2ER command is replaced with a transfer command to a maneuver recovery block that is to be executed if the ground command to disable it is not received and the CA/$\overline{\text{CA}}$ sequence is enabled or a loss of references has occurred (ACCW13 $\neq$ 0).

The flow diagram for this sequence is shown in Figure A-9.



*CHANGED TO A TRANSFER-TO-MANEUVER-RECOVERY-BLOCK COMMAND DURING S/C 32 SATURN ENCOUNTER

Figure A-9. Flow Diagram of the Optional Mini-Sequence for the Low-Gain Antenna Contingency

Change #4
16 July 1981

198

A-9 IRIS FLASH-OFF HEATER CONTINGENCY (S/C 32 ONLY)

This is an optional mini-sequence that may be used to turn on the IRIS flashoff heater if the receiver should fail on S/C 32. This mini-sequence is executed at the end of each cruise sequence after the load window for the next sequence. The 'next' load typically contains a command to clear a flashoff-heater-on enable within this mini-sequence. Thus, as long as command capability to uplink a new load exists, the flashoff heater mini-sequence will be disabled. If command capability is lost, then the mini-sequence will not be disabled and the IRIS flashoff heater will be turned on.

The mini-sequence is executed by a table start command from a primary sequence (cruise). The IRIS replacement heater is turned off immediately, unless a command was received to disable the sequence. Also, if not disabled, the IRIS flashoff heater is turned on one minute after the replacement heater is turned off. The mini-sequence then re-enables itself for its next use.

The flow diagram for this sequence is shown in Figure A-10.

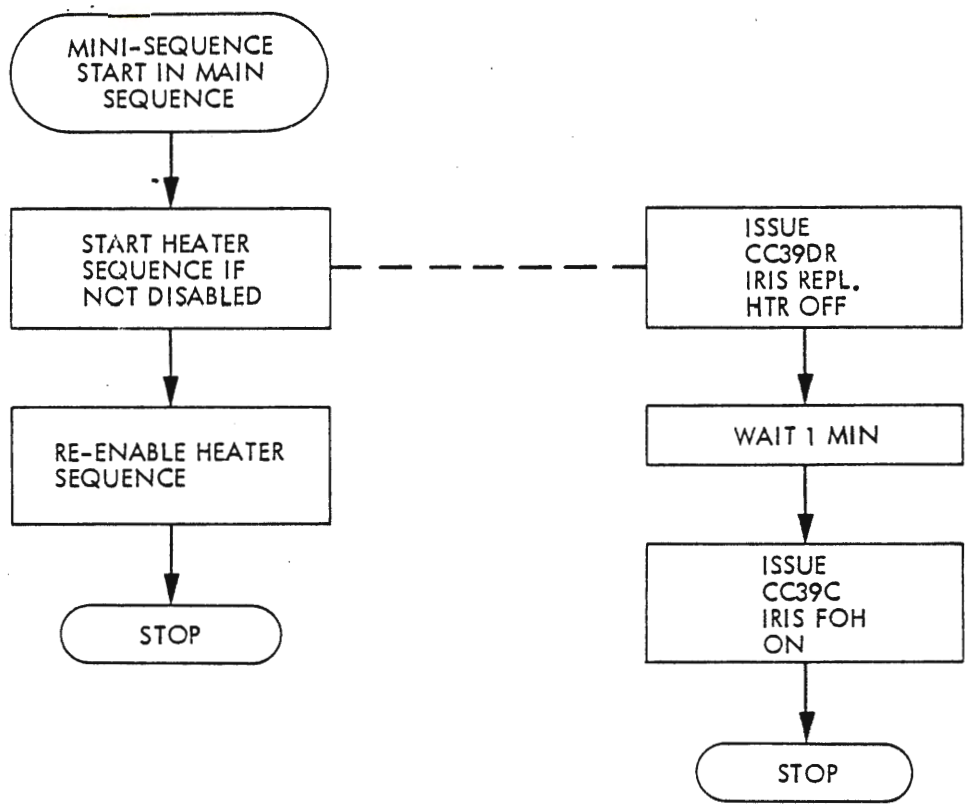


Figure A-10. Flow Diagram of the Optional Mini-Sequence for IRIS FOH On.

Change #3
9/8/80

A10 JSU MANEUVER RECOVERY BLOCK (S/C 32 only)

This is an optional mini-sequence that may be used at the end of a maneuver sequence to enhance spacecraft recovery after an anomalous maneuver. The function of this routine is to execute a -360° roll turn while transmitting on the high-gain antenna. This presumes that the anomaly is associated with the wrong roll reference and that normal sun acquisition is in effect. The roll turn on the high-gain antenna should produce a ground observed telemetry signal for a short period of time as the antenna sweeps past earth-point. This point can be noted by Voyager operation personnel and an appropriate commanded turn can be calculated for subsequent transmission to the spacecraft to realign the high-gain antenna with earth.

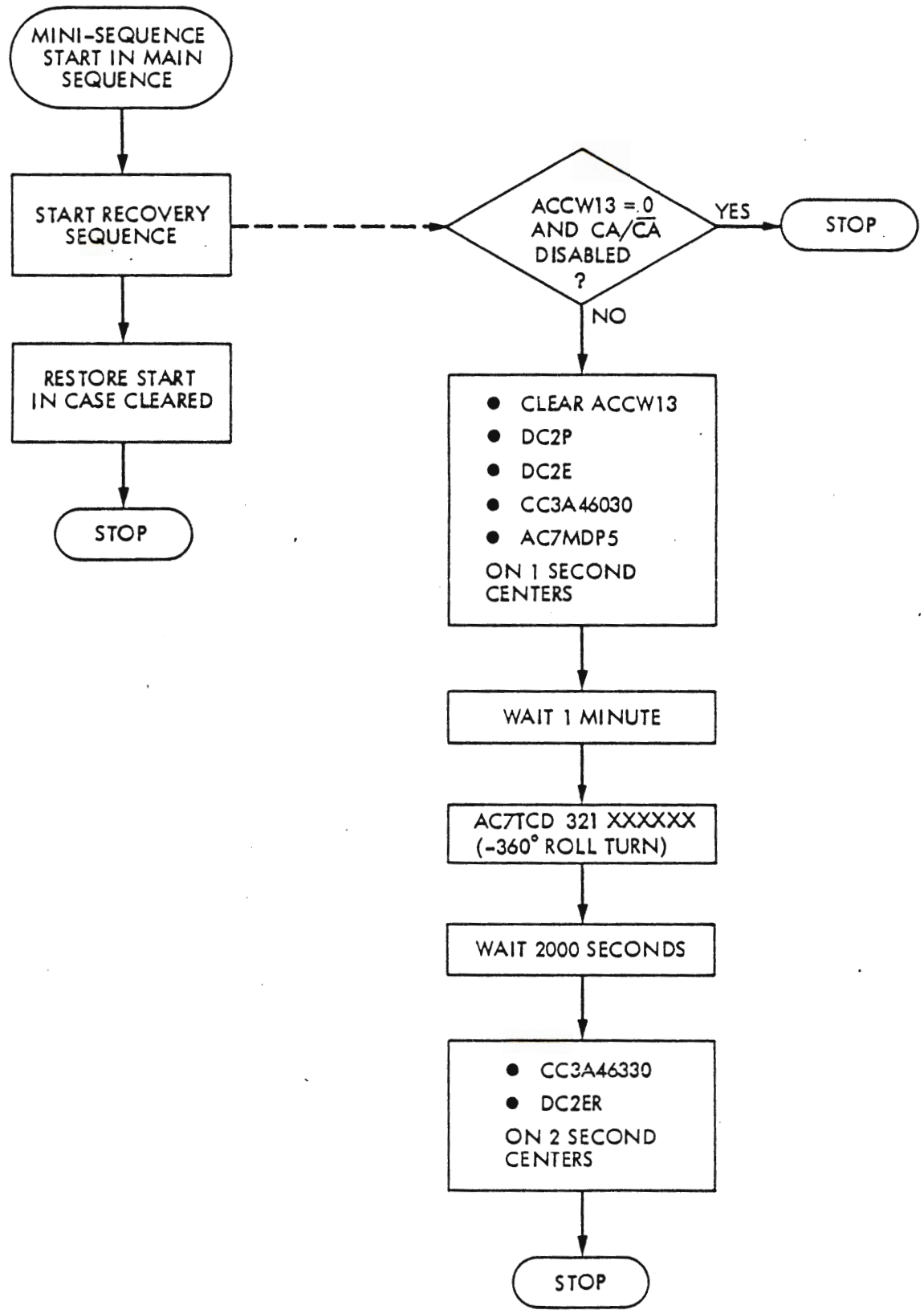
In addition to a roll inertial command and the -360° roll turn, this sequence will also clear ACCW13 and configure the downlink to transmit on the high-gain antenna in the two-way non-coherent mode. At the end of the -360° roll, the low-gain antenna will be selected.

It is intended that this routine be utilized in a manner that allows ground command disabling similar to the low-gain antenna option sequence (see A-8). In fact, the low-gain antenna option code can be used for this function by replacing the 2ER command with a command to start the maneuver recovery sequence.

Actual execution of this sequence is further constrained by a check on ACCW13 and the CA/$\overline{\text{CA}}$ enable locations in the CCS. If ACCW13 is not set, and the CA/$\overline{\text{CA}}$ sequence is disabled, implying that a normal acquisition has occurred, the recovery sequence will not be executed, even though a ground command was not received by the spacecraft.

The flow diagram for this sequence is shown in Figure A-11.

Change #6
7/26/82



Change #6
7/26/82

Figure A-11. Flow Diagram of the Optional Maneuver Recovery Block.