

General Disclaimer

One or more of the Following Statements may affect this Document

- This document has been reproduced from the best copy furnished by the organizational source. It is being released in the interest of making available as much information as possible.
- This document may contain data, which exceeds the sheet parameters. It was furnished in this condition by the organizational source and is the best copy available.
- This document may contain tone-on-tone or color graphs, charts and/or pictures, which have been reproduced in black and white.
- This document is paginated as submitted by the original source.
- Portions of this document are not fully legible due to the historical nature of some of the material. However, it is the best reproduction available from the original submission.

(NASA-TM-87385) AUTONOMOUS ATTITUDE
DETERMINATION SYSTEM (AADS). VOLUME 1:
SYSTEM DESCRIPTION (NASA) 356 P
HC A16/MF A01

N85-11559

CSSL 09B

Unclas
G3/61 01193

AUTONOMOUS ATTITUDE DETERMINATION SYSTEM (AADS) VOLUME 1: SYSTEM DESCRIPTION

APRIL 1982



National Aeronautics and
Space Administration

Goddard Space Flight Center
Greenbelt, Maryland 20771

**AUTONOMOUS ATTITUDE
DETERMINATION SYSTEM (AADS)
VOLUME 1: SYSTEM DESCRIPTION**

APRIL 1982



National Aeronautics and
Space Administration

Goddard Space Flight Center
Greenbelt, Maryland 20771

FOREWORD

The Systems Technology Laboratory (STL) is a computational research facility located at the Goddard Space Flight Center of the National Aeronautics and Space Administration (NASA/GSFC). The STL was established in 1978 to conduct research in the area of flight dynamics systems development. The laboratory consists of a VAX-11/780 and a PDP-11/70 computer system, along with an image-processing device and some microprocessors. The operation of the Laboratory is managed by NASA/GSFC (Systems Development and Analysis Branch) and is supported by SYSTEX, Inc., Computer Sciences Corporation, and General Software Corporation.

The main goal of the STL is to investigate all aspects of systems development of flight dynamics systems (software, firmware, and hardware), with the intent of achieving system reliability while reducing total system costs. The flight dynamics systems include the following: (1) attitude determination and control, (2) orbit determination and control, (3) mission analysis, (4) software engineering, and (5) systems engineering. The activities, findings, and recommendations of the STL are recorded in the Systems Technology Laboratory Series, a continuing series of reports that includes this document. A version of this document was also issued as Computer Sciences Corporation document CSC/SD-82/6032V1.

The primary contributors to this document include

Kishor Saralkar	(Computer Sciences Corporation)
Yury Frenkel	(Computer Sciences Corporation)
Gerald Klitsch	(Computer Sciences Corporation)
Kuen-San Liu	(Computer Sciences Corporation)
Eugene Lefferts	(Goddard Space Flight Center)

Other contributors include

Keiji Tasaki	(Goddard Space Flight Center)
Frank Snow	(Goddard Space Flight Center)
James Garrahan	(Computer Sciences Corporation)

Single copies of this document can be obtained by writing to

Keiji Tasaki
Code 582.1
NASA/GSFC
Greenbelt, Maryland 20771

ABSTRACT

This document provides the information necessary to understand the Autonomous Attitude Determination System (AADS). It describes AADS requirements, program structure, mathematical algorithms used, and gives information for AADS execution and system generation. The document also describes all interfaces with the AADS simulator and other information necessary for system maintenance. A second volume of this document, containing the AADS module descriptions, will be produced in September 1982 under the document number CSC/SD-82/6032V2.

PRECEDING PAGE BLANK NOT FILMED

TABLE OF CONTENTS

VOLUME 1: SYSTEM DESCRIPTION

<u>Section 1 - Overview</u>	1-1
1.1 Introduction.	1-1
1.1.1 Purpose.	1-1
1.1.2 Document Organization and Use.	1-1
1.2 System Requirements and Implementation.	1-5
1.2.1 Main Requirements.	1-5
1.2.2 Changed and New Requirements	1-6
1.2.3 Implementation Scheme.	1-6
1.3 System Description.	1-7
1.3.1 AADS Simulator	1-7
1.3.1.1 Ground Support Simulator	1-10
1.3.1.2 Sensor Data Simulator.	1-10
1.3.1.3 Onboard Support Simulator.	1-11
1.3.2 AADS	1-11
1.3.2.1 Executive.	1-11
1.3.2.2 Input.	1-14
1.3.2.3 State Propagation.	1-14
1.3.2.4 State Update	1-14
1.3.2.5 Output	1-15
1.4 Design Status	1-16
1.4.1 How the Requirements Are Met	1-16
1.4.2 Changes in the Previous Design	1-16
1.4.3 Areas of Concern	1-18
1.5 Design Critique	1-19
1.5.1 Design Rationale and Alternate Designs	1-19
1.5.2 Alternate Designs.	1-23
1.5.3 Suggestions for Improvement.	1-26
1.6 AADS Operation Information.	1-28
1.6.1 Operation on the VAX Computer.	1-28
1.6.2 Operation on the Intel-VAX Computers	1-29
1.6.3 Changes to the System During Flight Qualification.	1-30

TABLE OF CONTENTS (Cont'd)

Section 1 (Cont'd)

1.6.4	AADS Startup Scenario and Flow Diagram . .	1-31
1.6.5	Uplink of Control Parameters	1-33
1.6.6	Input Processing	1-34

Section 2 - Specifications Summary

2.1	Main Requirements	2-1
2.2	Design Constraints.	2-2
2.2.1	Hardware Requirements.	2-3
2.2.2	Operating System Requirements.	2-4
2.2.3	FORTRAN Compiler Requirements.	2-9
2.2.4	AADS Simulator Requirements.	2-9
2.3	AADS Executive Requirements	2-11
2.4	Input Data Processing Requirements.	2-13
2.4.1	Message Format for Uplink and Downlink . .	2-15
2.5	Attitude Data Processing Requirements	2-16
2.5.1	Gyro Data Processing Requirements.	2-17
2.5.2	State Propagation Requirements	2-17
2.5.3	Star Tracker Data Processing Requirements	2-18
2.5.4	Star Identification Requirements	2-19
2.5.5	State Update Requirements.	2-20
2.6	Output Data Processig Requirements.	2-20
2.7	Special Considerations.	2-22
2.7.1	Input Processing	2-22
2.7.2	AADS Startup Scenario.	2-24
2.7.3	Star Camera Occultation.	2-26
2.7.4	Attitude Maneuver.	2-26
2.7.5	Contingencies and Error Handling	2-27
2.7.5.1	Minor Errors	2-27
2.7.5.2	Severe Errors.	2-28

Section 3 - AADS Baseline Diagrams

3.1	Executive Process	3-5
3.1.1	Input/Output	3-5

TABLE OF CONTENTS (Cont'd)

Section 3 (Cont'd)

	3.1.1.1	Input	3-5
	3.1.1.2	Output	3-5
3.1.2		Processing	3-7
3.2		Sensor Data Collection Process	3-14
3.2.1		Input/Output	3-14
	3.2.1.1	Input	3-14
	3.2.1.2	Output	3-14
3.2.2		Processing	3-16
3.3		Input Process	3-20
3.3.1		Input/Output	3-20
	3.3.1.1	Input	3-20
	3.3.1.2	Output	3-20
3.3.2		Processing	3-20
3.4		Output Process	3-26
3.4.1		Input/Output	3-26
	3.4.1.1	Input	3-26
	3.4.1.2	Output	3-28
3.4.2		Processing	3-29
3.5		Gyro Data Process	3-33
3.5.1		Input/Output	3-33
	3.5.1.1	Input	3-33
	3.5.1.2	Output	3-35
3.5.2		Processing	3-35
3.6		State Propagation Process	3-40
3.6.1		Input/Output	3-40

TABLE OF CONTENTS (Cont'd)

Section 3 (Cont'd)

3.6.1.1	Input.	3-40
3.6.1.2	Output	3-40
3.6.2	Processing	3-42
3.7	Star Data Process	3-47
3.7.1	Input/Output	3-47
3.7.1.1	Input.	3-47
3.7.1.2	Output	3-49
3.7.2	Processing	3-49
3.8	Star Identification Process	3-55
3.8.1	Input/Output	3-55
3.8.1.1	Input.	3-55
3.8.1.2	Output	3-55
3.8.2	Processing	3-57
3.9	State Update Process.	3-61
3.9.1	Input/Output	3-61
3.9.1.1	Input.	3-61
3.9.1.2	Output	3-61
3.9.2	Processing	3-61
3.10	AADS Star Catalog Generation Utility.	3-67
3.10.1	Input/Output	3-67
3.10.1.1	Input.	3-67
3.10.1.2	Output	3-69
3.10.2	Processing	3-69
<u>Section 4 - Requirements for Execution</u>		<u>4-1</u>
4.1	General Information	4-1
4.2	Resources	4-2

TABLE OF CONTENTS (Cont'd)

Section 4 (Cont'd)

4.2.1	File Structure on the VAX.	4-8
4.2.2	Data Set Definitions	4-10
4.2.3	Memory Requirements.	4-10
4.2.4	Timing Study	4-13
4.3	Execution and System Generation	4-15
4.3.1	Command Files.	4-15
4.3.2	System Execution	4-15
4.3.2.1	VAX System Configuration	4-16
4.3.2.2	VAX-Intel Configuration.	4-16
4.3.3	System Generation.	4-20
4.3.4	Execution and Creation of Utility Program.	4-26
4.4	AADS Control Parameters	4-33
4.4.1	Schedules and Priorities	4-33
4.4.2	Initial State and Time, and Gyro Process- ing Parameters	4-35
4.4.3	Star Data Processing and Star Identifica- tion Parameters.	4-37
4.4.4	Occultation Prediction Parameters.	4-42
4.5	AADS Error Messages	4-42
4.5.1	System Executive Messages.	4-43
4.5.2	Input/Output Messages.	4-45
4.5.3	Gyro Data Processing Messages.	4-48
4.5.4	Star Data Processing Messages.	4-50
4.6	AADS Message Formats.	4-51
4.6.1	General Header (Uplink/Downlink) Format.	4-52
4.6.2	Commands and Uplink Data	4-53
4.6.3	Spacecraft Ephemeris From OSS to AADS.	4-54
4.6.4	Ephemeris Request From AADS to OBC(OSS).	4-57
4.6.5	Data Annotation Report	4-57
4.6.6	Update Report.	4-58
4.6.7	Raw Data Report.	4-60
4.6.8	Activity Log Report.	4-62
4.6.9	AADS Error Messages.	4-63
4.6.10	Sensor Data Formats.	4-64

TABLE OF CONTENTS (Cont'd)

<u>Section 5 - Basic Algorithms</u>	5-1
5.1 Propagation	5-1
5.1.1 Gyro Models	5-1
5.1.2 The State Equation	5-2
5.2 State Propagation	5-3
5.3 State-Error Equations	5-6
5.4 Covariance Propagation	5-7
5.5 Update	5-17
<u>Appendix A - Timing and Memory Estimates</u>	A-1
A.1 Timing and Memory Estimates	A-1
A.1.1 AADS Memory Requirements Study	A-2
A.1.2 AADS Timing Requirements Study	A-4
<u>Appendix B - Questions and Answers</u>	
<u>Appendix C - Weekly Design Review Documentation</u>	
<u>References</u>	
<u>VOLUME 2: MODULE DESCRIPTIONS</u>	

LIST OF ILLUSTRATIONS

Figure

1-1	AADS System Testing Configuration	1-8
1-2	AADS/AADS Simulator Data Flow Diagram	1-9
1-3	AADS External Interfaces.	1-12
1-4	AADS Internal Interfaces.	1-13
1-5	Normal Processing: AADS Initialized.	1-32
1-6	Sample of AADS Options.	1-36
2-1	Computer System Configuration and Information Transfer Rates.	2-5
2-2	VAX-Intel System Configuration.	2-6
2-3	Normal Processing; AADS Initialized	2-23
3-1	Flowcharting Symbols.	3-2
3-2	AADS External Interfaces.	3-3
3-3	AADS Internal Interfaces.	3-4
3-4	EXEC Process External Interfaces.	3-6
3-5	EXEC Process Data Flow.	3-11
3-6	EXEC Process Baseline Diagram	3-12
3-7	SENSIN Process External Interfaces.	3-15
3-8	SENSIN Process Data Flow.	3-18
3-9	SENSIN Process Baseline Diagram	3-19
3-10	INPUT Process External Interfaces	3-21
3-11	INPUT Process Data Flow	3-24
3-12	INPUT Process Baseline Diagram.	3-25
3-13	OUTPUT Process External Interfaces.	3-27
3-14	OUTPUT Process Data Flow.	3-31
3-15	OUTPUT Process Baseline Diagram	3-32
3-16	GYPROC Process External Interfaces.	3-34
3-17	GYPROC Process Data Flow.	3-38
3-18	GYPROC Process Baseline Diagram	3-39
3-19	PROPAG Process External Interfaces.	3-41
3-20	PROPAG Process Data Flow.	3-45
3-21	PROPAG Process Baseline Diagram	3-46
3-22	STRACK Process External Interfaces.	3-48
3-23	STRACK Process Data Flow.	3-53
3-24	STRACK Process Baseline Diagram	3-54
3-25	STARID Process External Interfaces.	3-56
3-26	STARID Process Data Flow.	3-59
3-27	STARID Process Baseline Diagram	3-60
3-28	UPDATE Process External Interfaces.	3-62
3-29	UPDATE Process Data Flow.	3-65
3-30	UPDATE Process Baseline Diagram	3-66
3-31	CATGEN Utility External Interfaces.	3-68
3-32	CATGEN Utility Data Flow.	3-72
3-33	CATGEN Utility Baseline Diagram	3-73
4-1	DCL Procedure for AADS Simulation Run	4-17
4-2	Compilation of Input/Output Functions File [AADS.IO2]COMP.COM.	4-21

LIST OF ILLUSTRATIONS (Cont'd)

Figure

4-3	Compile Input File [AADS.IO2]SENSIN.COM	4-22
4-4	Compile Input File [AADS.IO2]DCAPIN.COM	4-22
4-5	Compile Input File [AADS.IO2]OUTPUT.COM	4-22
4-6	Link Command File [AADS.IO2]LINK.COM.	4-23
4-7	Link-Editor Input File [AADS.IO2]SENSIN1.COM.	4-24
4-8	Link-Editor Input File [AADS.IO2]DCAPIN1.COM.	4-24
4-9	Link-Editor Input File [AADS.IO2]OUTPUT1.COM.	4-24
4-10	Link-Editor Input Options File [AADS.CM2] COMMON.OPT.	4-25
4-11	SENSIN Link-Editor Input Options File [AADS.IO2]COMMON.OPT.	4-27
4-12	Compile Command File [AADS.EX2]COMP.COM	4-27
4-13	Compile Input File [AADS.EX2]EXEC.COM	4-28
4-14	Link Command File [AADS.EX2]LINK.COM.	4-28
4-15	Link Input File [AADS.EX2]EXEC1.COM	4-29
4-16	Link Input Options File [AADS.EX2]COMMON.OPT.	4-29
4-17	VAX-Intel System Configuration Used for System Generation of Intel Executable Image of AADS.	4-30
4-18	DCL Procedure To Link and To Execute TMPCOMCRE Utility Program on VAX.	4-31

LIST OF TABLES

Table

1-1	Scheduling Activities of AADS	1-17
3-1	EXEC Process Component Descriptions	3-9
3-2	SENSIN Process Component Descriptions	3-17
3-3	INPUT Process Component Descriptions.	3-23
3-4	OUTPUT Process Component Descriptions	3-30
3-5	GYPROC Process Component Descriptions	3-37
3-6	PROPAG Process Component Descriptions	3-44
3-7	STRACK Process Component Descriptions	3-52
3-8	STARID Process Component Description.	3-58
3-9	UPDATE Process Component Descriptions	3-64
3-10	CATGEN Utility Component Descriptions	3-71
4-1	Data Set MATCAT.DAT Contains Stars Selected From the SKYMAP Star Catalog, Version 3.1	4-11
4-2	Format of the Sorted Data Set SORTED.SAV.	4-12
4-3	Memory Requirement Summary.	4-14

LIST OF TABLES (Cont'd)

Table

A-1	Memory Requirements on the Intel 8086/8087. . .	A-3
A-2	Comparison of Instruction Timings on the NSSC-1 and Intel 8086/8087 Computers.	A-5
A-3	Frequency of Use of Arithmetic Instructions in Applications Software Cycle.	A-7
A-4	Timing Estimates for AADS Processes Implemented on the Intel 8086/8087.	A-8

SECTION 1 - OVERVIEW

1.1 INTRODUCTION

1.1.1 PURPOSE

Autonomous attitude determination onboard a spacecraft for the purpose of automatic science data annotation is one of the key objectives of the National Aeronautics and Space Administration (NASA) End-to-End Data System (NEEDS). The Autonomous Attitude Determination System (AADS) described here is designed to use gyro and star tracker data to compute the spacecraft attitude. The system will perform autonomously for long periods of time with only minimal ground support.

To demonstrate the feasibility on the ground of such a system, the attitude determination system must be developed along with a simulator that will provide all necessary input and will receive the computed attitude and associated system performance measures. AADS and the AADS simulator could then be used to search for an appropriate set of initial conditions for each type of mission (or for a variety of missions and mission scenarios).

A detailed AADS system description is presented in this document. A preliminary version of the document was prepared earlier (Reference 1). That version has been updated by adding AADS design changes and by giving additional information on system execution and system generation procedures.

1.1.2 DOCUMENT ORGANIZATION AND USE

This system description is designed to help users understand and maintain the system. The document is divided into five major sections. Section 1 provides an overview of the system and can be used to gain both a basic understanding of

the system and the rationale for choosing this particular design to implement the requirements. Section 2 summarizes the system requirements, the design constraints imposed due to the selection of hardware for developing and implementing AADS, and the requirements for the system software of the target microcomputer. Section 3 gives detailed input, output, processing overviews, and the baselines for all component processes that make up AADS. Section 4 contains the requirements for execution, and Section 5 provides the descriptions of basic algorithms used for state propagation and update.

In addition, appendixes provide predevelopment timing and memory estimates prepared by the system designers, and other information that may be used for future enhancement of the system. This document is written with the assumption that readers are familiar with the basic concepts of real-time systems and multiprocessing event-driven operating systems.

AADS interfaces with the user by means of the Ground Support Simulator (GSS), which is a part of the AADS simulator (References 2 and 3). The user selects commands and AADS processing options that are displayed by GSS. GSS uplinks the information to AADS and reads AADS output transmitted to the ground. GSS then prepares AADS output for use by the operator and the analyst and displays error messages transmitted by AADS. A description of GSS and the basic operation of AADS will appear in a user's guide to be written by the design team (Reference 4).

The AADS user's guide is written for the operator, whereas this document is intended for four primary users:

1. System developers--This document is designed to assist system developers (who are usually responsible for implementing component processes), in learning processing modes (multiple processes), in supporting testing, and in

training analysts and operators. This document is also useful in designing systems to support similar missions.

2. Launch mission analyst--This document provides help to the launch mission analyst in understanding system requirements, in learning and refining processing modes, in reinforcing understanding of the system to support operator training, and in planning postlaunch operation. The analyst may also use the document as a guide in planning any system enhancements needed because of changes in mission requirements or oversights.

3. Postlaunch mission analyst--The postlaunch mission analyst, responsible for postlaunch analysis and software maintenance, can also use this document to reinforce understanding of the system and as a guide in planning system enhancements to streamline processing or to satisfy a change in mission requirements.

4. Operator--This document is designed to reinforce understanding of the system by the operator who has software responsibility or is interested in software system design and development.

New users should read the user's guide to understand the actual operation of AADS and then use the system description to acquire a deeper understanding of the system.

Additional documentation for the system includes the AADS software functional specifications, attitude textbook, technical memorandums, and the technical manuals describing the VAX-11/780 computer, the Intel 8086 computer, and the system software. A comprehensive list of references is given at the end of this document.

The attitude software functional specifications (References 1, 2, 5, and 6), written by the analysis team, contain the analytical details necessary to understand the

algorithms and mission requirements. The attitude textbook (Reference 7) defines the basic terminology, describes the sensor hardware, and provides references for further reading. Technical memorandums (Reference 8) describe new or modified algorithms and, in most cases, are included in the system description document.

The system description document provides the digital command language (DCL) procedures used for system generation and execution on the VAX computer; however, performing system maintenance may require additional details that will be found in various technical manuals. In particular, the Intel FORTRAN manual (Reference 9) contains information about the source language, and the operating system manual (Reference 10) provides information about the new operating system being developed by Systex to implement AADS on the Intel microcomputer.

General operation-specific information in this document includes (1) processing overviews, (2) data flow diagrams, (3) illustrations of internal and external interfaces, (4) DCL procedures for system execution, (5) descriptions of processing options, and (6) error messages.

Maintenance-specific information includes (1) illustrations of the relationships among system components (baseline diagrams), (2) illustrations of internal and external interfaces, (3) formats for AADS input/output, (4) DCL procedures for system creation, (5) descriptions of processing options, and (6) error messages.

The source code contains descriptions of subroutine and COMMON area components in the form of prolog and program design language (PDL) comments.

1.2 SYSTEM REQUIREMENTS AND IMPLEMENTATION

1.2.1 MAIN REQUIREMENTS

The main requirements for AADS are as follows:

- AADS will compute the spacecraft attitude to annotate science data onboard the spacecraft.
- The AADS design will meet the requirements of NASA's Multimission Modular Spacecraft (MMS) series. It will support types of spacecraft with attitudes that either vary smoothly in time (Landsat-D), or are nearly inertially fixed (Solar Maximum Mission (SMM)).
- AADS will determine spacecraft attitude with an accuracy of 30 arc seconds (3σ) for each axis.
- AADS will perform in a real-time mode and in an autonomous manner for a long period of time and will require only infrequent ground support.
- AADS will accept commands and data from the ground and will transmit attitude results, update results, activity logs, error messages, and raw sensor data to the ground.
- AADS will be implemented as a portable modular unit on a microprocessor onboard the spacecraft. The microprocessor will have 256 kilobytes of memory and no peripheral devices where 1 kilobyte equals 1024 bytes. AADS will be initially implemented on a VAX-11/780 computer to test feasibility.
- AADS will use gyro and star tracker data. Gyro data are used to propagate attitude, and data from identified stars are used to update attitude using a linear optimal filter (Kalman filter) that is stable against roundoff errors.

- AADS will provide for contingencies and error handling.

1.2.2 CHANGED AND NEW REQUIREMENTS

- AADS will use the regular form of the Kalman filter for state update. The Joseph form of the Kalman filter will not be used because it may require more computer time and may not eliminate roundoff errors (see question AT012 in Appendix B).
- AADS will acquire attitude when the initial attitude estimate is in error by up to 2 degrees. In addition, AADS will automatically change the star identification procedure depending on the current attitude uncertainty.

1.2.3 IMPLEMENTATION SCHEME

The implementation scheme is as follows:

1. Complete a detailed design of the input and output functions; use top-level design for the attitude determination function.
2. Code a skeleton (shell) system consisting of input and output functions. Complete the detailed design of the attitude determination function.
3. Implement the skeleton system on the VAX-11/780 computer and demonstrate the interface with the AADS simulator. Demonstrate scheduling logic in the executive. Complete code for the attitude determination function and detailed error handling.
4. Test attitude computation by using real or simulated Landsat-D telemetry data. Check attitude accuracy and star identification. Obtain detailed

timing estimates to determine any potential problems when the software is transferred to the target microprocessor (the Intel 8086).

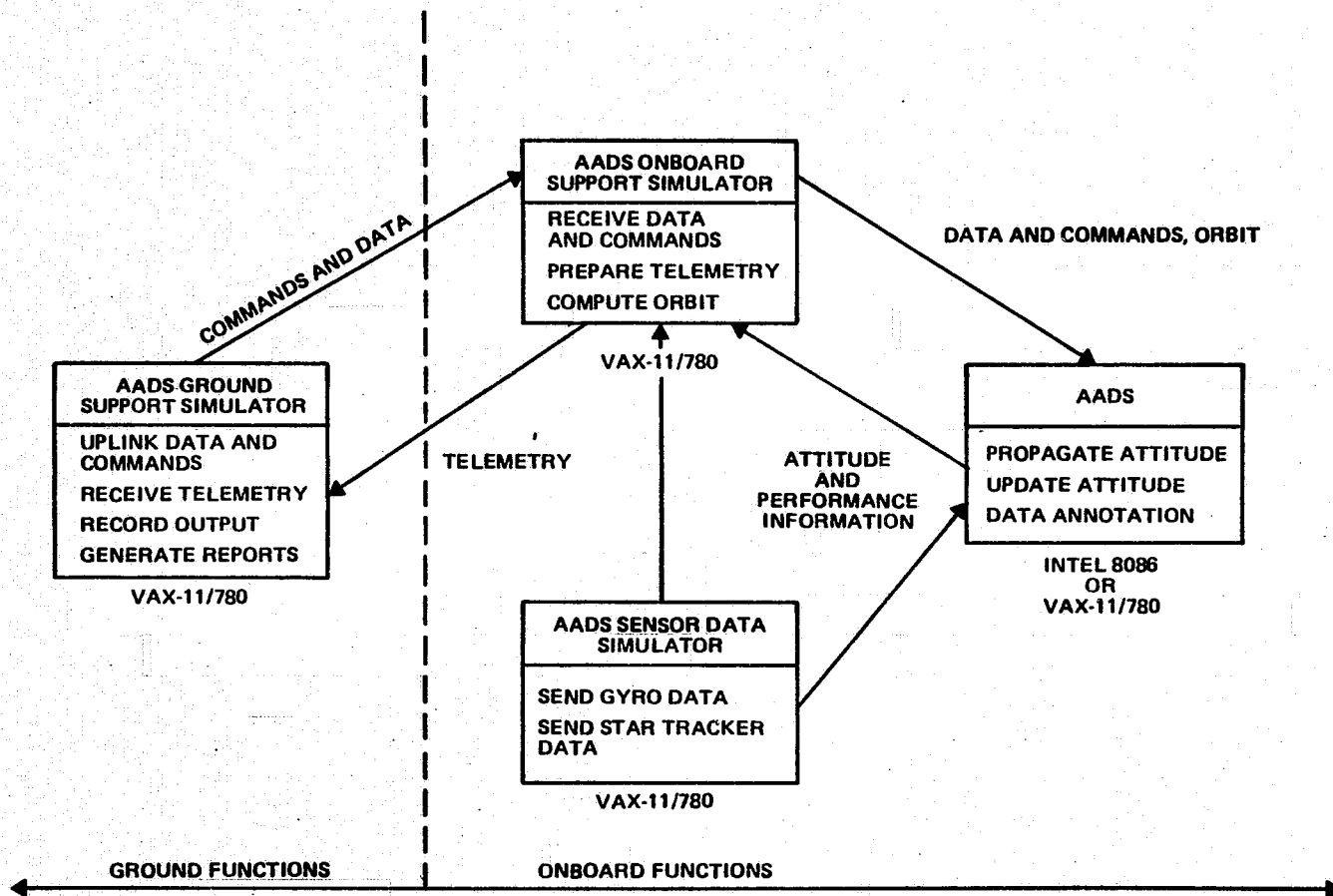
5. Transfer AADS to the Intel 8086 and determine whether the software meets all requirements, including those for execution time. Some interfaces with the operating system will be changed to implement AADS on the Intel 8086; otherwise, the program will be modified only if it cannot meet timing or memory requirements.
6. AADS will be tested in actual flight during an experiment onboard a spacecraft. The Onboard Support Simulator (OSS) and the Sensor Data Simulator (SDS) in the AADS simulator will be replaced by the onboard computer (OBC) and the Remote Interface Unit (RIU), respectively. The sensor data collection (SENSIN) process in AADS will be changed to interface with the RIU.
7. AADS will be used on a satellite for attitude determination and data annotation.

At present, items 1, 2, and 3 have been completed, and work is continuing on item 4. Item 5 will be completed on a time-available basis when Intel hardware and system software are available.

1.3 SYSTEM DESCRIPTION

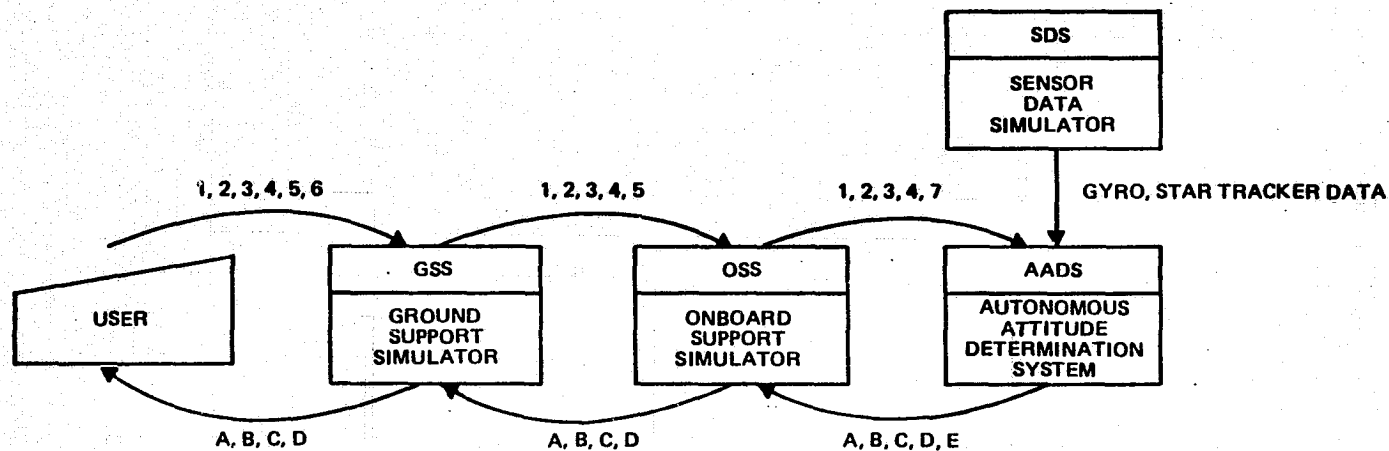
1.3.1 AADS SIMULATOR

The AADS simulator (References 3 and 4) provides commands, processing options, and sensor data during system testing and provides commands and processing options during a mission. Thus, AADS communicates with the operator through the AADS simulator. Figures 1-1 and 1-2 depict the AADS interfaces with the AADS simulator and the related data flow.



28/1858

Figure 1-1. AADS System Testing Configuration



1. PROPAGATION AND UPDATE PARAMETERS
2. STAR DATA PROCESSING AND STAR IDENTIFICATION PARAMETERS
3. AADS COMMANDS (START, STOP, HALT, RESUME, SET CLOCK)
4. OCCULTATION PREDICTION PARAMETERS
5. ORBIT PROPAGATION PARAMETERS
6. SIMULATION OPTIONS
7. S/C EPHEMERIS

- A. ATTITUDE, UPDATE RESULTS, AND STATISTICS
- B. SENSOR DATA REPORT
- C. ACTIVITY LOG
- D. CRITICAL MESSAGES
- E. REQUEST TIME OF S/C EPHEMERIS

8581/82

ORIGINAL PAGE 13
OF POOR QUALITY

Figure 1-2. AADS/AADS Simulator Data Flow Diagram

The AADS simulator consists of several, separate processes. These processes are briefly described below.

1.3.1.1 Ground Support Simulator (GSS)

GSS executes on the VAX to perform three major functions: uplink data preparation, uplink data and commands to AADS, and telemetry reception and output reporting. GSS also sends data and commands to the OSS during the ground testing of AADS.

GSS displays the AADS commands and processing options, and the OSS processing options, for operator input. GSS then formats the commands and data as transmission messages to be uplinked. GSS also receives the output from AADS and prepares formatted output for the operator. GSS displays the activity log during AADS operation and prepares and prints plots and tables of the attitude information downlinked by AADS.

1.3.1.2 Sensor Data Simulator (SDS)

SDS reads gyro and star tracker data from a sequential file created by the Landsat simulator and sends it to AADS. Because the data are available from the sensors onboard the spacecraft, SDS is not needed during a mission. Gyro data are available at 64 milliseconds, and star tracker data, at 50 milliseconds for alternate star trackers; these periods can be varied by option (multiples of the minimum period). In addition, SDS reads header information from the simulation tape. The header (containing such information as initial attitude, biases, and alignment matrices) and the raw sensor data are formatted and printed by the AADS simulator for the analyst. Header information is also made available to GSS for preparing uplink data to AADS.

1.3.1.3 Onboard Support Simulator (OSS)

OSS simulates the OBC during testing and simulation and forms the link between AADS and GSS. OSS accepts commands and data from GSS and then retransmits the AADS information to AADS. AADS sends annotated data and other reports and error messages to OSS for retransmission to the ground. Periodically, AADS requests and receives spacecraft ephemeris from OSS; OSS itself receives the orbit propagation conditions from GSS.

1.3.2 AADS

AADS comprises several coordinated standalone programs (processes) that process sensor data and determine spacecraft attitude in real time. Figures 1-3 and 1-4 are the AADS baseline diagrams. These diagrams are for AADS as a whole and for various processes within the system. Figures 1-3 and 1-4 show the AADS external and internal interfaces, respectively (see Figure 3-1 in Section 3 for definition of flowchart symbols used).

1.3.2.1 Executive (EXEC)

The executive acts as a miniature operating system and controls the various processes (standalone programs) in AADS. It uses a table of process priorities and periods to schedule the processes. All priorities and scheduling periods are variable and are set by command from the ground. The executive sets up a timer request for the "next call" whenever it schedules a new process. It starts a new process when the previous process completes or when a timer request matures. The executive computes the occultation of star cameras, checks for spacecraft maneuver indicators set by other processes, and uses special scheduling during these conditions. Finally, the executive keeps a log of activities performed and errors encountered during scheduling.

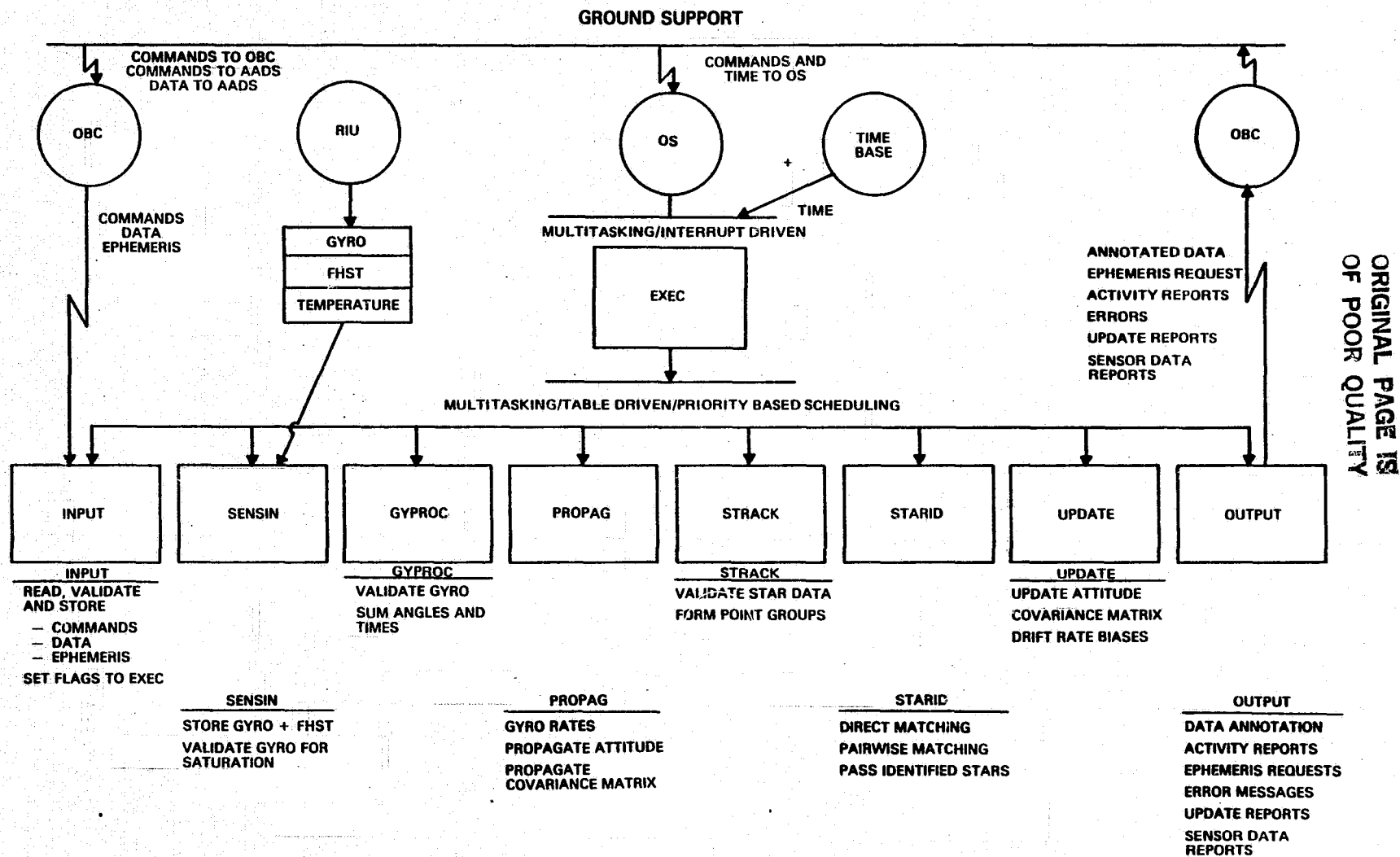


Figure 1-3. AADS External Interfaces

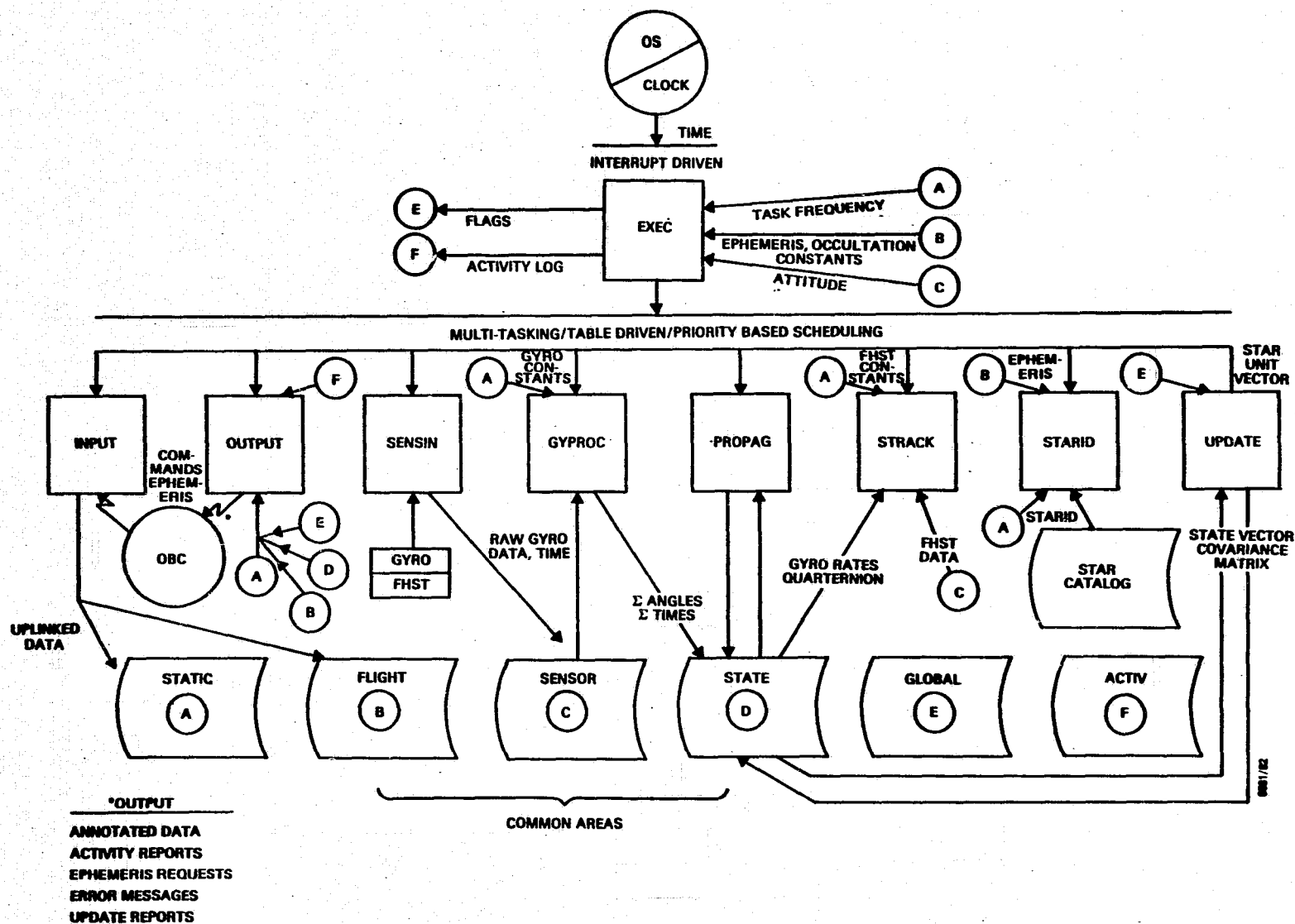


Figure 1-4. AADS Internal Interfaces

1.3.2.2 Input

Two separate processes (SENSIN and INPUT) perform the input function. SENSIN executes at a high priority to access gyro data (every 64 milliseconds) and star tracker data (every 100 milliseconds for both star trackers). SENSIN stores the data in a global COMMON area for use by the attitude computation processes. The INPUT process periodically receives (nominally every 20 minutes) the ephemeris information sent by the OBC(OSS). INPUT also accepts the commands and data sent from the ground by means of the OBC(OSS). The commands are then decoded and given to EXEC for immediate action. EXEC resumes the INPUT process later, when AADS is idle, to validate and store the ephemeris and uplinked processing options.

1.3.2.3 State Propagation

State propagation is performed by two separate processes: GYPROC, which validates and converts gyro data, and PROPAG, which propagates the state and the state covariance matrix. Nominally called every 512 milliseconds, these processes use gyro data accumulated during that period. The scheduling periods for both GYPROC and PROPAG are variable and can be changed by ground command. GYPROC decodes the gyro data to determine whether a maneuver is in progress. It validates gyro counts and converts the counts into angles. PROPAG uses these angles to compute the spacecraft angular velocity, which is then used to propagate the state and the state covariance matrix. The propagated state is used for data annotation.

1.3.2.4 State Update

The state update function is performed by three separate processes: STRACK, STARID, and UPDATE. STRACK collects and validates star tracker data and groups star observations. The update period depends on the availability of star data,

tracker group formation options, and scheduling periods for the STRACK process. STARID matches those star observations with stars from the star catalog; UPDATE updates the state and the state covariance matrix using the observed and predicted (catalog) stars. The state update is nominally performed as soon as at least one star is identified (all times are variable and are set by command from the ground). UPDATE uses observations (unit star vectors) in a Kalman filter to correct the current state for errors that may have accumulated during state propagation using gyro data.

The executive schedules the STRACK process whenever gyro saturation is detected during gyro data validation. This condition indicates a possible loss of attitude; therefore, immediate star identification followed by state update is necessary to determine the correct attitude.

The executive suspends star data processing during spacecraft maneuvers and when both star cameras are occulted. During these periods, state propagation can be performed at a higher frequency to limit attitude error.

1.3.2.5 Output

The OUTPUT process is scheduled by the executive for periodic output of annotated data (after state propagation), performance reports (after state update), and ephemeris requests (nominally every 20 minutes). All periods are variable and are set by ground command. OUTPUT is also invoked when a critical error is detected and sends the activity log, sensor data report, and an error message to the ground.

Annotated data include time-tagged spacecraft attitude data. Performance reports contain information about star data processing and update, and the activity log contains scheduling information, important events, and error indicators. The raw sensor data report contains gyro and star data collected during a short time period just preceding the

error event. The ephemeris request is for a 20-minute span, and sent in a lookahead manner when the ephemeris data that remain in the ephemeris arrays are sufficient for the next update cycle.

1.4 DESIGN STATUS

1.4.1 HOW THE REQUIREMENTS ARE MET

Figures 1-3 and 1-4 and Table 1-1 show that all the major functional requirements are met. The baseline diagrams in Section 3 and the component descriptions (prologs and PDL) in Volume II show that all detailed requirements are satisfied. Attitude accuracy requirements will be met by the use of a Kalman filter.

1.4.2 CHANGES IN THE PREVIOUS DESIGN

The following changes have been made to the previous design:

- Sensor data access using direct-memory access (DMA) in Intel 8086 version of AADS.
- Pairwise star matching technique required when current attitude uncertainty exceeds specified tolerance.
- No triplet star matching technique required.
- No occultation table necessary; occultation computed onboard using Sun/Moon/Earth ephemerides.
- SENSIN process does not check for gyro saturation; check is performed in the GYPROC process. Gyro saturation indicates possible loss of attitude; therefore, executive immediately schedules STRACK, STARID, and UPDATE processes to update state.
- Gyro rates are computed using valid end points in gyro data and not by accumulating individual valid increments.

Table 1-1. Scheduling Activities of AADS

<u>Function</u>	<u>Priority</u>	<u>Period¹ (nominal)</u>	<u>Comments</u>
EXEC	High	NA	Schedules other processes
SENSIN	High	100 msec 64 msec	Collects data from both star trackers; collects gyro data
INPUT	High	-	Receives commands; receives and stores ephemeris data
	High	20 min	
	Low	After a major cycle ²	Updates tables and validates tables and ephemeris data
OUTPUT	Med	512 msec	Outputs annotated data every nth propagation
	High	-	Outputs update report every n updates
	High	Critical error	Outputs activity report, sensor data report every m propagations
	Med High	20 min -	Outputs ephemeris request; outputs error messages
GYPROC	Med	512 msec	Performs gyro data processing
PROPAG	Med	512 msec	(or every nth gyro data processing)
STRACK	Low	2.0 seconds	Nominally, starts immediately after a state update
STARID	Low	-	Scheduled when sufficient star groups formed by STRACK
UPDATE	Low	-	Scheduled when sufficient stars identified by STARID; error message sent when no state update for 7 minutes

¹All periods can be changed by ground command.

²A major cycle includes all activities up to and including a state update.

- The attitude computation function was changed to implement the new and/or changed requirements described in Section 1.2.2. AADS will now automatically adjust the star identification procedure depending on the current attitude uncertainty. For example, larger window sizes and pairwise matching algorithms, will be automatically used when the current attitude uncertainty exceeds a user-specified tolerance. In addition, the regular form of the Kalman filter will be used for attitude determination.

1.4.3 AREAS OF CONCERN

The following areas of concern have been identified. They do not hamper the implementation of a prototype AADS on the VAX computer, but should be considered for the final AADS version on the Intel 8086 computer, and the version of AADS used during a satellite mission.

- Function of the RIU is not understood very well. The AADS simulator does not simulate RIU functions accurately.
- Data collection for downlink telemetry is not understood very well.
- Communications protocol between AADS and the OBC (OSS) is not defined very well.
- Time required to transmit a buffer from the OBC to AADS or vice versa is not known.
- The real-time system clock proposed for AADS may slow down testing if high-priority time is not available on the VAX-11/780. Long simulations will need extensive programmer-present time, a condition that could affect other VAX users adversely.

- Hardware test procedures for RAM, ROM, and CPU should be established to verify system integrity.
- Electrically erasable programmable memory (EPROM) should be considered as an alternative to RAM to ensure system integrity and reliability.

1.5 DESIGN CRITIQUE

1.5.1 DESIGN RATIONALE AND ALTERNATE DESIGNS

The paragraphs below explain the rationale for the AADS design and describe alternate designs.

Design Constraints. All constraints must be satisfied as follows:

- The target microprocessor has no peripherals; 240K byte available memory; 64K-byte process size
- Gyro propagation performed in 512 milliseconds; attitude update in at least 7 minutes
- 30-arc-second (3σ) accuracy for each of the three axes
- Must be autonomous for long periods (that is, the entire star catalog must be stored in memory)

Design Goals (Optimize). Major design goals are as follows:

- Clear, simple, flexible design; major scheduling logic visible from the executive and not hidden inside the individual processes
- Functional integrity
- General purpose (nonspecialized)
- Portable between different computers
- Minimum overhead

The design presented in this document has been examined to see whether it meets the design constraints and optimizes the design goals. Results of the examination are as follows:

1. Completeness. The design meets all functional requirements. The specified accuracy is expected to be achieved by using a Kalman filter. Several components influence the accuracy. These include gyro errors, bias drift, star identification, star measurement, and dynamic errors in quaternion computation. The Kalman filter reduces only the observation residuals.

2. Time. A rough estimate made during the previous design period (References 11, 12, and 13) shows that a single execution of all ADDS attitude determination processes will require approximately 260 milliseconds for the worst-case scenario, leaving nearly 50 percent of the 512 milliseconds processing time period available for input/output and other activities. However, it is possible that AADS cannot meet timing requirements because of unforeseen circumstances. Then, either AADS code must be optimized with respect to time, or the gyro propagation period must be increased. Time optimization, if required, will be performed after AADS is implemented. Analysis of AADS test runs should point out those areas of code whose execution efficiency will significantly lower the execution times. As a rule of thumb, 20 percent of the code uses 80 percent of the execution time. Time optimization can be achieved by redesigning this 20 percent of the code if the system does not perform within the available time. A simple design will permit redesign without a large effort.

Time optimization may not be necessary if the attitude accuracy requirements can be met using a gyro propagation period of 1024 milliseconds. GSFC and CSC personnel are analyzing the attitude propagation algorithm to determine

the maximum gyro propagation period that satisfies accuracy requirements.

The Intel 8086 has been shown to be approximately twice as powerful as the NSSC-1 microcomputer (Reference 13) that was used for attitude determination and control of the SMM spacecraft. In the near future, there will be newer versions of the Intel and other microcomputers that will be more powerful than the Intel 8086. The portability of AADS (use of FORTRAN) will allow it to be implemented on these later version microcomputers without major changes to the design. Thus, time requirements can be also met by using newer microcomputers if necessary.

3. Memory. A study (Reference 11) performed during the previous design period (Task 92300) gave an estimate of approximately 175 kilobytes of memory for AADS component processes. The star catalog was estimated at 142 kilobytes of memory for all SKYMAP stars with star magnitudes between 2.0 and 6.5, a major portion of the available memory; therefore, memory optimization efforts were directed toward compressing the star catalog. In the previous design of the star catalog, some redundant and unnecessary information was stored. Task personnel have redefined the AADS star catalog structure to reduce the space requirement to 55 kilobytes. However, additional logic and computer time are needed to decode the contents of the star catalog. Task personnel have verified through test runs that this additional time does not adversely affect the performance of AADS.

The AADS processes require approximately 107 kilobytes of memory space and 67 kilobytes for the global COMMON areas including the star catalog (see Section 4.2.3 for further information). Thus, the design satisfies the memory constraints when AADS is implemented on the VAX computer. However, it is possible that the size of the AADS executable

images will increase when the Intel 8086 FORTRAN compiler is used to compile the AADS source code. Using a conservative expansion factor of 2.0 between the object code produced by the VAX FORTRAN compiler and the Intel 8086 FORTRAN compiler, the memory requirement would be approximately 320 kilobytes. This means that AADS can be tested on the Intel microcomputer development system (MDS), but cannot be implemented on the target microcomputer (with 240 kilobytes of memory). However, a precise determination of the object code expansion factor is needed before attempting any memory optimization.

4. Overhead. The Intel operating system (under development) will support shared global COMMON areas but will not support a shared mathematical function library. This library is expected to require approximately 2 kilobytes of memory, and each process must be built with a separate copy of the library. Some of this overhead can be reduced by a proper combination of some processes or if reentrant code is supported by the FORTRAN compiler. Again, this combination should be made so that the design retains its flexibility and the main scheduling logic remains in the executive (not hidden in the lower level components). All interprocess communication will be via global COMMON areas; therefore, this factor should not affect the design.

5. Design Goals. The design team feels that the current design satisfies all design goals. It is a simple design that achieves functional integrity by assigning a separate process for each major function. Design flexibility was demonstrated by the fact that a major change in the star identification requirements--use of a more powerful pairwise matching technique--was implemented very easily without appreciable impact on the design schedule. Task members have also determined that simple changes to the logic of the executive main driver and the process schedule

table will allow the addition of a coarse attitude determination system for enhanced versions of AADS. The system is portable through the use of FORTRAN as mentioned earlier. Thus, it can be implemented on newer microcomputers that support a multiprocessing operating system and standard FORTRAN compiler and can satisfy other basic memory and time requirements. Therefore, the present design was selected. However, other designs are shown and compared with the current design.

1.5.2 ALTERNATE DESIGNS

The alternate designs listed below were considered.

1. Design A. Current design.
2. Design B. Single process (load module). All asynchronous actions taken using the vectored interrupt facility.
3. Design C. Groups processes as follows:
 - EXEC
 - SENSIN, INPUT, OUTPUT
 - GYPROC, PROPAG, STRACK, STARID, UPDATE
4. Design D. No executive process. The operating system performs the executive functions as was done for the SMM OBC program (Reference 13).
5. Design E. Groups processes as follows:
 - EXEC
 - SENSIN
 - INPUT
 - OUTPUT
 - GYPROC, PROPAG
 - STRACK, STARID, UPDATE

Design B. The executive will be the main driver. All other processes will become subsystems and will be called by the

executive (not scheduled). This design may not be technically feasible, and it would be very difficult to implement regardless of its feasibility. The subsystems will be called in succession only when the previously active subsystem is completed. All asynchronous processes will be called by the operating system via vectored interrupts. The SENSIN process will be called after a timer interrupt to collect data at a high frequency. The high-frequency state propagation may have to wait for state update completion since the executive will not be able to suspend and resume processes. This design shows that, in general, functions that are performed at different frequencies should be designed as separate processes. This philosophy was used for Design A.

Design C. Functional integrity is destroyed because several diverse functions are grouped together without logical reason. Consequently, the clarity of design is destroyed; all the major scheduling logic is taken out of the executive and placed inside each process.

Design D. The operating system will assume the executive function of process scheduling; execution will be table driven. Each process will have an entry in the table. The entry will consist of the process location, priority, and period. Processes will be given control at the location given in the tables. Because this design was successfully used for the SMM mission, it will be compared in greater detail with the current design (Design A). See Reference 13 for further details on Design D.

<u>Design D</u>	<u>Design A</u>
1. Processes are accessed by location in the table. Therefore, the system is more general.	Processes are accessed by name. Scheduling is specific to mission but therefore easier to understand.

Design D

2. Easy to change processes when either sensors or the computational methods are changed. Only the new table has to be designed.
3. Nonnominal processing will be handled to a greater extent by each process. This logic will be hidden inside the process.
4. The operating system is microprocessor-specific. It is usually written in the assembly language for that particular microprocessor and is very economical with respect to execution time and memory.
5. Not a portable system. Difficult to maintain because of assembly language coding.

Design A

The executive must be modified to accommodate new sensors and so on.

Nonnominal processing is handled at the top level by the executive resulting in simpler control at the top level. For example, the special processing requested during occultation is clearly seen at the executive level.

Cannot be as efficient.

Can be used on a different microprocessor if a compiler is available. It will be easier to maintain because of the use of a standard language.

Design A has been selected because of the importance of factor 5. In addition, Design D cannot be used because the specific microprocessor to be used for a flight-qualified AADS had not been defined at the time of design and because a fair amount of work had been already completed on Design A.

Design E. This design is similar to (current) Design A except that some of the processes have been combined. This will save approximately 6 kilobytes of memory since the mathematical function library will not be duplicated in the combined processes. However, additional logic will be needed inside these combined processes because gyro data

processing and state propagation functions are performed at different frequencies and if star identification is performed on each star group instead of on all groups. Because the 6-kilobyte memory savings achieved in Design E is not critically important, Design A was selected. However, the design can be modified if the experience gained during implementation of the prototype warrants it. It is also expected that redesign because of requirement changes will be easier because of the simplicity and flexibility of the current design (Design A).

Given that the scheduling requirements for the attitude computation functions have stabilized during the development period, Design E could have been selected instead of Design A. Again, the selection of Design E would put some of the scheduling logic inside the component processes instead of the executive. More important is a further change in scheduling requirement may require a large redesign effort depending on the nature of the change.

1.5.3 SUGGESTIONS FOR IMPROVEMENT

- Run the program on the designated computer under various conditions to obtain timing and accuracy information.
 - Use different frequencies for gyro propagation, output, and star processing to find optimal conditions.
 - Record all identified stars during long periods for studying star catalog requirements and performance of the existing star identification methods.
 - Run a Boole & Babbage Problem Program Evaluator (PPE) (Reference 14), if available, to obtain accurate timing estimates.

- Change the program only if current requirements are not met or if requirements have changed.

- It is possible that the accuracy requirement can be achieved at a gyro propagation period of 1024 milliseconds. Thus, the 1024-millisecond period may be used if attitude computation cannot be performed in the 512 milliseconds available or if more powerful star identification and attitude update methods are desired.

- Identify code for optimization. For example, star identification and state update will require large amounts of time, but it may not be possible to shorten the time required. The SENSIN process may be more critical since it runs at a high frequency. The sensor data collection function is quite easy and, therefore, may be coded in the Intel assembly language to minimize execution time.

- Consider reducing the size of the star catalog by omitting stars dimmer than a stellar magnitude of 6.2. There are approximately 1000 stars between the magnitude range of 6.2 to 6.5. This will save nearly 12 percent of the space required for the present catalog (8500 stars between the magnitudes of 3 to 6.5). This space may be used for other processes if required.

- Consider more powerful star identification methods such as triplet matching.

- Consider the addition of a coarse attitude determination function (Sun-magnetometer (Sun/Mag) sensors or Sun/infrared (Sun/IR) sensors). These functions could compute attitude to within a degree or less. This will make AADS completely autonomous since initial attitude estimates for star identification and attitude update can be provided by the coarse attitude function if required. The addition of

this function may also satisfy the current time and memory constraints in the following ways:

- Coarse attitude determination will be performed only when the attitude is not known to the accuracy required by the star identification process. Star identification will not be performed during this time.
- Direct or simple pairwise matching techniques may be used since attitude is known accurately; this will mean a simpler and smaller star identification process.

1.6 AADS OPERATION INFORMATION

1.6.1 OPERATION ON THE VAX COMPUTER

Executable images of all processes in AADS and the AADS simulator reside on the VAX disk files. A DCL procedure is used to initiate a simulation. GSS, SDS, and OSS (see Section 1.3.1) are invoked to start the simulation. SDS reads the header information from the simulation tape and makes it available to GSS for setting up commands and data for up-link. SDS also starts sensor data transmission to AADS to simulate the condition during a mission when the attitude sensors are active, even though AADS itself is not active. The operator selects AADS commands and processing options and OSS orbit propagation information from the displays provided by GSS and then sends this information to AADS via OSS. Section 1.6.4 describes in detail a startup scenario and AADS scheduling.

GSS accepts AADS output transmitted via OSS and prints it for operator evaluation. Attitude information and other information on biases, validation statistics, and star identification is plotted or printed in a tabular format as required for evaluating the simulation results. The operator

can test and/or control the system by sending new commands and new processing options. The available commands are as follows:

<u>Command</u>	<u>Function</u>
START	Synchronizes AADS clock and starts AADS operation
STOP	Performs an orderly shutdown of AADS after completing pending functions; AADS then goes into a "wait" state until a new START command is sent from ground
HALT	Halts star data processing and state update; state is propagated using gyro data
RESUME	Resumes star data processing and update functions
SET CLOCK	Synchronizes AADS clock to correct any time drift

Clock synchronization is not needed for the VAX version of AADS because the VAX clock maintains a single time for all processes in AADS and the AADS simulator. The SET CLOCK command is not implemented in the current version of AADS.

1.6.2 OPERATION ON THE INTEL-VAX COMPUTERS

To reiterate, the executable images of all processes reside on the VAX disk files; however, the AADS process images are stored in the Intel absolute object image format. All AADS simulator processes are invoked and prepared for the start of simulation as described in Section 1.6.1. At this point, a utility program on the VAX transmits the AADS process images to the Intel, where a loader loads the programs in memory, and gives control to the AADS executive. The current VAX time is transmitted to the Intel at the very end of the loading operation and is stored in a specified area of AADS. The startup and consequent operation of AADS is identical to that process described in Sections 1.6.1 and 1.6.4,

except for the following internal differences that do not affect the user:

1. A clock in the Intel computer maintains AADS time. However, this time is synchronized with the AADS time with a timer pulse taken from the VAX clock.

2. Communication between the AADS simulator and AADS is through the Intel operating system. In particular, the Intel operating system reads the sensor data transmitted by SDS and makes it available to AADS in a specified area inside AADS. This procedure simulates, as closely as possible, the function performed by a hardware device known as the RIU. The RIU accesses the data from sensors onboard the spacecraft and writes the data in a specified area in AADS memory.

1.6.3 CHANGES TO THE SYSTEM DURING FLIGHT QUALIFICATION

For actual flight usage the following changes will take place:

- AADS will execute on the Intel 8086 computer.
- The OSS will be replaced by the OBC.
- The SDS will be replaced by an RIU or some other hardware device that will directly update the data buffers using shared memory or direct-memory access techniques. The sensor data collection (SENSIN) process will be redesigned if necessary.
- The timing signal from the VAX-11/780 will probably be replaced by a central time base onboard the spacecraft. Code for accessing time may be changed. In addition, the SET CLOCK command may have to be implemented.

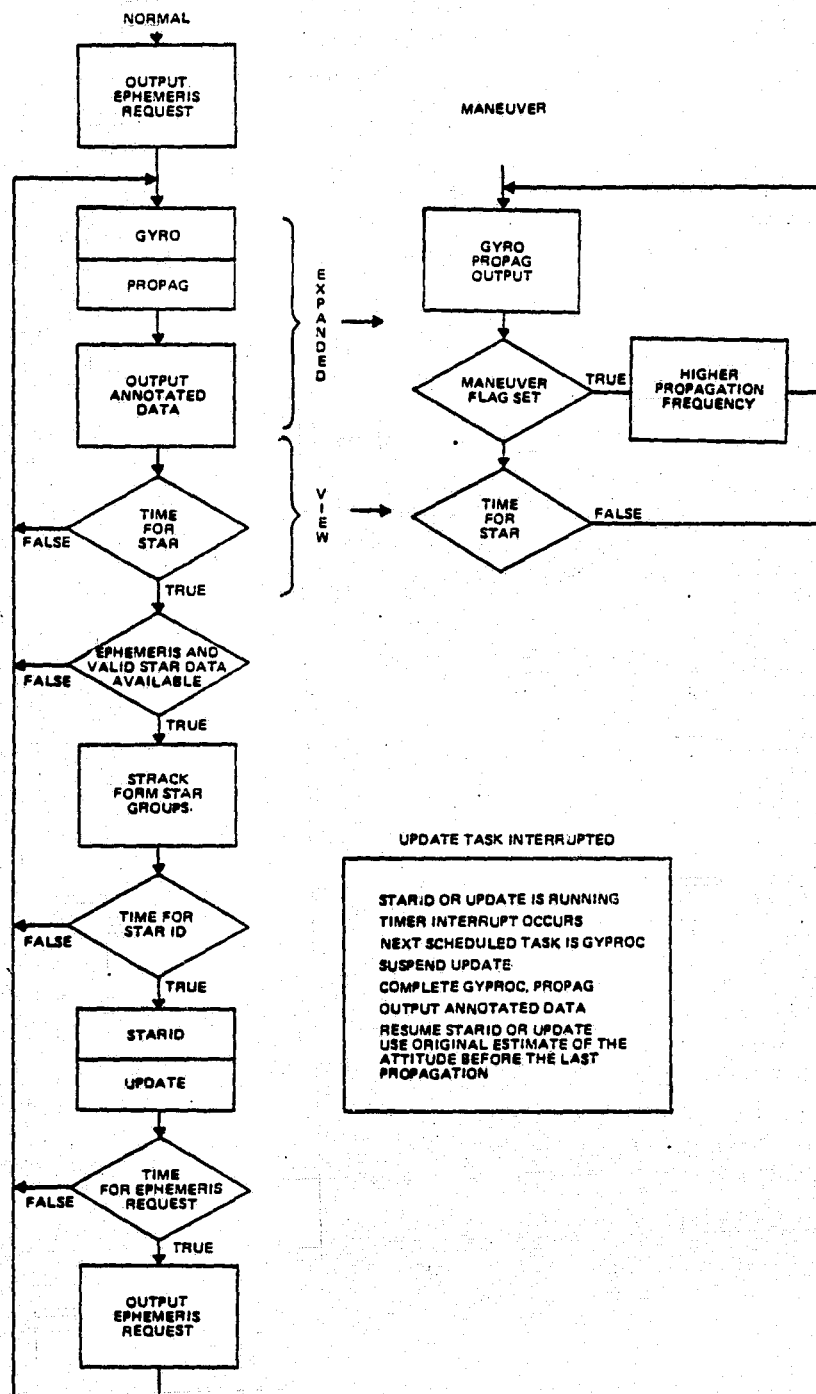
1.6.4 AADS STARTUP SCENARIO AND FLOW DIAGRAM

This section contains a startup scenario for AADS data processing when AADS is implemented on the Intel computer. Figure 1-5 is a flow diagram of the process.

The following events describe the startup scenario:

- After uplink, the operating system is running. The executive has been invoked. Various processes are initialized in a dormant (suspended) state except for the input data processing (INPUT) process, which is active and waiting for input after issuing a read request. No sensor readings are being taken.
- Uplink data to update several COMMON areas may be received. These COMMON areas contain AADS control parameters.
- A START command is received from the ground (any other command is invalid at this stage). The INPUT process accepts the command, sets a flag for the executive, issues another read request for future input, and suspends itself. As part of the startup procedure, the ground has sent time information to the operating system, and the system clock has been synchronized. This may be done when the system is initially loaded.
- The executive schedules the output data processing (OUTPUT) process to generate an ephemeris request and schedules the gyro data processing (GYPROC) and SENSIN processes. The OUTPUT process is scheduled at the proper frequency for data annotation output.
- The executive computes star camera occultation from ephemeris information. The star data processing functions are not invoked if (1) ephemeris data are

ORIGINAL PAGE IS
OF POOR QUALITY



NOTE: THE INPUT TASK HAS RECEIVED (OPTIONAL) TABLE UPDATE FOLLOWED BY A START COMMAND. CASES SHOWN INCLUDE NORMAL PROCESSING, PROCESSING DURING MANEUVER, AND PROCESSING WHEN THE UPDATE TASK IS INTERRUPTED AND DELAYED BY GYRO PROCESSING.

Figure 1-5. Normal Processing: AADS Initialized

not available, (2) star cameras are occulted, or (3) the spacecraft maneuver flag is on.

- Under normal circumstances (ephemeris data available, no occultation, and no maneuver), the executive will schedule the following processes:
 - High-frequency, high-priority SENSIN process to collect sensor data and perform some validation
 - High-frequency gyro processing and propagation processes to propagate attitude
 - OUTPUT process for attitude results to annotate science data
 - High-frequency star data processing process to collect star data, to perform data validation, and to form tracker point groups
- NOTE:** This process is rescheduled at a high frequency N seconds after an attitude update and continues periodically until the next update.
- Star identification and attitude update processes, respectively to output when sufficient star groups are formed and sufficient stars are identified
 - OUTPUT process to output attitude results after update
 - OUTPUT process to generate ephemeris request every 20 minutes

1.6.5 UPLINK OF CONTROL PARAMETERS

AADS commands and processing options may be specified interactively through GSS. The commands are described in

Section 1.6.1. AADS processing options are divided into four broad categories:

1. Scheduling frequencies and priorities
2. Propagation and update parameters
3. Star processing and identification parameters
4. Occultation prediction parameters

A series of menus and parameter displays allows the user to enter the desired input. Figure 1-6 shows a sample of AADS processing options. See Reference 4 for further information. GSS formats the processing options into transmission records, which are then uplinked to AADS. An uplink consists of a command and/or one or more of the four categories of processing options just defined. The new values of the processing options are stored in AADS COMMON areas and used thereafter.

1.6.6 INPUT PROCESSING

- When the Intel is turned on, it passes control to the AADS executive. The executive invokes the INPUT process, which in turn issues a read request, and then suspends itself. The INPUT process always waits for a command, data from the ground, or ephemeris data from the OBC/OSS at a high priority.
 - It sets proper flags for the executive when commands are received from the ground.
 - It stores uplink data for table (COMMON) updates in a buffer area.
 - It updates the ephemeris table from the information given by the OBC.
- After processing input, the INPUT process issues another read request and then suspends itself.

STAT1,	UPLINK TASK SCHEDULES (YES/NO)	NO
	<u>NOMINAL TASK PERIODS</u>	
SNDELT1,	GYRO DATA COLLECTION (SECONDS)	0.064
SNDELT2,	STAR CAMERA DATA COLLECTION (SECONDS)	0.100
GYDELT1,	GYRO DATA PROCESSING (SECONDS)	0.512
IGYRP1,	MULTIPLE OF GYDELT1 FOR SCHEDULING PROPAGATION (NO UNIT)	1
NPROUT1,	NUMBER OF TIMES PROPAGATION SCHEDULED BEFORE DATA ANNOTATION REPORT (NO UNIT)	1
STDELT,	STAR DATA PROCESSING (SECONDS)	2.0
UPLIM,	STAR IDENTIFICATION AND UPDATE	420.0
UPDELT,	DIFFERENCE BETWEEN THE LAST SCHEDULED UPDATE AND NEXT STAR DATA PROCESSING (SECONDS)	30.0

GYRO PERIODS DURING A MANEUVER

GYDELT2,	GYRO DATA PROCESSING (SECONDS)	0.256
IGYRP2,	MULTIPLE OF GYDELT2 FOR SCHEDULING PROPAGATION (NO UNIT)	1
NPROUT2,	NUMBER OF TIMES PROPAGATION SCHEDULED BEFORE DATA ANNOTATION REPORT	1

Figure 1-6. Sample of AADS Options

- The executive will reactivate the INPUT process at the end of a major cycle to update COMMON areas or to validate ephemeris data and information received earlier. A major cycle includes all AADS activities up to and including a state update.
- COMMON areas are updated at the end of an attitude update (major cycle) so that the attitude computation will not be disrupted during a major cycle. If several COMMON areas are to be updated, the ground will send commands to STOP processing, to update COMMON areas, and then to START processing.

SECTION 2 - SPECIFICATIONS SUMMARY

This specifications summary has been compiled from References 5 through 15, and those documents should be consulted for further details. Specific references also appear in the text.

2.1 MAIN REQUIREMENTS

The main AADS requirements are as follows:

- AADS will compute the spacecraft attitude to annotate science data onboard the spacecraft.
- The AADS design will meet the requirements of NASA's MMS series. It will support spacecraft with attitudes that either vary smoothly in time (Landsat-D type) or are nearly inertially fixed (SMM type).
- AADS will maintain an attitude accuracy of 30 arc seconds (3σ) for each axis.
- AADS will perform in a real-time mode and in an autonomous manner for a long period of time and will require only infrequent ground support.
- AADS will accept commands and data from the ground and will transmit attitude results, update results, activity logs, error messages, and raw sensor data to the ground.
- AADS will be implemented as a portable modular unit on a microprocessor onboard the spacecraft. The microprocessor will have 256 kilobytes of memory and no peripheral devices. A prototype system will be implemented on a VAX-11/780 computer to test feasibility.

- AADS will use gyro and star tracker data. Gyro data are used to propagate attitude, and data from identified stars are used to update the attitude using a linear optimal filter (Kalman filter) that is stable against roundoff errors.
- AADS will provide for contingencies and error handling.

2.2 DESIGN CONSTRAINTS

AADS will have the design goals of functional integrity, functional strength, flexibility of design, and maintainability, while meeting the following design constraints:

- 240 kilobytes of memory space is available for AADS. No peripheral devices will be available, compelling all code and data to reside in memory.
- Multiprocessing is supported. Maximum process size is restricted to 64 kilobytes of memory for instructions and 64 kilobytes of memory for data. Overlay structures are not permitted.
- AADS will execute in real time. It will perform gyro processing and propagation within a nominal period of 512 milliseconds and will update the attitude using star tracker data nominally at least once every 7 minutes. These periods are variable and can be changed during processing by ground commands.
- AADS will use a propagation and update scheme to compute the attitude with an accuracy of 30 arc seconds (3σ) for each axis. Variables that are required to be used in double-precision to achieve accuracy goals will be identified.
- AADS will be implemented initially on a VAX-11/780 computer for test purposes. Later, it will be transferred to a microprocessor. (The target microprocessor is the Intel 8086.)

The following design changes will be necessary before transfer to the target microprocessor can be accomplished:

(1) modify the input and output interface components if required to provide proper communication, (2) modify the system clock logic to interface with the Intel operating system, and (3) establish specific memory maps as required.

2.2.1 HARDWARE REQUIREMENTS

AADS hardware requirements are as follows:

- AADS code developed and tested on the VAX-11/780 will be downloaded to the Intel 8086/8087. The code will be downloaded using the same asynchronous parallel line over which the VAX-installed simulators and AADS communicate. A micro-based operating system and Intel-VAX communications software will be resident on the Intel. System software will be available to compile AADS source and create AADS executable images on the Intel 8086.

The target computer should be equivalent to the Intel 8086 microprocessor with respect to execution time, memory requirements, and system architecture (see Reference 12 for additional details and specific values) as follows:

- Memory addressing capability of 256 kilobytes
- Direct addressing capability of 64 kilobytes
- Cycle times, memory access times, and instruction execution times comparable to the Intel 8086
- Floating-point arithmetic capability with an efficient floating-point instruction set, including single- and double-precision add, subtract, multiply, and divide instructions
- 16-bit arithmetic, addressing protection, interrupt for arithmetic exception, mechanisms to provide for position-independent code

Communication between the VAX-11/780 computer and the Intel 8086 will be needed during development and testing. Figures 2-1 and 2-2 show a typical system configuration.

2.2.2 OPERATING SYSTEM REQUIREMENTS (Intel-VAX CONFIGURATION)

The Intel operating system requirements are as follows:

- A number of system software packages must be developed for the Intel 8086 and VAX-11/780 to execute AADS and the AADS simulator simultaneously.

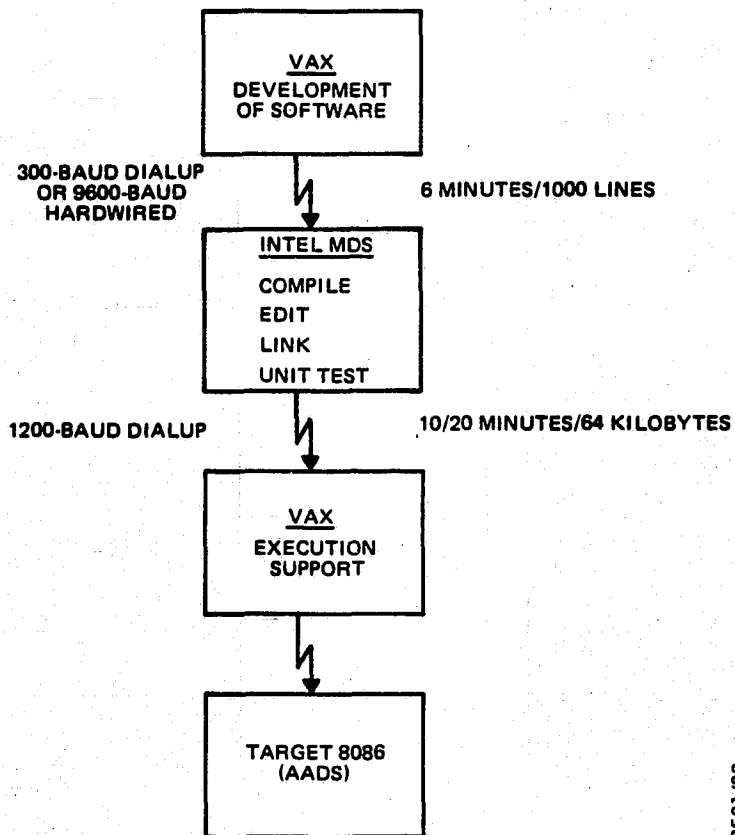
- The Intel operating system must be compatible with the VAX operating system, so that major design changes will not be needed when AADS is transferred to the Intel.

- The operating system must support multiprocessing, and it must have the capability to initiate (schedule) and to terminate process images (up to 64 kilobytes of memory) after the process images have been loaded into the RAM of the Intel 8086 according to their priority levels. There will be no requirement to remove processes after execution completion.

- A designated process (executive) must be able to set priorities and to start, suspend, resume, and halt other processes. Other processes must be able to suspend themselves.

- The operating system must provide the following services associated with the system timing requirements:

- Time of day and day of year
- An interrupt (approximately) every 10 milliseconds to the Intel for clock maintenance; an interrupt after N milliseconds where N is set by the executive process



8581/82

Figure 2-1. Computer System Configuration and Information Transfer Rates

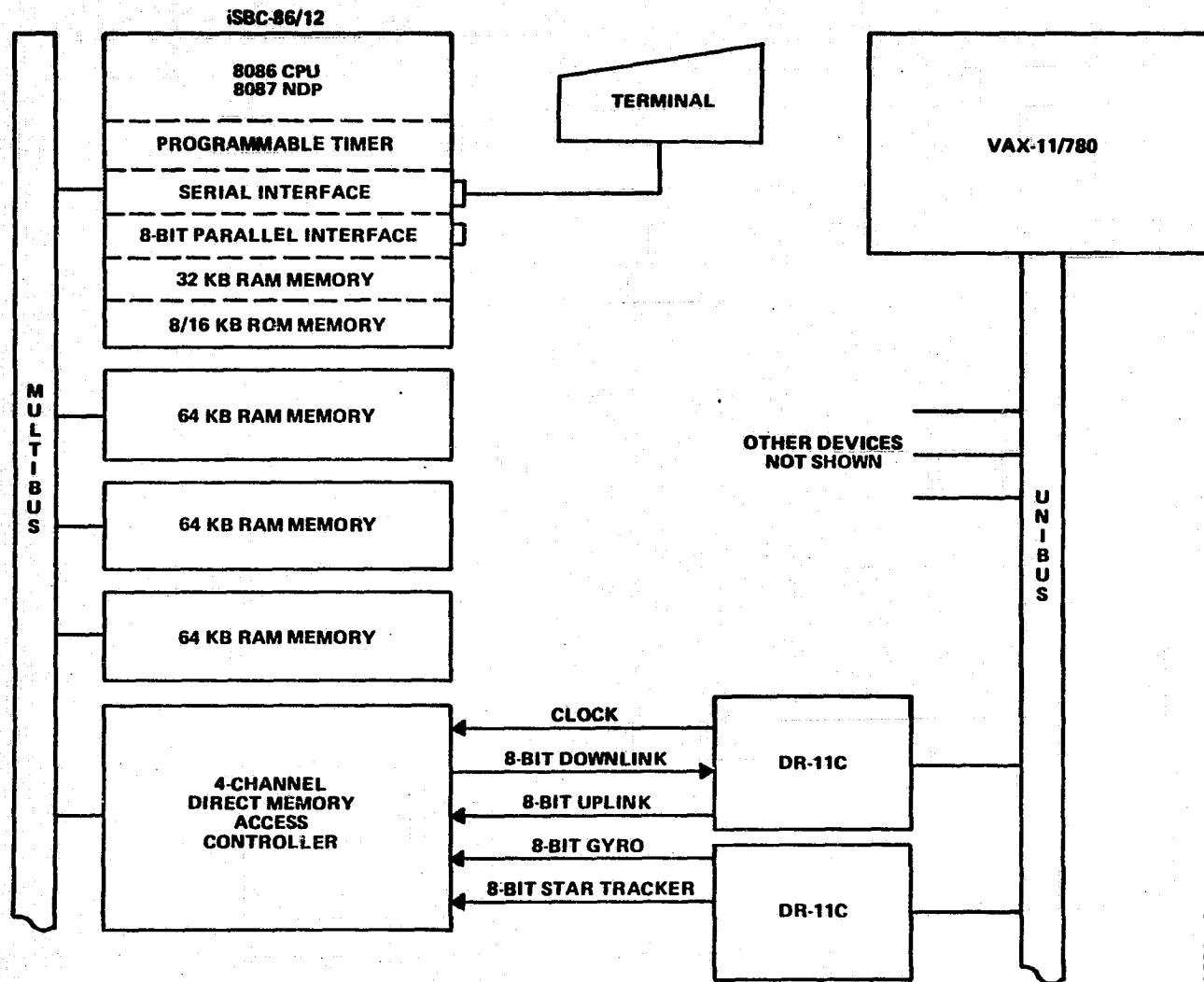


Figure 2-2. VAX-Intel System Configuration

ORIGINAL PAGE IS
OF POOR QUALITY

28/188

- Synchronization (reset) of the clock in response to a ground command
- Process wait and periodic process execution
- The operating system must support global COMMON areas, but the mathematical subroutine library will not be shared among processes. The operating system and the mathematical package will require approximately 16 kilobytes of memory.
- The operating system must be able to detect arithmetic exceptions and to take appropriate actions (that is, transfer control to an error message routine to generate a descriptive message and wait for user action). The operating system will provide addressing protection and bus timeout indication.
- Two parallel channels using DR-11C drivers must be available for communication between the VAX and the Intel computers.
- After the AADS prototype is moved to the Intel 8086 computer, the AADS simulator will remain on the VAX and will communicate with the AADS via parallel transmission lines.
- The operating system must have a facility to accept incoming messages from another processor asynchronously. When a message is sent via a serial or a parallel line from another processor (the VAX), all currently executing processes should be suspended, and a high-priority handler should be initiated immediately to accept and store the incoming message. Upon reception, a predetermined flag should be turned on to notify any applications software that may be waiting for the arrival of the message. Finally, all suspended programs must be resumed to continue the normal execution.

- Assuming that executable images of the operating systems and a number of applications programs exist on VAX files, a facility to downline load these images into the RAM of the Intel 8086 will be needed to run various AADS simulation cases. This facility would require the development of software for both the VAX and the Intel computer systems. Only the available (currently existing) interface modules will be used for this purpose. The wiring will be determined later when the interface is established. The speed of data transfer should be at least 9600 bits per second. This facility is needed if a cross-compiler and linkage editor program resident on the VAX are used to create the Intel-executable image. If the cross-compiler is not used, a FORTRAN compiler and linkage editor are needed for the Intel. (It should be noted that the cross-compiler is not needed because the Intel Development System has a FORTRAN compiler and linkage editor.)

- AADS will be implemented using a FORTRAN language. However, it is possible that a few assembly language routines (invoked by a FORTRAN CALL statement) may have to be coded to implement certain data structures, and for time and memory optimization, if required. In that case, an assembler is needed for the Intel 8086 assembly language.

- CSC will identify additional operating system capabilities (such as mailbox, send-receive type communication, and utility software).

- The operating system will provide any memory dump or memory load capability if AADS or the star catalog is required to be uplinked and/or downlinked.

2.2.3 FORTRAN COMPILER REQUIREMENTS

FORTRAN compiler requirements are as follows:

- Support double-precision arithmetic
- Interface with the Intel 8087 numeric data processor
- Provide a single-precision mathematical subroutine library to include trigonometric, logarithmic, and other required routines
- Support calls to assembly language routines via a FORTRAN CALL statement

2.2.4 AADS SIMULATOR REQUIREMENTS

AADS simulator requirements are summarized below. See the AADS simulator requirements document (Reference 2) for a detailed explanation of the requirements.

- The AADS simulator is intended to support the operation of AADS in a laboratory environment. To achieve this, it must (1) simulate or provide gyro and star tracker data--Sensor Data Simulator (SDS), (2) perform onboard functions such as spacecraft ephemeris computation and time information--Onboard Support Simulator (OSS), and (3) simulate ground support functions--Ground Support Simulator (GSS).

- The simulator will provide input test data (simulated gyro and star camera measurements) and reference output vectors (for example, reduced and calibrated gyro and star vectors and catalog star identification numbers).

- Various commands are sent to AADS via the simulator; in addition, data for AADS control parameter modification is uplinked. The simulator generates reports of the telemetry from the spacecraft, attitude results, and activity logs transmitted by AADS.

- The AADS simulator will provide data from a Landsat-D simulator generated tape in the AADS format. See Reference 3 for a description of this format.

- Sensor data (gyro counts, star tracker data, and intensity and temperature information) are stored by SDS in a buffer area for use by AADS. The data are not queued but are refreshed at regular intervals.

- Gyro data are refreshed every 64 milliseconds.

- Star tracker data are refreshed at 50-millisecond intervals for alternate star trackers (or 100-millisecond interval for both trackers).

- OSS will contain a full orbit generator. It will be able to generate spacecraft ephemeris data for a given timespan and frequency. The orbit accuracy is expected to be within 500 meters over 24 hours.

- The gyros will operate in the following two modes: (1) high rate mode has a resolution of 0.8 arc-second and a saturation value of 1.6 degrees; and (2) low rate mode has a resolution of 0.05 arc-second and a saturation value of 400 arc-seconds. The gyro data will be provided in 24 bits containing incremental gyro counts, a low/high mode flag bit, and a power on/off bit for each of the six channels.

- Star camera output includes two 12-bit words for horizontal (H) and vertical (V) counts. The H and V counts, when converted, give the X and Y components of the unit star vector in the star tracker frame. The resolution is 7 arc-seconds per count.

- The AADS simulator will contain options for transmission failure probability (TFP) and message corruption probability (MCP).

2.3 AADS EXECUTIVE REQUIREMENTS

AADS executive requirements are as follows:

- The executive will determine scheduling needs using a table of active processes, priority levels, and scheduling frequencies.

- The operating system will schedule the highest priority process currently pending. AADS executive will schedule the next subsystem (process) when the current subsystem (process) is suspended after using up the allotted time slice or after completing its activity. A ground command or a critical error will force the AADS executive to take appropriate action immediately.

- The executive will perform time slicing by scheduling "next wake up" call or timer request when a process completes. The time slice is implemented by the operating system using a mark-time utility function. Round-robin scheduling will be used where possible. This allows processes with equal priority to execute one after another during successive time slices. Not all the time slices are equal, and some of them are large compared with the usual round-robin scheduling.

- Processes will be scheduled by the executive and will use global COMMON areas for interprocess communication.

- The executive must be compact and efficient since it is executed at the end of each time slice, or when a process completes.

- In the case of severe errors, the executive will

- Schedule activity log, and raw sensor data report for immediate downlink
- Schedule messages for immediate downlink

- Suspend the error-producing process or take other appropriate action
- Allow other processes to continue processing

In the case of other errors, the executive will

- Enter them in the activity log if required
- Resume normal processing

- The executive will maintain an activity log containing statistics relating to the performance of AADS.

- The executive will initialize AADS.

- The SENSIN process will be scheduled at regular intervals to collect gyro and star camera data for processing.

- The executive will time the state update process and will generate critical error messages if a specified time limit is exceeded.

- The executive will compute occultation of the star camera field of view (FOV) by the Earth and Moon and will reschedule the star data processing function as required. The executive will compute Moon ephemeris data. The bright object sensor will shut down the star camera when the Sun approaches the camera FOV.

- The executive will schedule the OUTPUT process to request spacecraft ephemeris. This request will be generated when approximately one update cycle (7 minutes) of spacecraft ephemeris information remains in the ephemeris table.

- The INPUT process will be given the highest priority to capture ground commands or to store spacecraft ephemeris data transmitted by the OSS. The INPUT process will be resumed immediately after an update to validate

ephemeris data and to validate and store new table parameters (AADS control parameters).

- The executive will check for the maneuver flag set by the gyro data processing process and take the following actions:

- Reschedule state propagation at a higher frequency during the maneuver
- Initiate star processing immediately after the completion of the maneuver is detected

2.4 INPUT DATA PROCESSING REQUIREMENTS

AADS input data processing (INPUT) requirements are as follows:

- AADS will collect sensor data and will accept input messages that contain (1) data for updating data bases (AADS control parameters), (2) commands generated by ground control, and (3) spacecraft ephemeris data transmitted by the OBC. The INPUT process will perform input validation. It will check for gyro saturation and invalid ephemeris data times and intervals. It will also validate input commands and other data as required.

- Commands and data are accepted immediately upon transmission. However, validation and table updating (updating AADS control parameters) occur immediately after an update is completed.

- AADS ground-supplied data base parameters (AADS control parameters) include

- Process scheduling table, which contains information controlling the priority and frequency of AADS process execution
- Constants, which will be used throughout AADS processing. These will require only infrequent

changing. They include scale factors and biases for reducing raw attitude sensor data, tolerances for editing attitude sensor data, and physical constants

- Star catalog table, which contains the coordinates, magnitudes, and identification numbers for SKYMAP stars with stellar magnitudes brighter than 6.5. This table is stored in a COMMON area
- Kalman filter initialization parameters, which include the a priori state vector, the state covariance matrix, and the measurement noises. This table will be uplinked at the start of AADS execution and then later at the user's discretion
- Star identification control parameters, which include the tolerances for direct and pairwise match tests and the number of stars that must be identified before performing state update
- These parameters are described in greater detail in Section 4.4 and in the AADS simulator document (Reference 2)

- AADS will accept spacecraft ephemeris data provided by the spacecraft onboard computer.

- AADS will retrieve raw gyro and star sensor data that will be available in buffers in AADS memory. However, a read request must be issued to read data for the prototype version. Star tracker and gyro data will be refreshed in the buffers at periodic intervals. A high priority SENSIN process will collect star tracker and gyro data for attitude computation. The sensor data report will be downlinked in the event of a critical error.

2.4.1 MESSAGE FORMAT FOR UPLINK AND DOWNLINK

The basic message format for the data transmission between the AADS onboard support function and the ground simulators will be identical for uplinking and downlinking. The terms are defined as follows:

1. Record--A 20-byte header + data + filler to comprise 256 bytes
2. Block--A collection of records
3. Transmission--A collection of blocks in which the last record is an end-of-transmission (EOT) record

The detailed message format is specified in Section 4.6 and is described below for convenience.

- The spacecraft ephemeris request to the OBC/OSS will contain start time, period, and number of points (20 points, nominally at 1-minute intervals).

- The ephemeris table will have 40 entries and will contain the times along with the vectors for spacecraft position and velocity. The table will be stored in a wrap-around fashion. There will be no overlapping times in the table because of the wraparound design and because the ephemeris request is made at least one major cycle in advance.

- AADS control commands, given in the simulator requirement document, are as follows:

RESUME	Resume star processing
HALT	Abort star data processing but continue to propagate with gyro data
STOP	Stop current AADS processing in a normal manner, that is, finish the update process, stop propagating, and downlink the last output; wait for the next command

START

Start accepting data from sensors; begin normal propagate/update process when the epoch for the initial state is reached in real time

SET CLOCK

Set system clock to a new time

The control variables are described in Section 4.4 and in the AADS simulator (AADSIM) requirements (Reference 2). AADSIM will display these variables and block these properly for uplink. AADSIM will uplink the commands and/or data in the basic format.

- AADSIM will contain TFP and MCP. See the AADSIM requirements (Reference 2).

- The acknowledgment procedure (for example, INPUT process to the OBC(OSS), OBC to the INPUT process, and OUTPUT process to the OBC) has not been determined.

2.5 ATTITUDE DATA PROCESSING REQUIREMENTS

AADS attitude data processing requirements are as follows:

- AADS will provide propagated attitude information (for example, spacecraft quaternions and Euler angles) on a near-real time basis for experimental data annotation.

- AADS will use gyro and star sensor data to calculate, in an optimal fashion, estimates of the spacecraft attitude, gyro drift rate biases, and attitude uncertainty.

- AADS will generate commands requesting spacecraft ephemeris from the OBC. Sun and Moon ephemeris will be computed analytically.

- AADS will perform any preprocessing necessary to edit and validate raw attitude sensor data.

- AADS will access a resident star catalog that contains all the SKYMAP catalog stars down to a stellar magnitude of 6.5 (approximately 8700 stars). The entry for each

star will consist of right ascension, declination, and magnitude. The size of the star catalog will depend on the accuracy required for each quantity. Approximately 55 kilobytes of memory will be needed for the star catalog if 6 bytes are reserved for each star entry. This will give an accuracy of 0.00005 degree for the right ascension and declination angles and 0.05 units for star visual magnitude. See Section 3.10 for further details.

- AADS will operate an autonomous mode, receiving from the ground only control commands and data for updating the onboard data bases.

Attitude computation is performed by five separate processes. The detailed requirements for each process follow.

2.5.1 STATE DATA PROCESSING REQUIREMENTS

- Use the specified gyro configuration.
- Perform checks for limit, consistency, and redundancy between two gyro channels.
- Check the gyro rate mode flag for sensing maneuvers. If the high rate mode flag is on, then set a maneuver flag for the executive. The executive should suspend star data processing functions during the maneuver, and increase gyro data processing and propagation frequency.
- Reset the maneuver flag at the end of the maneuver when the gyro rate is low. The executive schedules the star data processing functions (if both star cameras are not occulted) to obtain an attitude update immediately following the maneuver.

2.5.2 STATE PROPAGATION REQUIREMENTS

- Use the accumulated gyro data to compute average gyro rate.

- Convert the gyro rate to body coordinates and eliminate drift rate biases.
- Use the gyro rate to propagate the quaternion.
- Propagate the covariance matrix.
- Store the gyro rate, quaternion, and Geocentric Inertial (GCI) to body rotation matrix for the star data processing processes.

2.5.3 STAR TRACKER DATA PROCESSING REQUIREMENTS

- Star data will be collected at a high frequency until sufficient star groups are formed. The frequency and duration of data collection is controlled by table parameters. Star data should be collected over relatively short time spans to reduce errors due to attitude propagation over large periods of time.
- Perform data editing and grouping. Reject invalid data through a limit check on H and V coordinates, temperature, and intensity. Reject the beginning and ending portion of data from a star.
- Compensate data for stellar image distortion due to camera temperature and star intensity effects.
- Correction due to the magnetic field effects is expected to be approximately 1 arc second. Therefore, magnetic field correction or the South Atlantic Anomaly check will not be performed.
- Form track point unit vector in tracker frame, and synchronize the unit vector to the nearest gyro-propagated quaternion time.
- Rotate the track point unit vector to geocentric inertial frame using the nearest gyro-propagated attitude.

- Average the track point unit vectors for each star.
- Determine the current attitude uncertainty from the last state covariance matrix. Select the number of star groups that are required to be complete before attempting star identification. Select the star identification method.
- Set a flag for the executive when sufficient star groups are complete.

2.5.4 STAR IDENTIFICATION REQUIREMENTS

- Return with error if spacecraft ephemeris data is not available for the data span.
- Compute solar ephemeris data.
- Correct observed (average) unit star vector for stellar aberration.
- Correct the star vector for precessional and nutational correction that are required because the AADS star catalog is created at epoch 2000.
- Obtain candidate stars from star catalog that are in the observation window for each observed star.
- Perform star identification by direct/pair matching techniques.
- Presently, there is no requirement to use a triplet matching technique.
- Pass observed star vectors and identified star vectors to the update process. Set a flag for the executive indicating successful star identification.
- Return with error if identification is not successful when fewer than the specified number of stars are identified in the given time and when a

specified percentage of observed stars could not be identified.

2.5.5 STATE UPDATE REQUIREMENTS

- Compute observed unit star vector in the star tracker frame.
- Compute predicted star unit vector in the star tracker frame.
- Compute observation residuals.
- Compute partial derivative matrix.
- Compute the gain matrix.
- Update the covariance matrix.
- Update the state vector.
- Update the drift rate biases.
- The attitude accuracy is 30 arc seconds (3[]) on each axis.
- An attitude update will be performed at least once every 7 minutes. Nominally, the updates are performed as soon as a star is identified. These update options are controlled from the ground, but update frequency depends on the star availability and current attitude uncertainty.

2.6 OUTPUT DATA PROCESSING REQUIREMENTS

AADS output data processing (OUTPUT) requirements are as follows:

- AADS will output propagated attitudes periodically in the form of quaternions and Euler angles (for example, roll, pitch, and yaw) for experimental data.

- AADS will output priority messages (critical error messages) to request special ground support such as error handling.
- AADS will output samples of raw attitude sensor data.
- AADS will output updated values of the attitude, gyro drift rate biases, state covariances matrix, and the Kalman filter gain matrix, as well as the SKYMAP catalog numbers of identified stars. (It should be noted that an internal star reference number can be sent to the ground to generate the actual SKYMAP numbers on the ground.)
- AADS will generate an activity log. It should be noted that the total number of updates attempted equals the total number of successful updates.
- AADS will issue messages requesting spacecraft ephemeris data from the onboard computer. The request will contain the start time, time increment, and number of data points. Nominally, 20 points will be requested at 1-minute intervals.
- The format of a downlinked record is the same as the format of an uplinked record.
- The OBC/OSS will indicate the start time of spacecraft ephemeris data, which will be validated by the INPUT process during ephemeris validation.
- The OUTPUT process will pass AADS output to the output buffer and will alert the OBC. The OBC(OSS) will transmit the output to the ground.
- The acknowledgment procedure (INPUT process to the OBC/OSS, OBC to the INPUT process, and OUTPUT process to the OBC) has not been determined.

See Section 4.6 for detailed formats of uplink/downlink messages.

2.7 SPECIAL CONSIDERATIONS

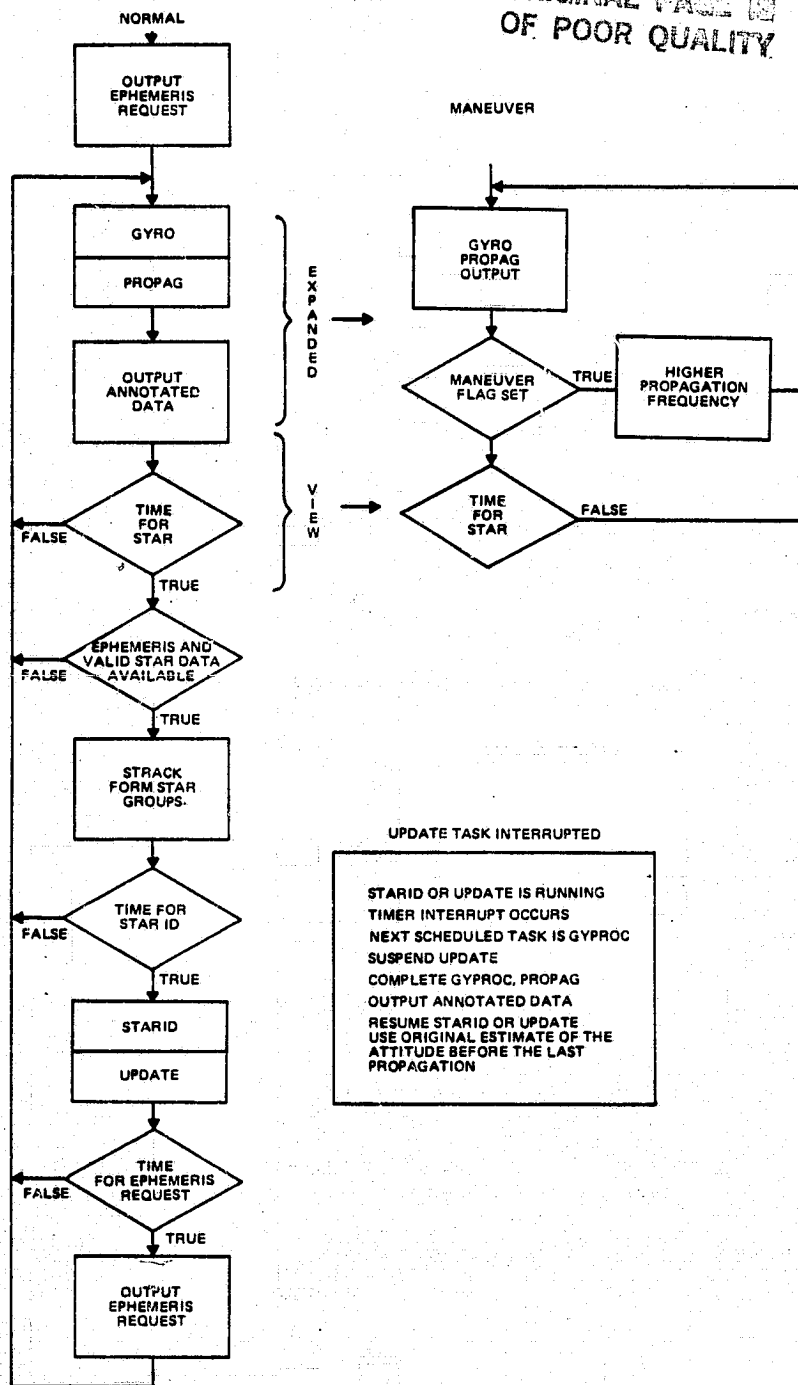
The following are AADS processing scenarios:

- INPUT processing scenario.
- Startup scenario and normal processing (see Figure 2-3).
- Processing to include star camera occultation: suspend star processing during occultation. Resume star processing; and update automatically when occultation ends.
- Processing to include a maneuver: suspend star data processing and state update; increase gyro frequency. Resume normal processing when the maneuver ends.
- Processing to include a critical error: stars not identified in the given time; all gyro data consistently flagged.

2.7.1 INPUT PROCESSING

- The INPUT process always waits for a command, data from the ground or ephemeris data from the OBC/OSS at a high priority.
 - It sets proper flags for the executive when commands are received from the ground
 - It updates the ephemeris table from the information given by the OBC
 - Data for table update are stored in a buffer area
- After processing input, the INPUT process issues another read request and then suspends itself.

ORIGINAL PAGE IS
OF POOR QUALITY



NOTE: THE INPUT TASK HAS RECEIVED (OPTIONAL) TABLE UPDATE FOLLOWED BY A START COMMAND. CASES SHOWN INCLUDE NORMAL PROCESSING, PROCESSING DURING MANEUVER, AND PROCESSING WHEN THE UPDATE TASK IS INTERRUPTED AND DELAYED BY GYRO PROCESSING.

Figure 2-3. Normal Processing; AADS Initialized (The INPUT process has received (optional) a table update followed by a START command.)

- The executive will reactivate the INPUT at the end of a major cycle to update COMMON areas or to validate ephemeris data and information received earlier.
- COMMON areas for AADS control parameters are updated at the end of a state update (major cycle) so that the attitude computation will not be disrupted during a major cycle. If several COMMON areas are to be updated, the ground will send commands to STOP processing, to update COMMON areas, and then to START processing.

2.7.2 AADS STARTUP SCENARIO

The following events describe the startup scenario:

- After uplink, the operating system is running. The executive has been invoked. Various processes are initialized in a dormant (suspended) state except for the INPUT, which is active and waiting for a command after issuing a read request. No sensor readings are being taken.
- Commands to update several COMMON areas may be received.
- A START command is received from the ground (any other command is invalid at this stage). The INPUT process accepts the command, sets a flag for the executive, issues another read request for future input, and suspends itself. As a part of the START procedure, the ground has sent time information to the operating system, and the system clock has been synchronized.
- The executive schedules the OUTPUT process to generate the ephemeris request and schedules the gyro data processing and SENSIN processes. The OUTPUT process is scheduled at the proper frequency. It should be noted that the SENSIN process is only scheduled when an actual (or

simulated) RIU (memory sharing, no interrupts) is used. Otherwise, the SENSIN process is interrupt driven.

- The executive computes star camera occultation from ephemeris information. The star data processing processes are not invoked if (1) ephemeris data is not available, (2) if both star cameras are occulted, or (3) if the attitude maneuver flag is on.

- Under normal circumstances (ephemeris data available, no occultation, and no maneuver), the executive will schedule the following processes:

- High frequency, high priority SENSIN process to collect sensor data and perform some validation
- High frequency gyro processing and propagation functions to propagate attitude
- OUTPUT process to prepare data annotation reports
- High frequency star data processing process to collect star data, perform data validation, and form snapshots. The process is scheduled at a high frequency N seconds before the scheduled attitude update
- Star identification and attitude update processes when star group formation is successful
- OUTPUT process to output update results at the update report frequency
- OUTPUT process to generate an ephemeris request every 20 minutes

- Again, all scheduling periods are variable, and can be selected from the ground.

2.7.3 STAR CAMERA OCCULTATION

- AADS is running normally scheduling all processes. The last process completed was gyro propagation.
- The executive will check for occultation of the star camera's FOV by the Sun, Earth, and Moon.
- If both star cameras are occulted, the executive will schedule star identification and attitude update if there are enough data for at least one star. The attitude will be updated when at least one star has been identified depending on the current attitude uncertainty.
- If both cameras are occulted, and there is no star data or if collected star data is not sufficient to form a tracker group, the executive will schedule the star data processing functions at a later time.
- The executive will schedule only gyro processing functions until such time as at least one star tracker is not occulted. A gyro-propagated attitude will be used for data annotation.
- The executive will schedule the star processing functions as soon as occultation has ended to obtain an immediate state update.

2.7.4 ATTITUDE MANEUVER

- AADS is running normally scheduling all processes. The next process to be scheduled is state propagation.
- State propagation will read gyro data, which contains a bit indicating a high/low gyro mode. If the bit value signifies the high-rate mode, then the gyro propagation process will set the maneuver flag in a global COMMON area. The gyro rates will be computed using a different conversion constant.

- The executive will check the maneuver flag when the gyro propagation process is completed or suspended.
- If the maneuver flag is set, the executive takes the following actions:
 - Schedules immediate state update if at least one star has been identified; otherwise, suspends the star processing processes
 - Uses a different (higher) frequency for scheduling gyro processing and gyro propagation
 - Continues data annotation with the gyro-propagated attitude estimates
- When the maneuver flag is reset by the gyro propagation process (indicating low-rate mode for gyros), then (1) schedule the star processing process immediately to get an attitude update and (2) resume gyro processing and propagation with normal (lower) frequencies.

2.7.5 CONTINGENCIES AND ERROR HANDLING

The following instructions describe how to handle contingencies and errors.

2.7.5.1 Minor Errors

If:

- All gyro data are flagged before a gyro propagation
- A tracker group was rejected during validation
- A tracker group could not be identified

Then,

- Add error messages to the error log
- Continue processing normally

There is one error log (OUTBUF). The executive does not schedule the OUTPUT process immediately for minor errors.

2.7.5.2 Severe Errors

If:

- Several successive failures occur during star identification and no stars are identified during a specified time period:

Then,

- Schedule the OUTPUT process for sending a critical error message
- Suspend star processing; continue gyro processing and data annotation with gyro propagation
- Wait for table modification and a RESUME (resume star processing) command from the ground

If:

- All gyro data are consistently flagged for a specified period:

Then,

- Send a critical message to the ground
- Downlink all sensor raw data from the buffer
- Halt processing and wait for a table modification and START command

If:

- There is an arithmetic exception:

Then,

- Take a specified (TBD) action (set the result equal to 0.0)
- Send a critical error message to the ground
- Continue processing

- Ground will probably request software dump from the operating system.

If:

- There is an addressing exception (attempted overwrite):

Then,

- Send a critical error message to the ground
- Send all raw data to the ground
- Shut down and wait for a START command
- The ground will request a software dump

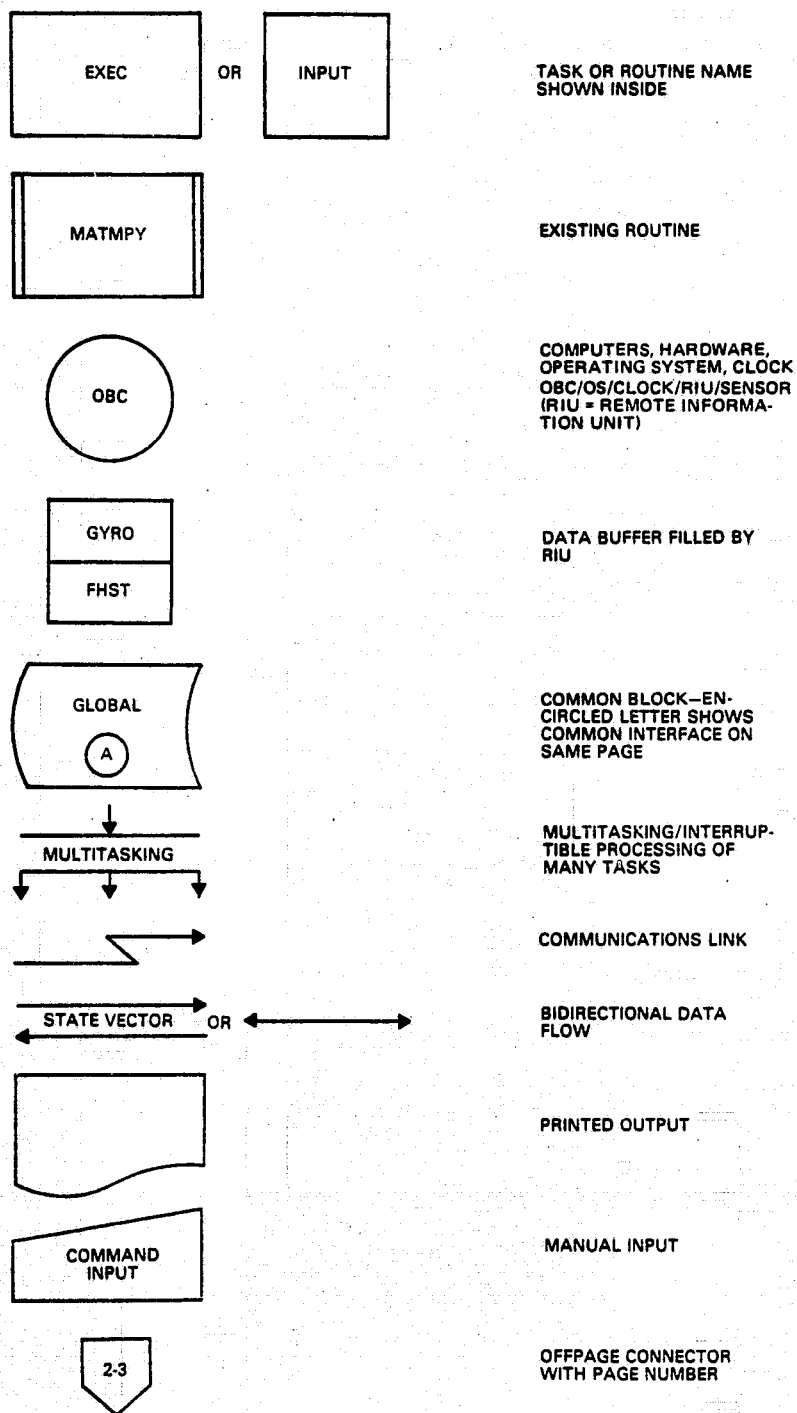
SECTION 3 - AADS BASELINE DIAGRAMS

This section presents detailed baseline diagrams for AADS as a whole and for various processes within the system. This section also contains processing overviews for and descriptions of components in the processes. Figure 3-1 describes the flowcharting symbols used in the various diagrams. Figures 3-2 and 3-3 show the AADS external and internal interfaces, respectively.

Each of the first nine subsections describes one process in AADS. Each subsection describes the input data required for processing and the output from the process. A simple figure illustrates the input and output. The processing in each case is also described and is intended for use with the data flow and baseline diagrams. A table contains a description of each component in the baseline diagrams. Section 3.10 describes the AADS star catalog generation utility.

It should be noted that /STATIC/ refers to the four COMMON areas /STAT1/, /STAT2/, /STAT3/, and /STAT4/ for convenience. These COMMON areas contain the AADS control parameters.

ORIGINAL PAGE IS
OF POOR QUALITY.



8561/82

Figure 3-1. Flowcharting Symbols

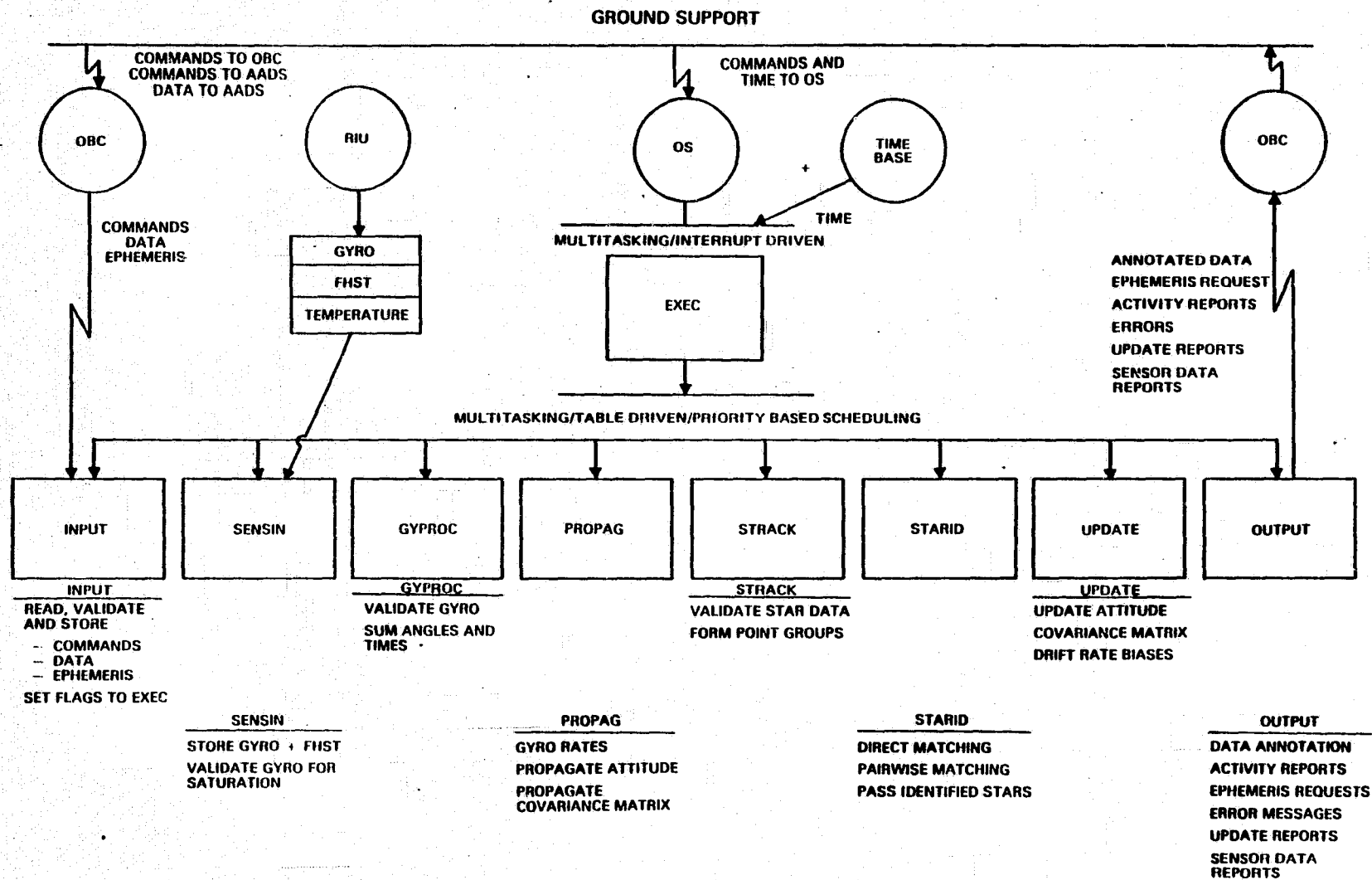


Figure 3-2. AADS External Interfaces

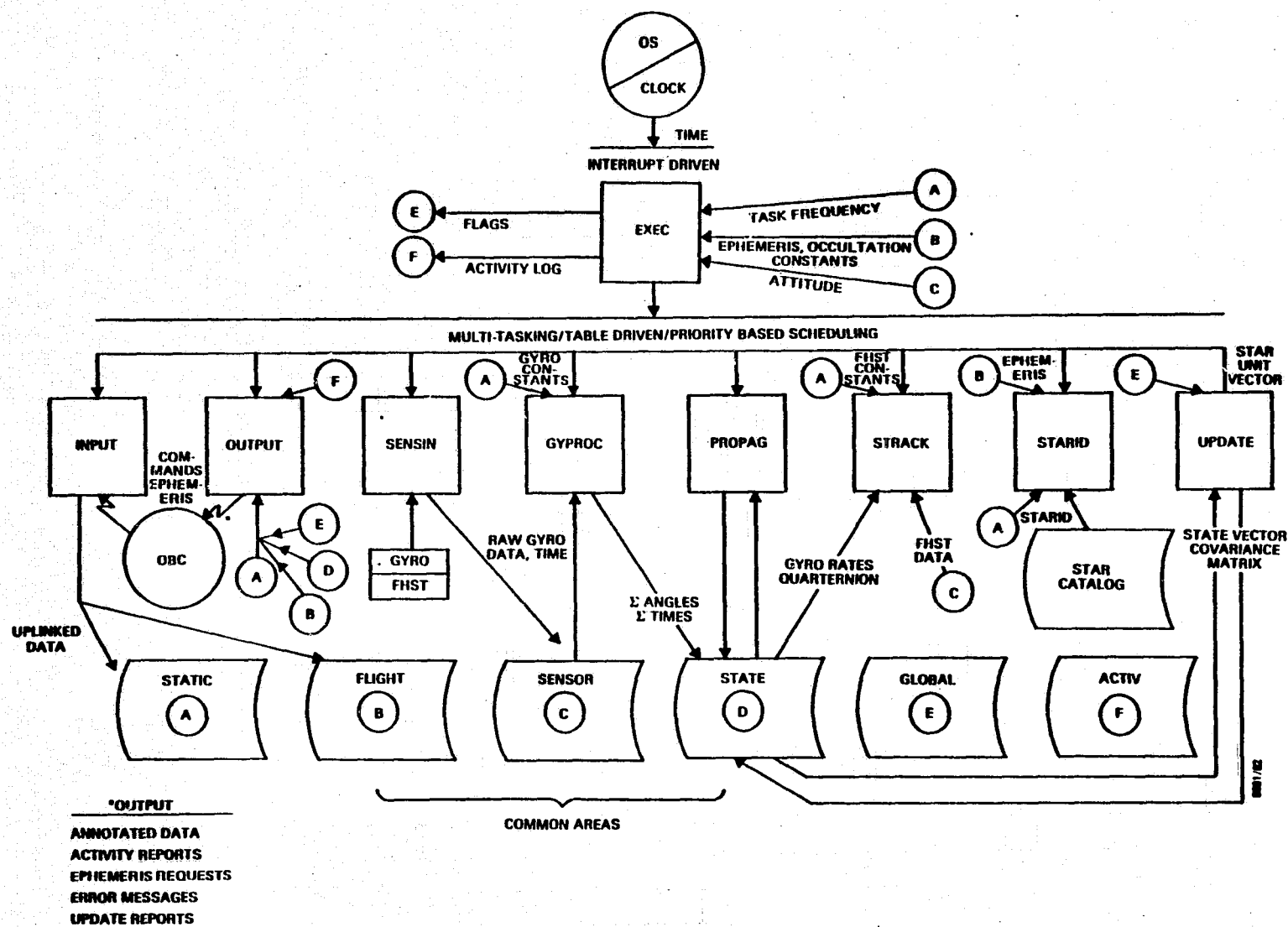


Figure 3-3. AADS Internal Interfaces

ORIGINAL PAGE IS
OF POOR QUALITY

3.1 EXECUTIVE PROCESS

This section describes the executive (EXEC) process.

3.1.1 INPUT/OUTPUT

The following subsections describe the input and output parameters for the EXEC process. Figure 3-4 illustrates the external interfaces.

3.1.1.1 Input

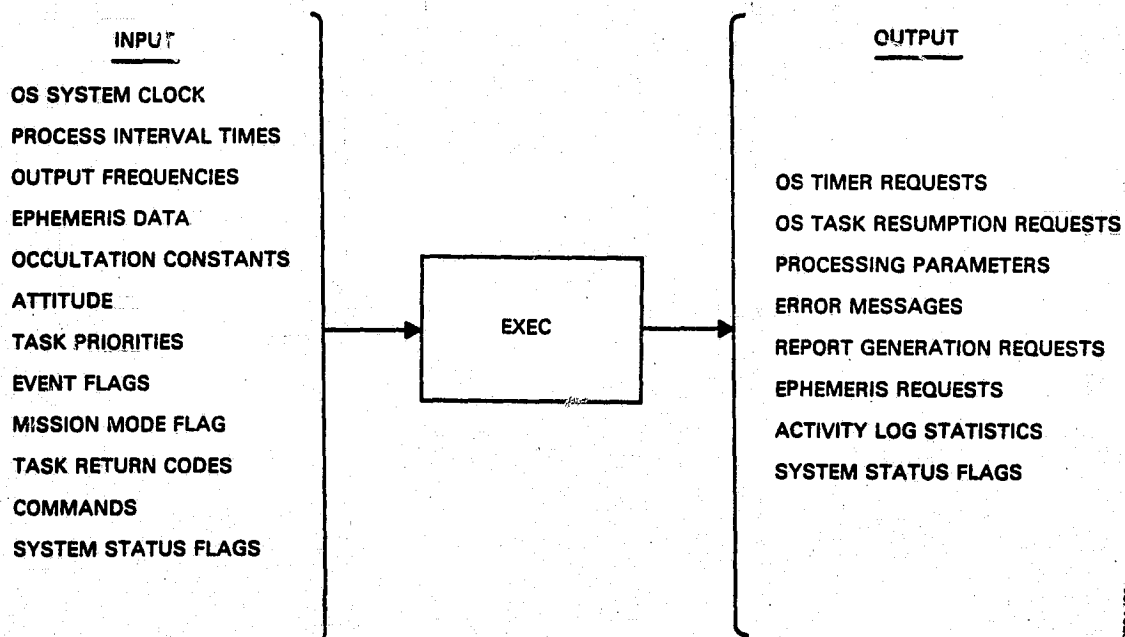
The following items are required as input:

- Constants and data needed for scheduling
 - Operating system (OS) clock and timer interrupts
 - Process interval times
 - Output frequencies
 - Process priorities
 - Event flags and process completion interrupts
 - Maneuver flag
 - Commands
 - Process return codes and status flags
- Constants and data needed for occultation prediction
 - Ephemeris data
 - Occultation constants
 - Spacecraft attitude

3.1.1.2 Output

The following items are output from the EXEC process:

- Parameters and operating system requests for scheduling



8581/82

Figure 3-4. EXEC Process External Interfaces

- OS timer requests
- OS process resumption requests
- Input processing parameters and system status flags
- Requests, messages, and reports for output
 - Error messages
 - Report-generation requests
 - Ephemeris requests
- Activity log statistics

3.1.2 PROCESSING

The EXEC process functions as a miniature operating system and schedules various AADS processes as required. The EXEC process resumes when a command is received, a timer interrupt occurs, or one of the other processes completes execution. The EXEC process performs required executive functions at a high priority.

AADS is initialized and VAX-specific initialization is performed. Commands are processed. Normal scheduling of processes is performed, and the scheduling is changed when maneuver or gyro saturation conditions are detected. When computations show that both star cameras are occulted, resumption of the star data processing (STRACK) function is delayed.

Scheduling is modified or other error handling is performed in response to process return codes. Attitude information for data annotation and reports is output at required periods. The ephemeris is checked at the end of a major cycle, and a request for more ephemeris data is made when insufficient ephemeris data remain for the next major cycle. The input data processing (INPUT) function is invoked when possible to validate ephemeris that is

received, or at the end of a major cycle to validate input data. The activity log is maintained. The state update cycle is timed, and a critical error message is generated when the update time limit is exceeded.

Table 3-1 contains the descriptions of EXEC process components to be used with Figures 3-5 and 3-6. Figure 3-5 illustrates data flow within the process and Figure 3-6 illustrates the relationship among process components.

Table 3-1. EXEC Process Component Descriptions (1 of 2)

<u>Component</u>	<u>Description</u>
MAIN	Driver for AADS executive process; initializes local COMMON variables and calls a subroutine to initialize AADS; waits for and tests event flags and calls subroutines to direct processing as needed by time-event, command reception, or process completion
INITL	Performs VAX-specific initialization and is driver for AADS initialization
PROCESSES	Creates each AADS process (except EXEC), allowing each process to initialize; priorities are set in process creation
FLAGS	Initializes AADS global COMMON areas
CMDPRO	Reads a command from buffer and processes it
TIMEUP	Resumes next scheduled process, updates schedule, and resets timer; checks for ephemeris data requiring validation
TSKDNE	Sets flags, increments counters, queues requests for reports, and resumes processes based on completion of a process
ACTLOG	Driver for subroutines to maintain activity log and to queue requests for reports and error messages
ENQUE	Queues error messages and requests for reports
BLDLOG	Maintains and updates activity log
CANCEL	Performs an orderly shutdown of AADS
UPSTAT	Directs state transitions in response to START, HALT, and RESUME commands
OCCULT	Checks for occultation of both star cameras
SCHED	Inserts next scheduled time into a table for a process and resets timer
TIMOBT	Obtains current simulation time and converts it to either seconds from September 1, 1957, or YYMMDD.HHMMSSMMM format
/EXEC/	Local COMMON area; which contains flags, schedule, and process identification number used by AADS executive.

Table 3-1. EXEC Process Component Descriptions (2 of 2)

<u>Component</u>	<u>Description</u>
B8ADD	Adds two integer quadwords
INTERP	Performs a six-point interpolation of ephemeris
OPENGL	User open routine for temporary global COMMON data sets

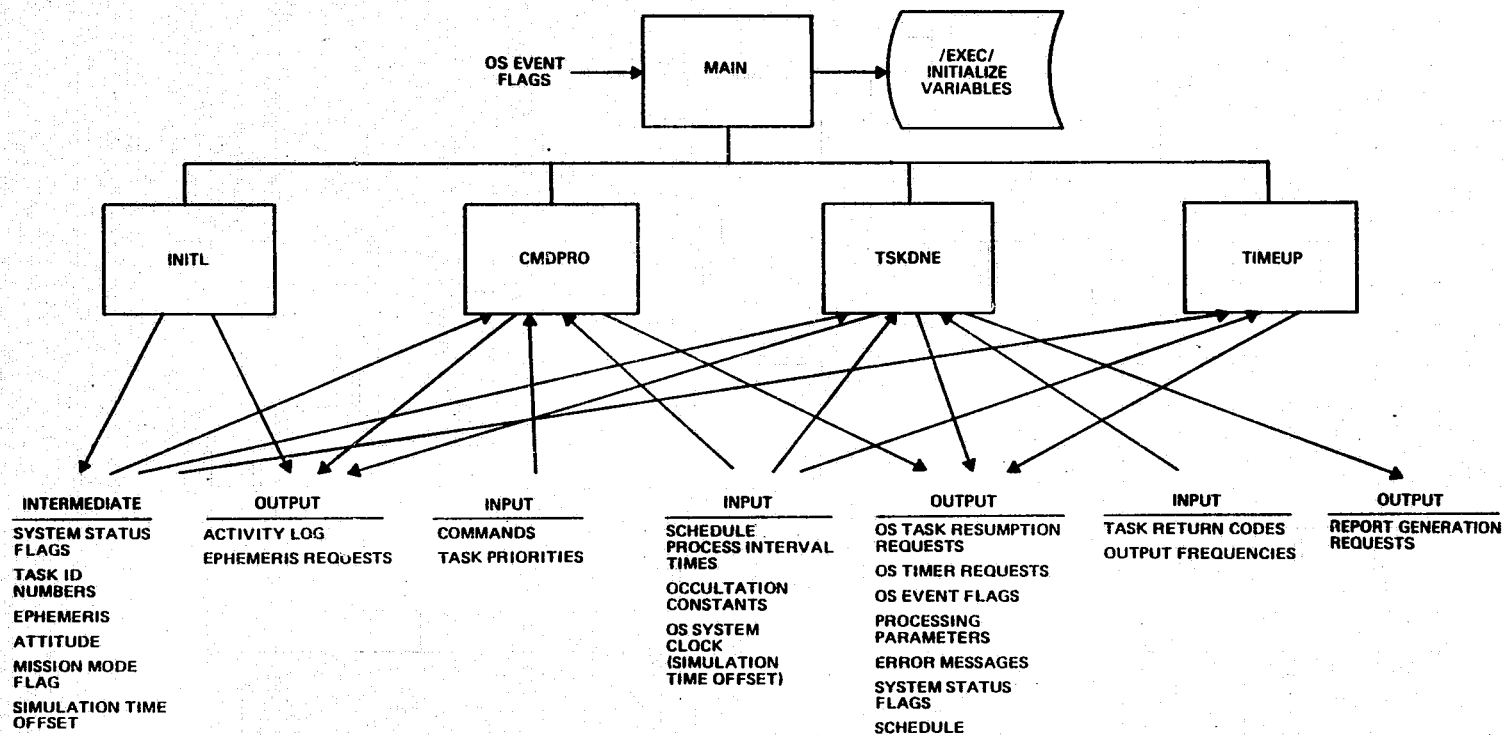


Figure 3-5. EXEC Process Data Flow

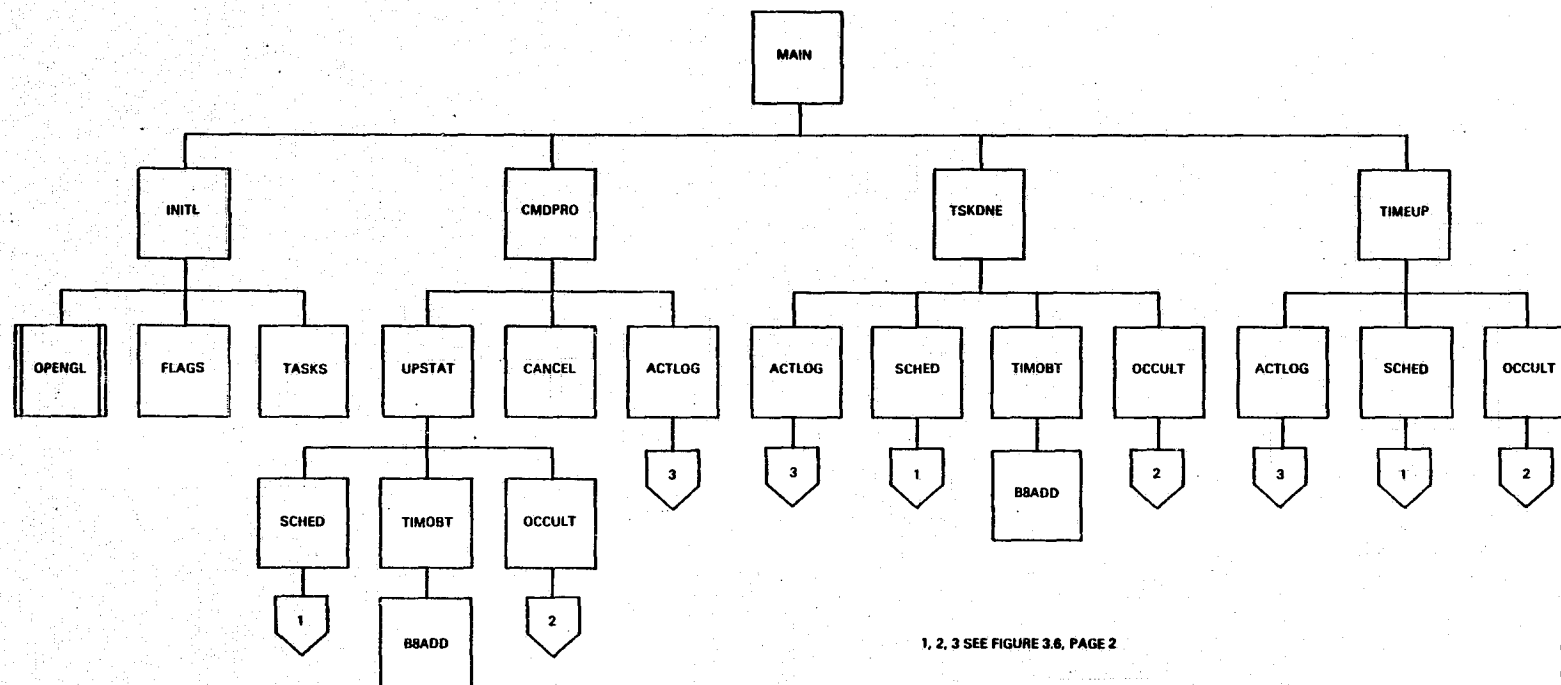
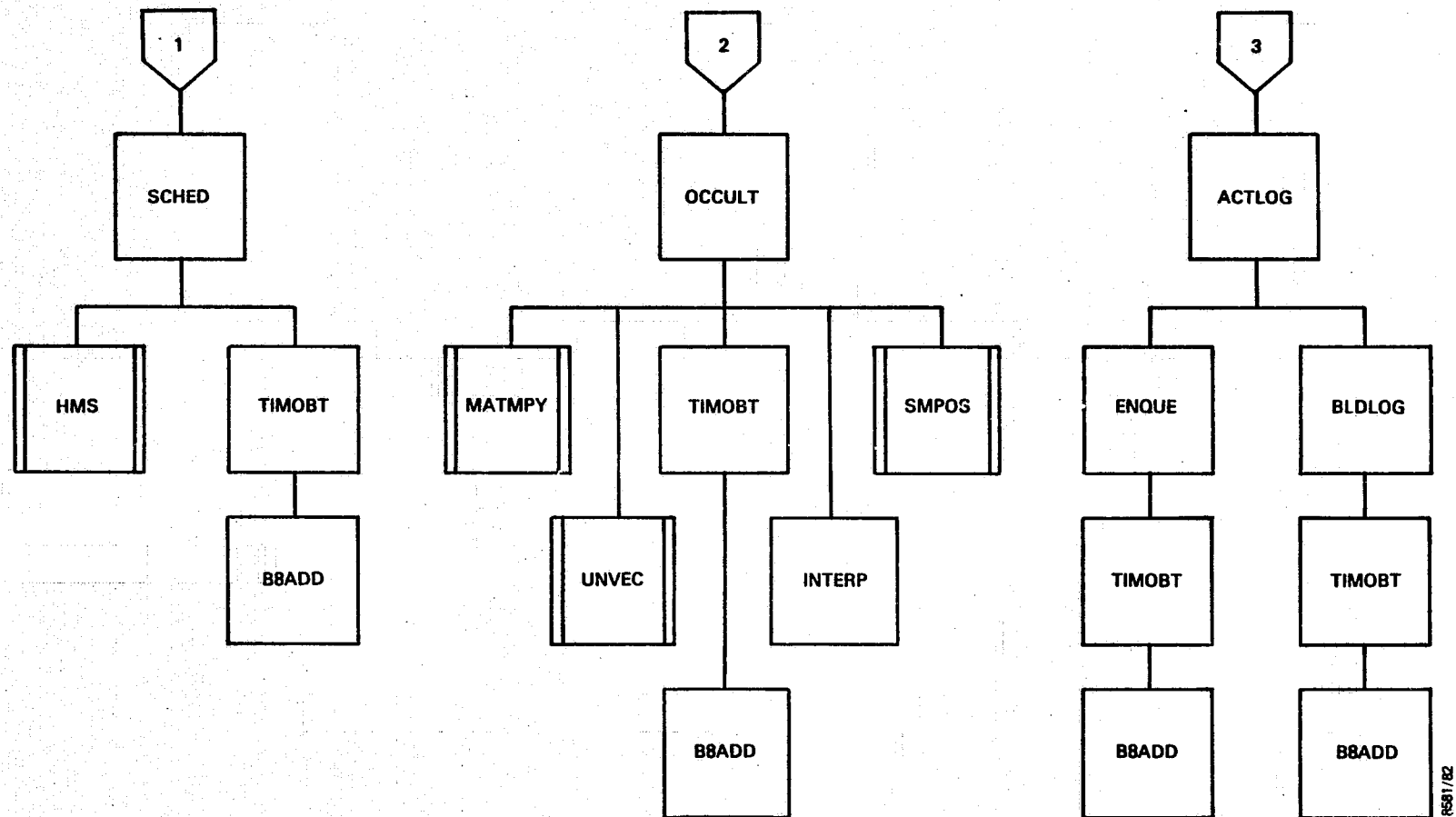


Figure 3-6. EXEC Process Baseline Diagram (1 of 2)

3-13



RSB/82

Figure 3-6. EXEC Process Baseline Diagram (2 of 2)

3.2 SENSOR DATA COLLECTION PROCESS

This section describes the sensor data collection (SENSIN) process.

3.2.1 INPUT/OUTPUT

The following subsections describe the input and output parameters for the SENSIN process. Figure 3-7 illustrates the external interfaces.

3.2.1.1 Input

The following items are required as input:

- Gyro data from six channels
 - Gyro counts
 - Power-on/off flag
 - Low/high-gain (rate) flag
- Star camera data from two cameras
 - H and V counts
 - Star camera temperature
 - Star intensity
 - Star presence flag
- Current time in seconds since September 1, 1957, 0 hours Universal Time (UT)

3.2.1.2 Output

The following items are output from the SENSIN process:

- Time-tagged gyro data
 - Time in seconds since September 1, 1957, 0 hours, UT
 - Number of observations

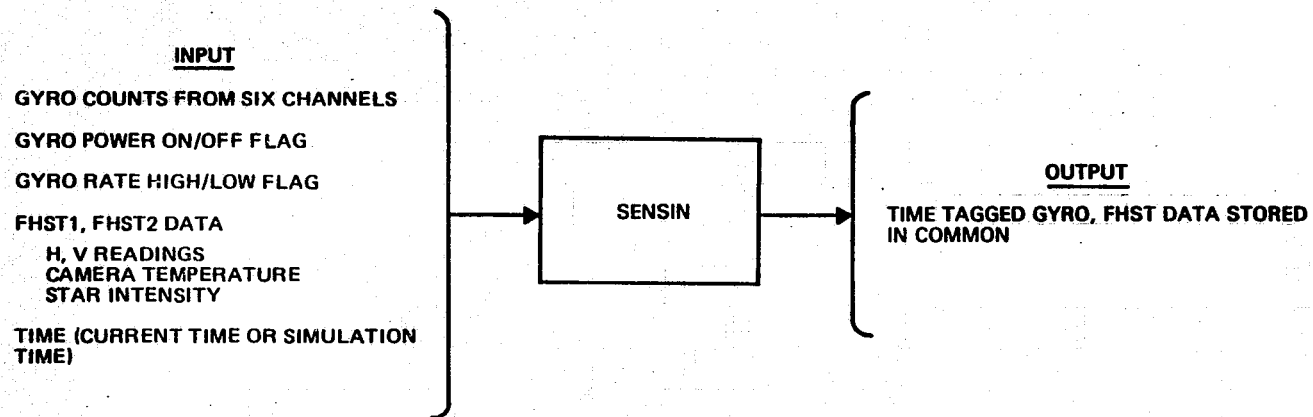


Figure 3-7. SENSIN Process External Interfaces

- Time-tagged star camera data
 - H and V counts, star camera temperature, and star intensity
 - Star presence flag
 - Number of observations

3.2.2 PROCESSING

The SENSIN process will be nominally invoked every 64 milliseconds to read gyro data and every 100 milliseconds to read star camera data. SENSIN will perform minimum required processing at a high priority.

Gyro data are stored in COMMON /SENSOR/ with the current time. Star tracker data and the associated time are stored in COMMON /SENSOR/ without any decoding. In both cases, the counters for gyro and star camera observations are updated. The gyro data and star tracker data buffers are used in a wraparound manner. That is, SENSIN starts filling data from the top of the buffer when the buffer is filled. SENSIN maintains the start and end pointers for data arrays. GYPROC and STRACK processes use the available data when invoked, and then reset the start pointer for the next cycle.

The times used for gyro and star camera data may be in error by 64 milliseconds and 50 milliseconds (maximum), respectively. The error in the computed attitude will not increase significantly because of these time errors.

Table 3-2 contains the descriptions of SENSIN process components to be used with Figures 3-8 and 3-9. Figure 3-8 illustrates data flow within the process and Figure 3-9 illustrates the relationship among process components.

Table 3-2. SENSIN Process Component Descriptions

<u>Component</u>	<u>Description</u>
SENSIN	Driver for high-frequency sensor data collection process
SIGYRO	Collects and stores gyro data
SIFHST	Collects and stores star tracker (FHST) data
TIMOB	Returns current simulation time in desired format
B8ADD	Adds two quadwords
OPENGL	User open routine for temporary global COMMON areas
/RIU/	Global COMMON area from which raw sensor data is obtained

ORIGINAL PAGE IS
OF POOR QUALITY

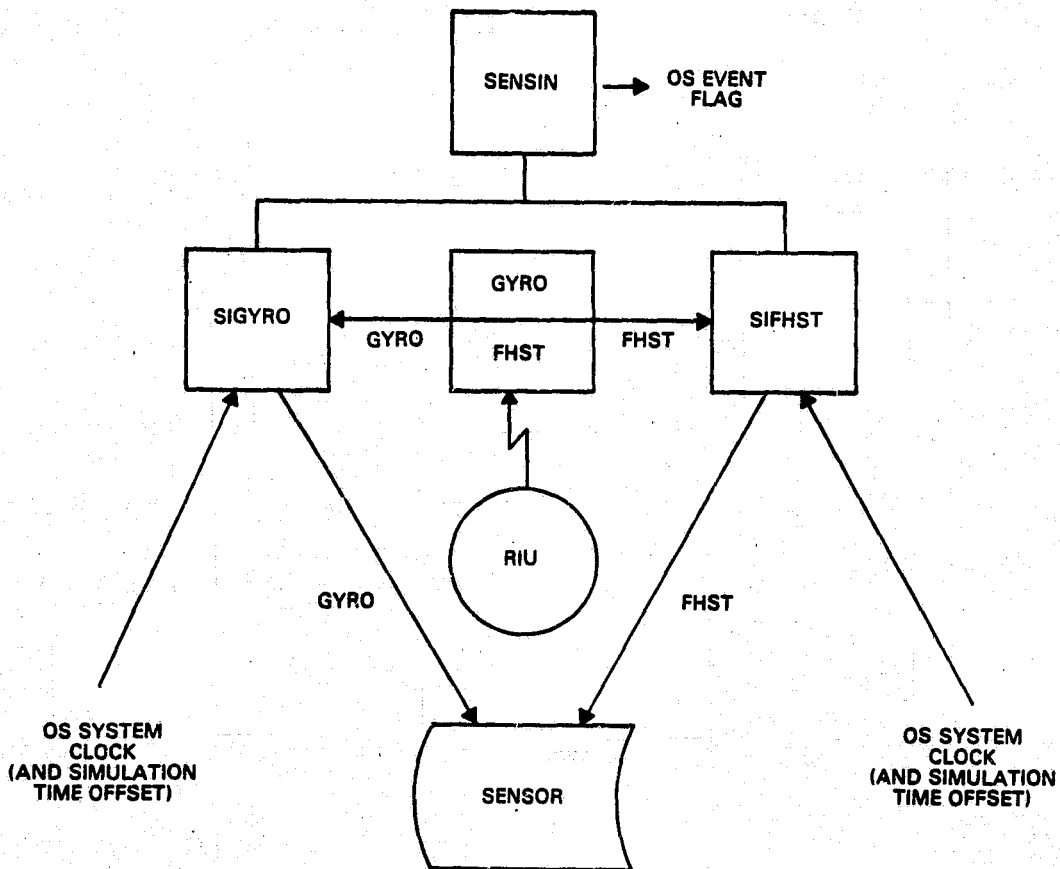


Figure 3-8. SENSIN Process Data Flow

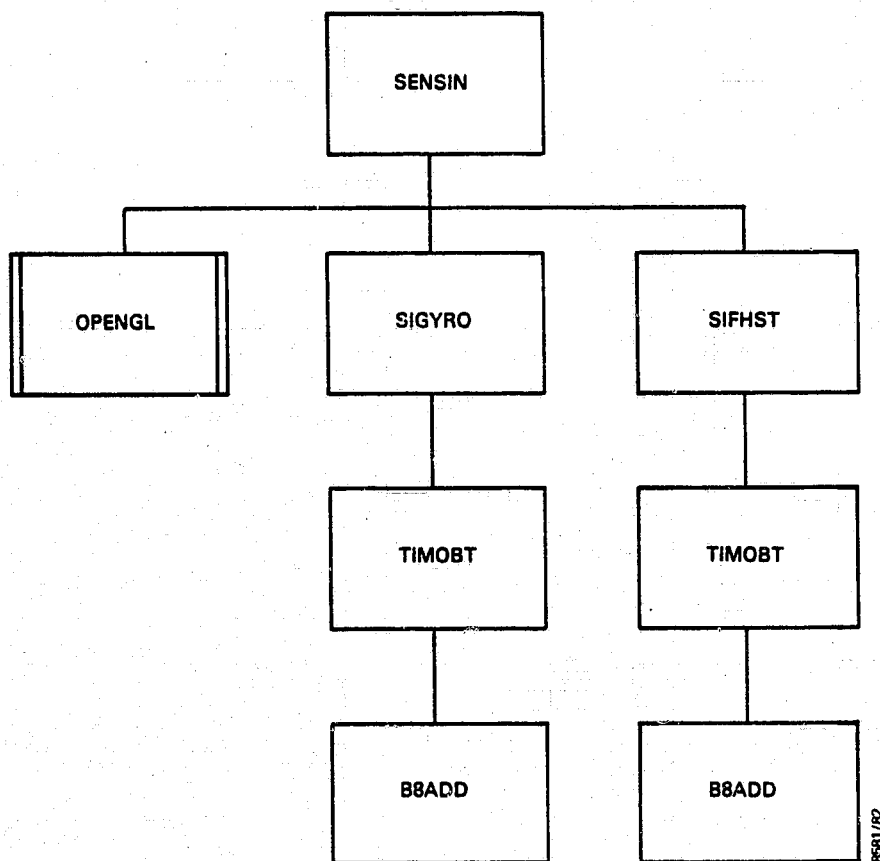


Figure 3-9. SENSIN Process Baseline Diagram

3.3 INPUT PROCESS

This section describes the input data processing (INPUT) function.

3.3.1 INPUT/OUTPUT

The following subsections describe the input and output parameters for the INPUT process. Figure 3-10 illustrates the external interfaces.

3.3.1.1 Input

The following items are required as input:

- Uplink
 - Commands: RESUME, HALT, STOP, START, and SET CLOCK
 - Data to update global COMMON /STATIC/
- Spacecraft ephemeris (computed by the onboard computer)

3.3.1.2 Output

The following items are output from the INPUT process:

- Decoded commands for the EXEC process
- Validated uplinked data stored in global COMMON /STATIC/
- Validated spacecraft ephemeris
- Status in global COMMON /GLOBAL/
- Error messages in global COMMON /OUTBUF/

3.3.2 PROCESSING

The INPUT process consists of two major subsystems: the data capture (DCAP) subsystem and the full input data processing (INPUT) subsystem. Each subsystem is called (as a subroutine) by the INPUT process driver, DCAPIN. DCAPIN

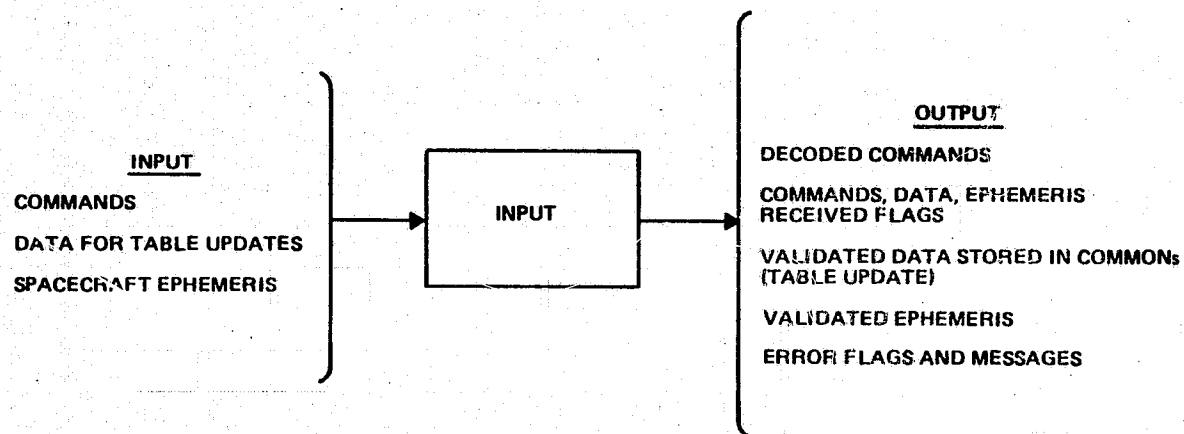


Figure 3-10. INPUT Process External Interfaces

itself is invoked in one of two ways: (1) when a read operation of input data has just been completed or (2) when the EXEC process indicates that queued uplink data and/or stored ephemeris data must be validated. In both cases, a unique event flag will have been set. DCAPIN invokes the DCAP subsystem for case 1 and the INPUT subsystem for case 2.

The purpose of the DCAP subsystem is to handle quickly the preliminary validation and processing of input data from the main spacecraft computer (sensor data is obtained by the SENSIN process). Primarily, the DCAP subsystem

- Checks synchronization byte(s)
- Saves an input command in global COMMON /GLOBAL/ for the EXEC process
- Stores ephemeris data in global COMMON /FLIGHT/
- Queues uplinked data for further validation and processing later

The EXEC process is activated when a command is present or an error has been detected.

The INPUT subsystem performs full validation of uplinked data and spacecraft ephemeris data. Uplinked data are validated according to header information; ephemeris data are validated according to computed magnitude constraints. The INPUT process finishes its processing of uplinked data by storing the data in the global COMMON /STATIC/. Upon completion of its processing in all cases, the INPUT process reports its status to the EXEC process and activates it before completing execution.

Table 3-3 contains the descriptions of INPUT process components to be used with Figures 3-11 and 3-12. Figure 3-11 illustrates data flow within the process and Figure 3-12 illustrates the relationship among process components.

Table 3-3. INPUT Process Component Descriptions

<u>Component</u>	<u>Description</u>
DCAPIN	Driver for input data processing function; invoked by operating system when commands or ephemeris data are received, or invoked by the executive at end of a major cycle to complete validation of ephemeris data, table parameters, and so on
DCAP	Driver for data capture subsystem; DCAPIN invokes DCAP subsystem when a command or an ephemeris read occurs--input is processed with minimum, preliminary validation--alerts EXEC process immediately about ground commands
STOEPH	Stores spacecraft ephemeris
QLONG	Queues uplinked data in full, 256-byte records for later validation
INPUT	Driver for input data subsystem; validates and stores data acquired by data capture subsystem
INEPH	Validates spacecraft ephemeris
INFULL	Validates and stores long block (256-byte) uplinked data
HEADER	Validates record according to header information
STCDAT	Stores control parameters in global COMMON /STATIC/
/DCAPIP/	Local COMMON area used to queue uplinked data for later validation
/MSGBUF/	Local COMMON area containing buffer into which data are initially read

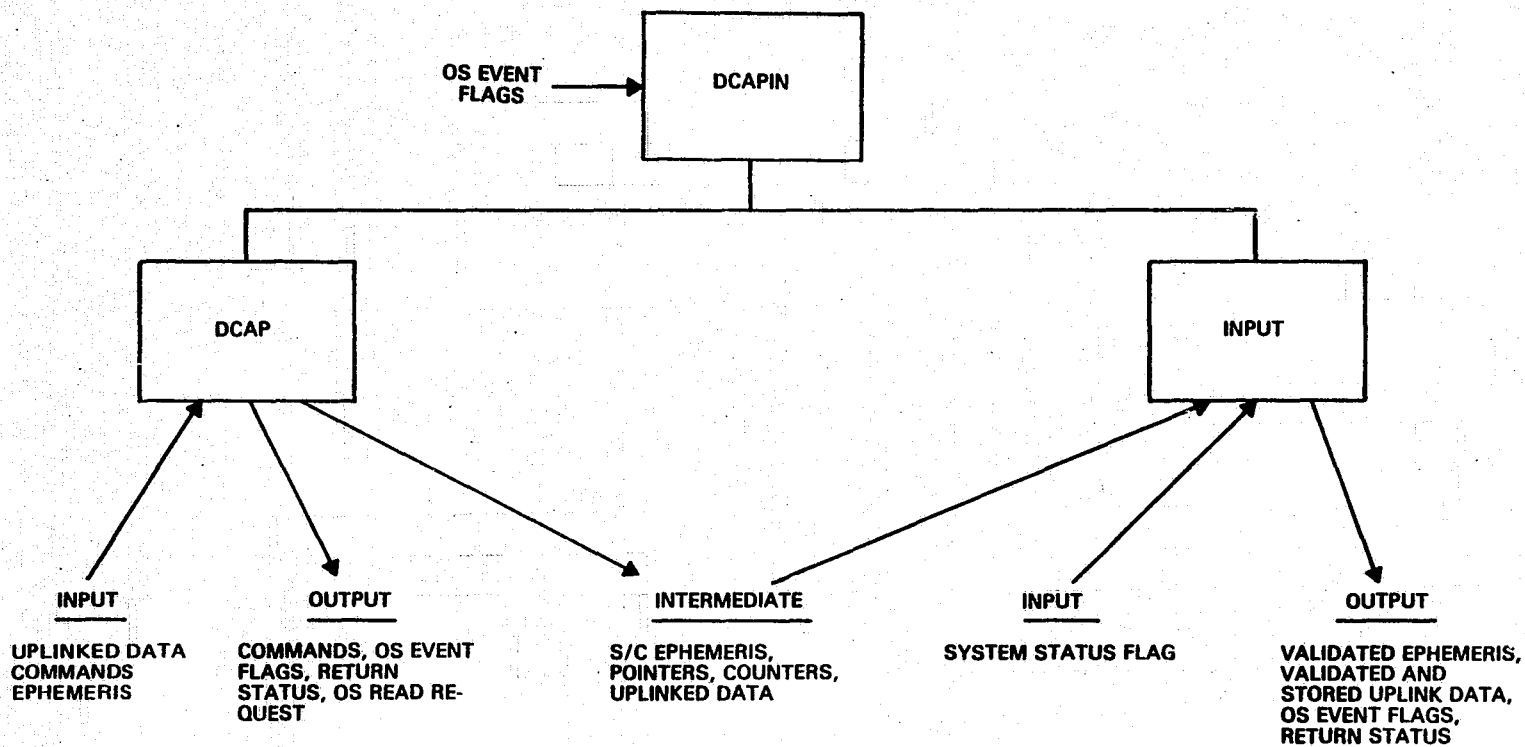


Figure 3-11. INPUT Process Data Flow

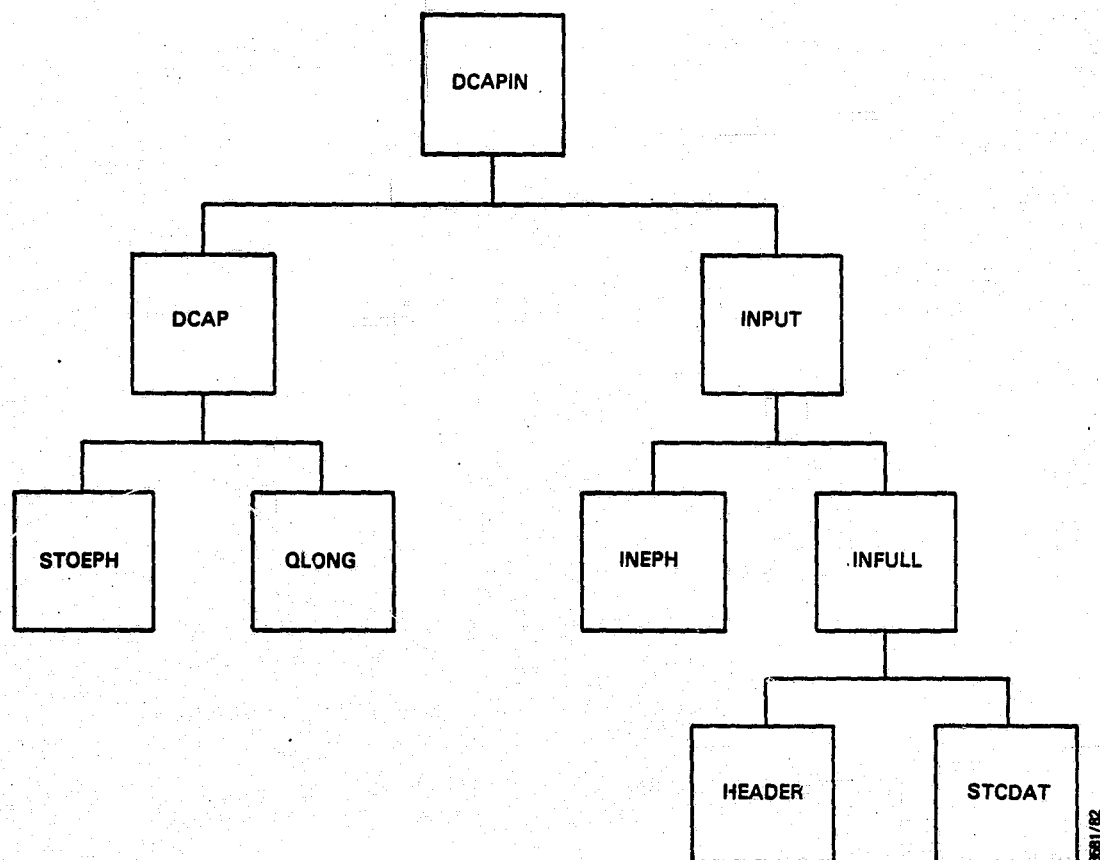


Figure 3-12. INPUT Process Baseline Diagram

3.4 OUTPUT PROCESS

This section describes the output data processing (OUTPUT) function.

3.4.1 INPUT/OUTPUT

The following subsections describe the input and output parameters for the OUTPUT process. Figure 3-13 illustrates the external interfaces.

3.4.1.1 Input

The following items are required as input:

- Annotation data
 - Quaternion
 - Associated time for quaternion
- Activity log data
 - Current time
 - Time of START command
 - Number of SET CLOCK commands received
 - Total number of state updates attempted
 - Total number of successful state updates
 - Total number of spacecraft ephemeris requests
 - Total number of gyro data saturations
 - Total number of identified stars
- Raw sensor data
 - Gyro data from six channels: gyro counts, power-on/off flag, and low/high-gain flag
 - Star data from two star cameras: H and V coordinate counts, star camera temperature, star intensity, power-on/off flag, and star presence flag

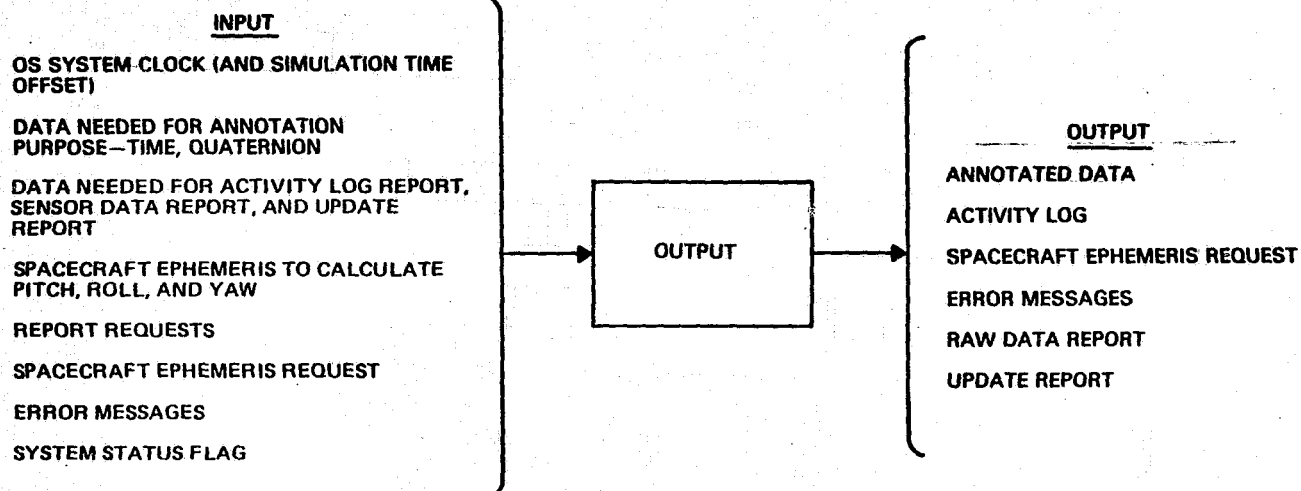


Figure 3-13. OUTPUT Process External Interfaces

- State update data
 - Quaternion
 - Associated time
 - Gyro drift biases
 - Gyro configuration
 - State covariance matrix
 - Kalman gain matrix
 - Time between first and last identified stars
 - Number of star vectors formed
 - Number of stars identified
- Spacecraft ephemeris (for inertial-to-orbital coordinate transformation)
- Report request indicators
 - Activity log output request
 - Raw sensor data output request
 - State update output request
 - Annotation data output request
- Queue of high-priority, critical error messages
 - Pointers for error queues
- Spacecraft ephemeris request data
 - Request indicator
 - Start time
 - Number of points
 - Time interval between points (seconds)

3.4.1.2 Output

The following items are output from the OUTPUT process:

- Annotation report
 - Quaternion
 - Euler angles
 - Associated time

- Activity log (formatted report of items detailed in Section 3.4.1.1)
- Raw sensor data report (see Section 3.4.1.1)
- State update report (see Section 3.4.1.1)
- Critical error messages
- Spacecraft ephemeris request

3.4.2 PROCESSING

The OUTPUT process is activated periodically by the EXEC process to output annotated data, the activity log, and ephemeris requests. It is also activated to report errors when the other processes indicate errors. The OUTPUT process communicates with the OBC; the OBC transmits AADS communications to the ground.

When an annotation data report is to be output, the spacecraft ephemeris is used to transform the spacecraft attitude in quaternion form to the pitch, roll, and yaw. The annotation report contains both the quaternion and the pitch, roll, and yaw angles.

When critical error messages are present, they are output. An ephemeris request is generated and transmitted when required. The indicators for the activity log, raw sensor data, and state update reports are checked; these reports are formatted and output as requested. The OUTPUT process activates the EXEC process after completing execution, setting its status in the status indicator array (SSTATE).

Table 3-4 contains the descriptions of OUTPUT process components to be used with Figures 3-14 and 3-15. Figure 3-14 illustrates data flow within the process and Figure 3-15 illustrates the relationship among process components.

Table 3-4. OUTPUT Process Component Descriptions

<u>Component</u>	<u>Description</u>
OUTPUT	Drives output data processing function; calls subroutines until all requested reports and queued messages are transmitted to the OBC
OBOUT	Sends data annotation reports and spacecraft ephemeris requests
HIPRI	Sends high-priority (critical error) messages to ground through the OBC(OSS)
REPORT	Formats and sends reports for downlinking via OBC(OSS)
RPY	Transforms a quaternion to roll, pitch, and yaw
RAWDAT	Formats and sends raw sensor data report
UPDRPT	Formats and sends update reports
OUTACT	Formats and sends activity log
TIMOB	Returns current simulation time in desired format
B8ADD	Adds two quadwords
INTERP	Performs a six-point interpolation of ephemeris

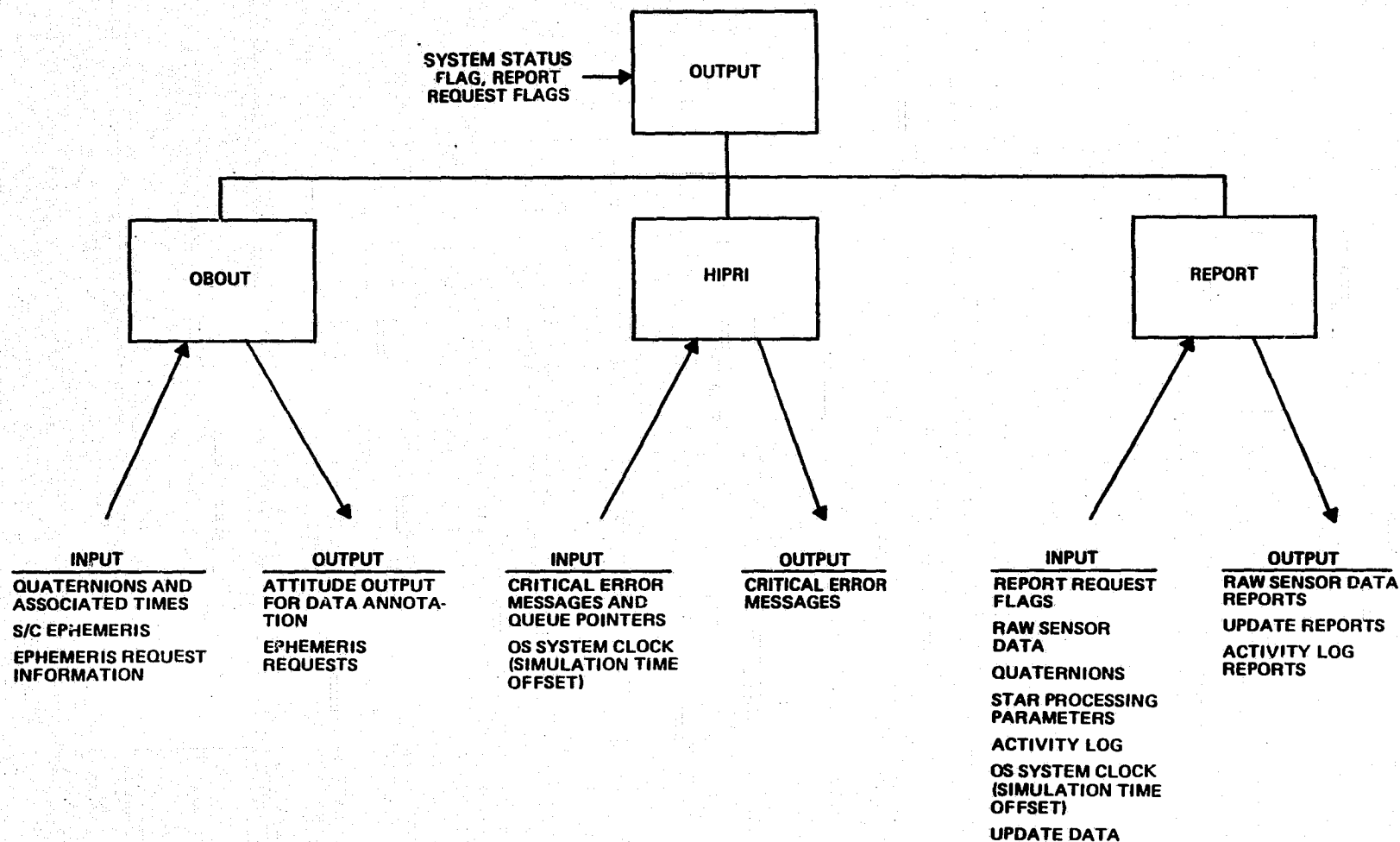


Figure 3-14. OUTPUT Process Data Flow

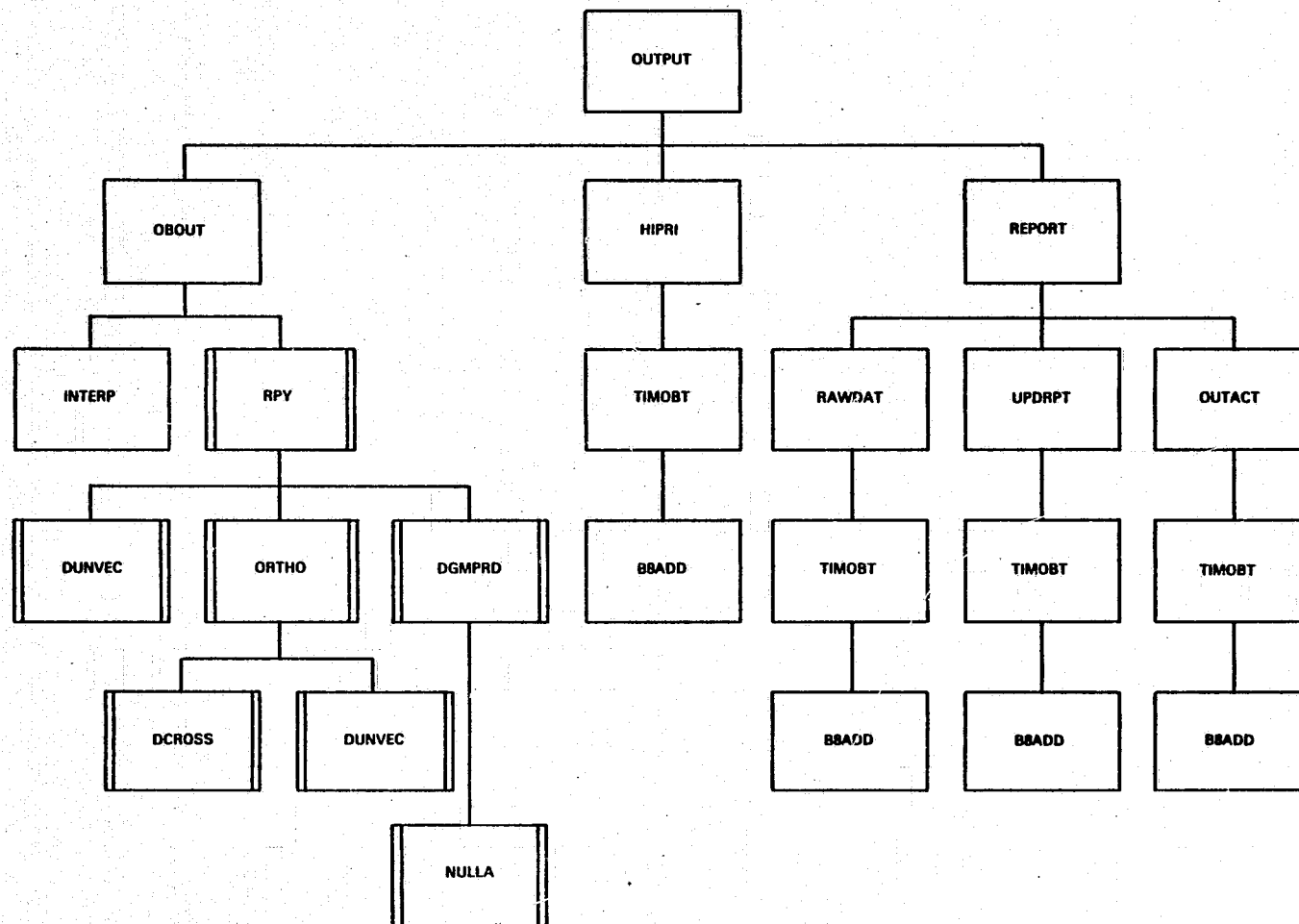


Figure 3-15. OUTPUT Process Baseline Diagram

3.5 GYRO DATA PROCESS

This section describes the gyro data processing (GYPROC) function.

3.5.1 INPUT/OUTPUT

The following subsections describe the input and output parameters for the GYPROC process. Figure 3-16 illustrates the external interfaces.

3.5.1.1 Input

The following items are required as input:

- Gyro data from six channels
 - Gyro counts
 - Power-on/off flag
 - Low/high-gain (rate) flag
- Times associated with the raw gyro data (time in seconds since September 1, 1957, 0 hours UT)
- Pointers to indicate the beginning and end of gyro data
- Control parameters
 - Gyro configuration
 - Maximum gyro counts
 - Upper/lower limits for gyro counts
- Conversion factors to convert gyro counts to angles
- Gyro alignment matrices for all possible channel combinations
- Flag set by the EXEC process to indicate initialization or a normal processing mode

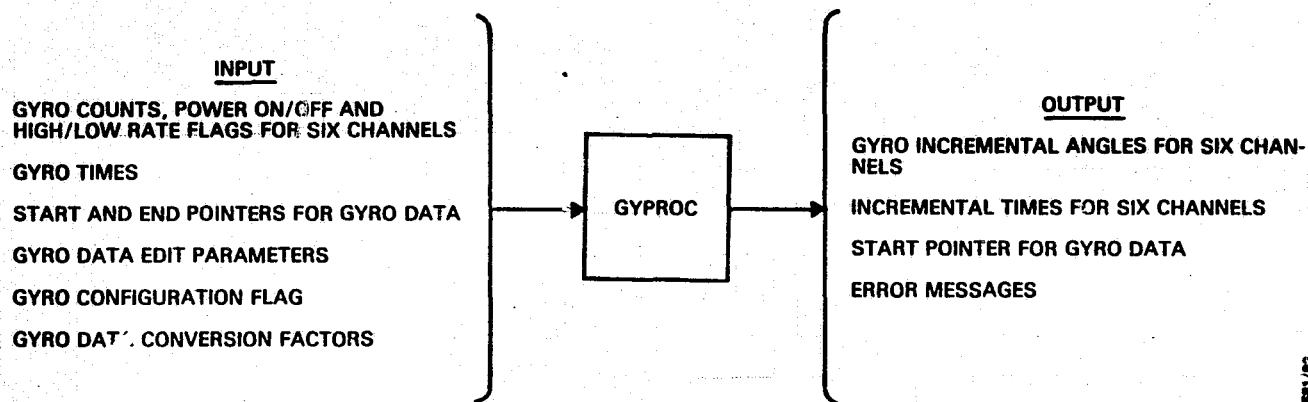


Figure 3-16. GYPROC Process External Interfaces

3.5.1.2 Output

The following items are output from the GYPROC process:

- Incremental angles for all channels that are on
- Incremental times for all channels that are on
- Maneuver indicator
- Error messages

3.5.2 PROCESSING

The GYPROC process is nominally invoked every 512 milliseconds to validate gyro data and to compute gyro incremental angles. It processes the data collected since the last time it was invoked. The time period to invoke the GYPROC process can be varied by ground command, but the upper and lower limits on the period are expected to be 1,024 milliseconds and 256 milliseconds, respectively.

The GYPROC process performs initialization depending on the flag set by the EXEC process. Power-on/off flags, low/high-rate flags, and gyro counts are extracted from the gyro data. Primary and secondary gyro channels are selected based on the gyro configuration specified. The raw data are validated for rollover, saturation, and successive difference tolerances. Error flags are set if the data fails validation checks. Gyro incremental angles and incremental times are computed for both primary and secondary configurations. They are stored in the global COMMON area /SENSOR/ for eventual use by the state propagation (PROPAG) process. Data pointers are updated in global COMMON /SENSOR/ to indicate that the data has been processed by GYPROC.

The GYPROC process sets the maneuver indicator for the processes following GYPROC execution when the rate mode flag indicates that the gyros are in the high-rate mode. When the spacecraft is being maneuvered, the GYPROC process may be automatically executed at a higher frequency to limit the

error in attitude computation. Since the star data processing functions are not invoked during spacecraft maneuvers, a higher execution frequency is possible.

Table 3-5 contains the descriptions of GYPROC process components to be used with Figures 3-17 and 3-18. Figure 3-17 illustrates data flow within the process and Figure 3-18 illustrates the relationship among process components.

Table 3-5. GYPROC Process Component Descriptions

<u>Component</u>	<u>Description</u>
GYPROC	Driver for gyro data processing function
GYROED	Gyro data validation driver; checks for spacecraft maneuver by change in gyro rate flag
CHECK	Checks for rollover, saturation, and glitches in gyro data
GYROUT	Converts gyro counts to angles; stores sums of incremental angles and times for all channels
SELECT	Selects primary and backup gyro channel combination from the gyro configurations indicator and selects corresponding alignment matrices
HEAD	Stores error number and current time in output buffer queue
MESS	Stores error message in output buffer queue
TCON21	Converts time from seconds since September 1, 1957, to calendar format (YYMMDD.HHMMSS)
DATVAX	Computes calendar date corresponding to a given Julian day
/PROC/	Local COMMON area

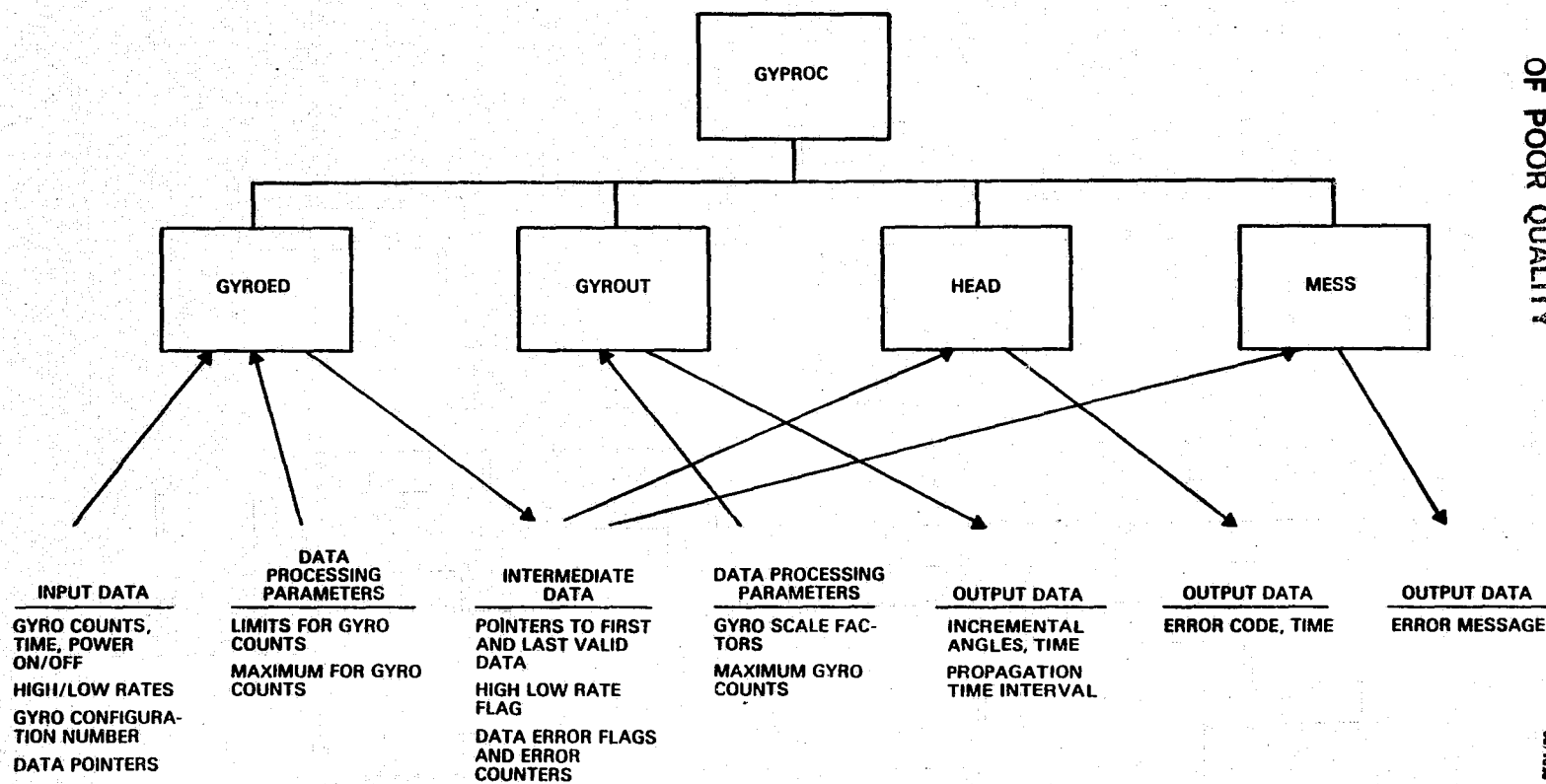


Figure 3-17. GYPROC Process Data Flow

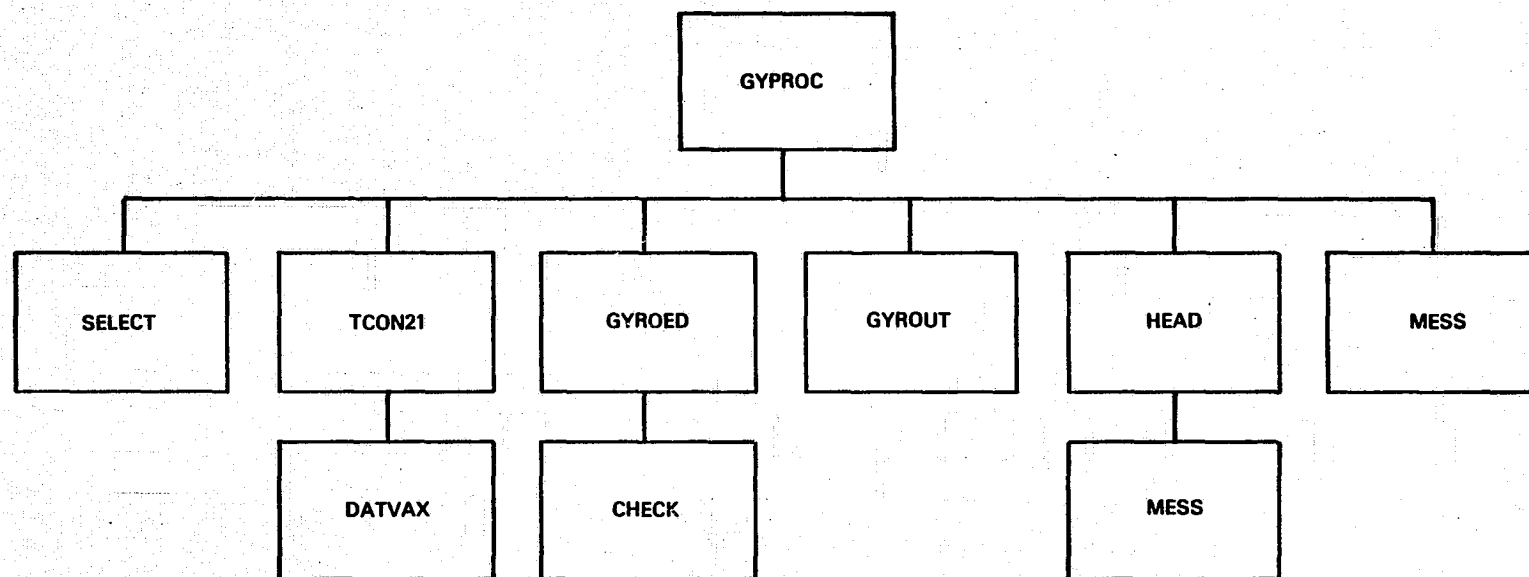


Figure 3-18. GYPROC Process Baseline Diagram

3.6 STATE PROPAGATION PROCESS

This section describes the state propagation (PROPAG) process.

3.6.1 INPUT/OUTPUT

The following subsections describe the input and output parameters for the PROPAG process. Figure 3-19 illustrates the external interfaces.

3.6.1.1 Input

The following items are required as input:

- Gyro configuration number
- Alignment matrices for all gyro configurations
- Incremental angles and times for all six channels
- Gyro biases (drift rates)
- Tolerances for redundancy between primary and backup channel combinations
- Quaternion and state covariance matrix
- Gyro rate noise (float torque) and gyro rate-rate noise (ramp noise)
- Flag set by the EXEC process to indicate initialization or normal processing mode

3.6.1.2 Output

The following items are output from the PROPAG process:

- Propagated quaternion
- Propagated state covariance matrix
- Spacecraft angular velocity vector in the spacecraft frame

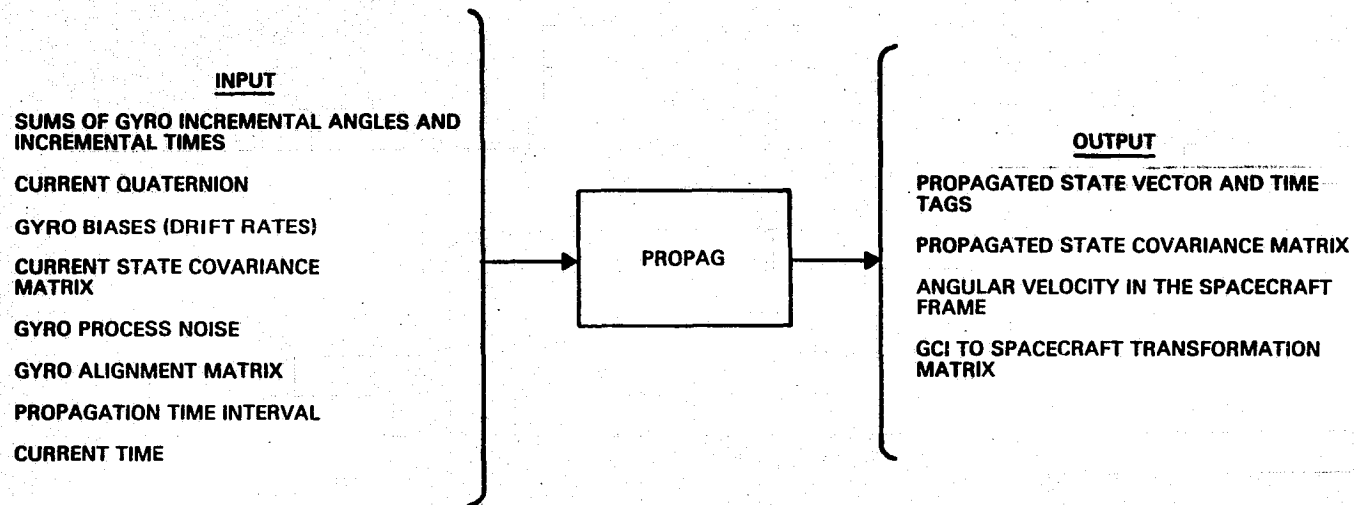


Figure 3-19. PROPAG Process External Interfaces

- GCI to spacecraft transformation matrix
- Error messages

3.6.2 PROCESSING

The PROPAG process is invoked after the GYPROC process is invoked N times; N is set by ground command and is nominally set to 1. Thus, PROPAG propagates the quaternion and the state covariance matrix nominally every 512 milliseconds.

Body rates in the gyro coordinate frame are computed from the valid incremental angles and the incremental times for both the primary and backup channels. PROPAG then performs a redundancy test to check the rate differences between the primary and backup channels and sets appropriate error flags if the validation check fails.

The gyro rates are rotated from the gyro coordinate frame to the spacecraft (body) coordinate frame. Drift biases are eliminated. These body rates and the propagation time interval are used to compute the body rotation matrix, state transition matrix, and the covariance transition matrix.

The process noise matrix (consisting of the gyro noise variances), the body rotation matrix, and the propagation time interval are used to compute the noise covariance matrix.

The spacecraft attitude (in the quaternion form) is propagated by using the current quaternion and the state transition matrix. The state covariance matrix is propagated by using the current state covariance matrix, state covariance transition matrix, and noise covariance matrix. The propagated quaternion and the state covariance matrix are saved in global COMMON /STATE/. PROPAG propagates the quaternion part of the state vector and does not propagate the gyro biases in the state vector.

Table 3-6 contains the descriptions of PROPAG process components to be used with Figures 3-20 and 3-21. Figure 3-20 illustrates data flow within the process and Figure 3-21 illustrates the relationship among process components.

Table 3-6. PROPAG Process Component Descriptions

<u>Component</u>	<u>Description</u>
PROPAG	Driver for state propagation process
BRATE	Eliminates gyro biases and computes body-rate vector and body-rotation skew symmetric matrix
PRINTR	Computes state transition matrix (3x3) for three-quaternion propagation; computes state covariance transition matrix and noise covariance matrix
QPROP	Uses current quaternion and state transition matrix to propagate quaternion
COPROP	Uses current state covariance matrix, state covariance transition matrix, and noise covariance matrix to propagate state covariance matrix
PROPIN	Initializes PROPAG process
/PROPG/	Local COMMON area
SGMPRD	Computes product of two matrices
SGMTRA	Computes transpose of the matrix
SDIAG	Computes sum of square matrix and identity matrix
SMTADD	Computes sum of two matrices
SMLMAT	Computes product of a scalar and matrix
SDTPRD	Computes dot product of two vectors
SCOPY	Copies one two-dimensional area to other two-dimensional areas
SNULLA	Stores zero in all elements of a matrix
HEAD	Stores error number and current time in error buffer
MESS	Stores error message in error buffer
TCON21	Converts time in seconds since September 1, 1957, to calendar format (YYMMDD.HHMMSS)
DATVAX	Computes calendar date corresponding to a given Julian day

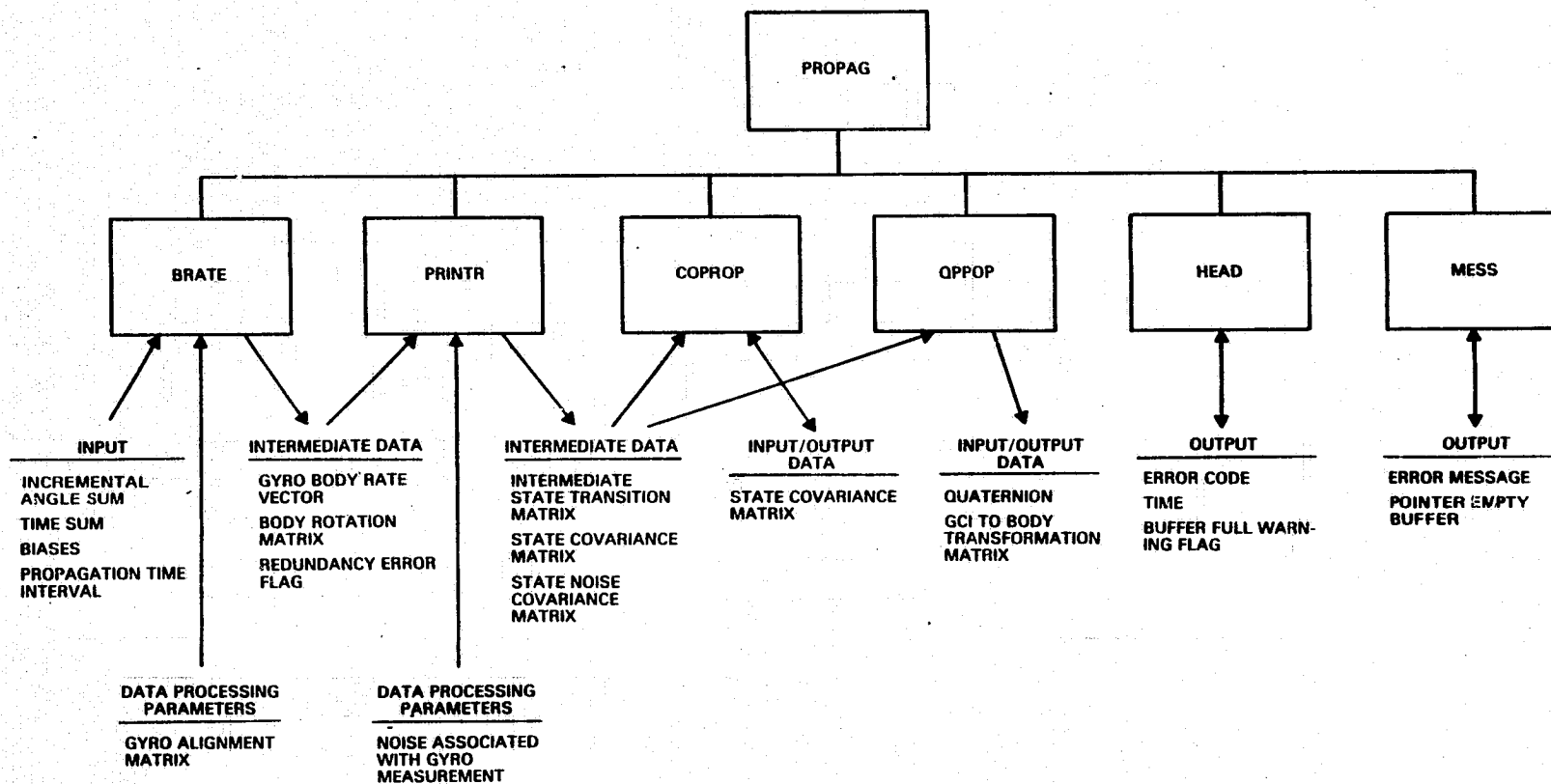


Figure 3-20. PROPAG Process Data Flow

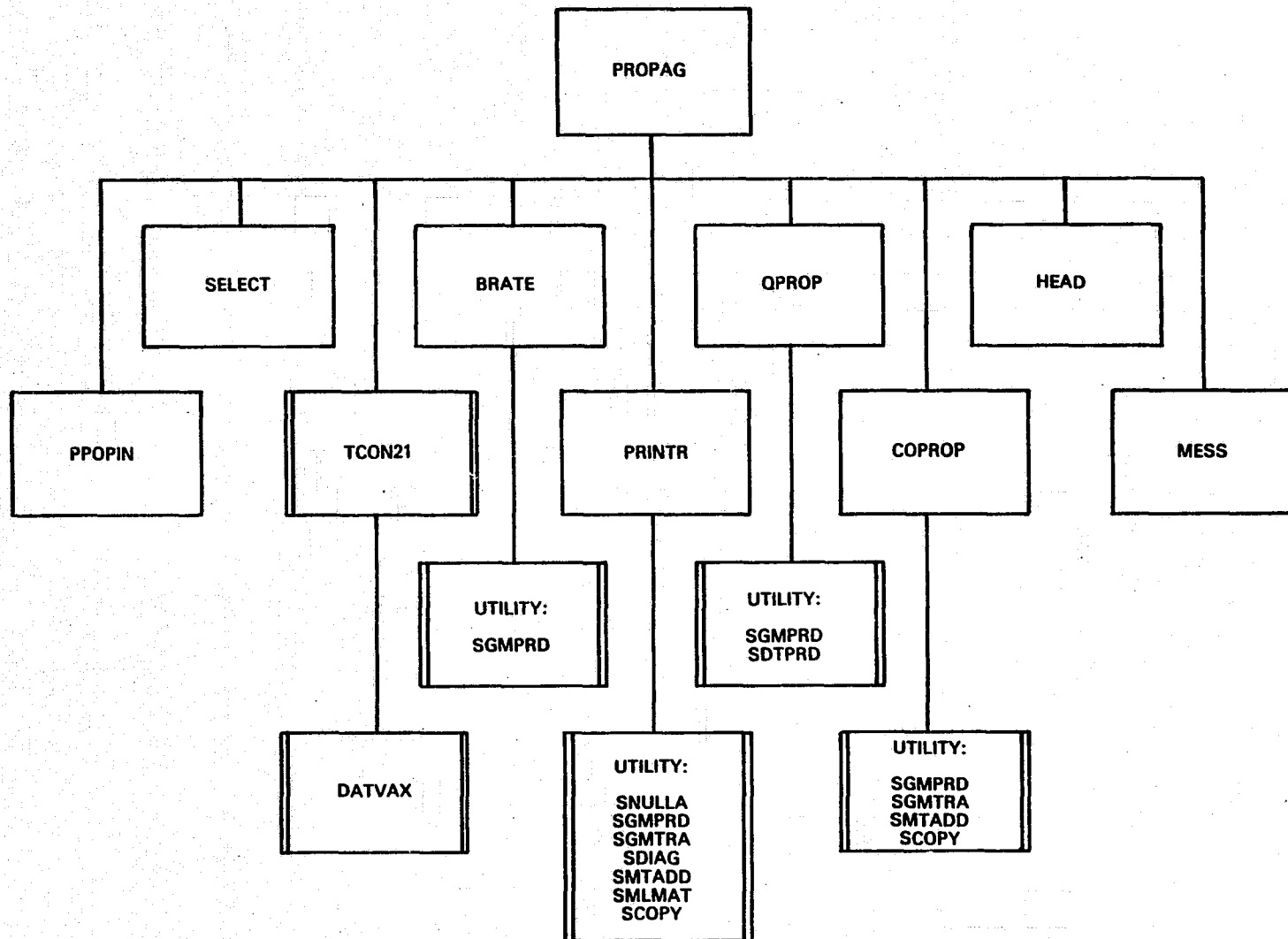


Figure 3-21. PROPAG Process Baseline Diagram

3.7 STAR DATA PROCESS

This section describes the star data processing (STRACK) function.

3.7.1 INPUT/OUTPUT

The following subsections describe the input and output parameters for the STRACK process. Figure 3-22 illustrates the external interfaces.

3.7.1.1 Input

The following items are required as input:

- Raw star tracker data
 - H and V counts
 - Times
 - Star intensity (counts)
 - Star tracker temperature (counts)
 - Star presence flag
 - Star tracker power-on/off flag
- Fresh raw data location indicators
- Invalid data timespan tolerance parameters
- H, V, intensity, and temperature editing parameters
- H and V calibration parameters to correct for intensity and temperature effects
- H, V, intensity, and temperature conversion parameters
- Track point grouping parameters
- GCI to spacecraft transformation matrices
- Star tracker alignment matrices
- Spacecraft angular velocity vectors in the body frame and times
- Attitude quaternions and times

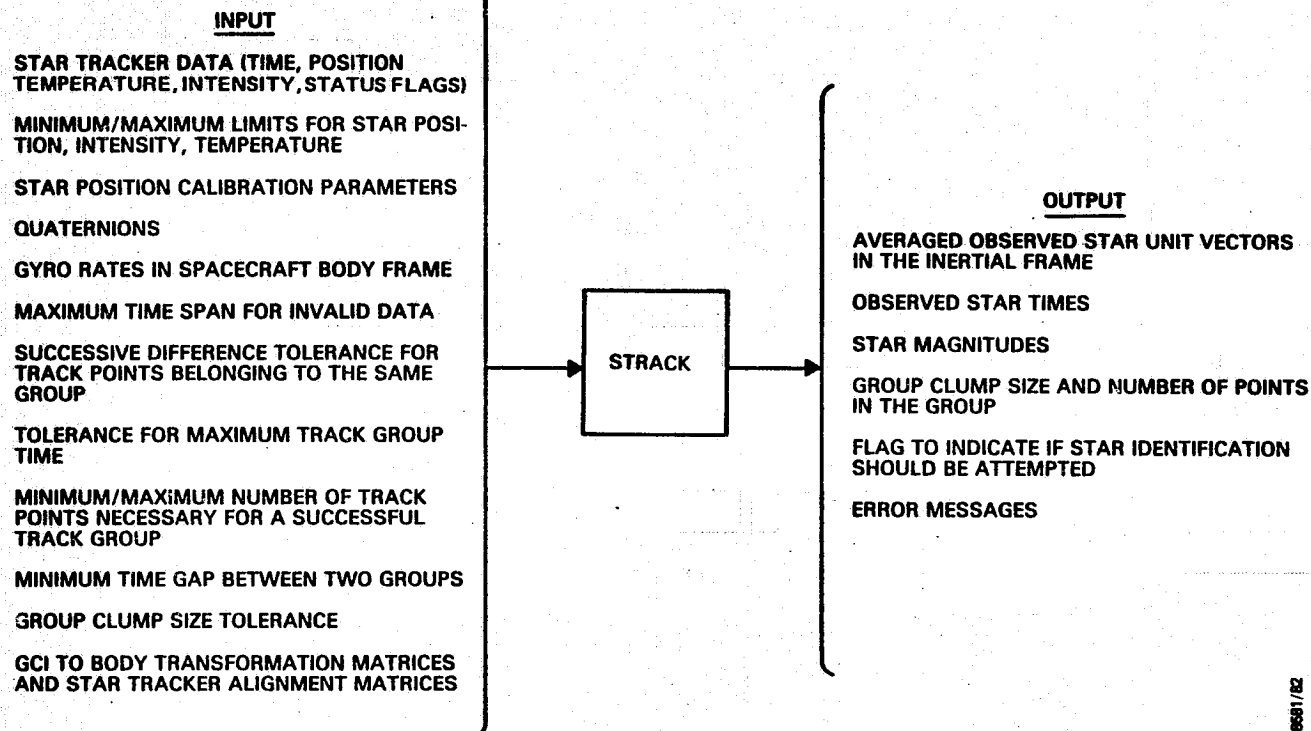


Figure 3-22. STRACK Process External Interfaces

- Intensity-to-magnitude conversion parameters
- Minimum number of groups necessary before attempting star identification
- Current attitude uncertainty and attitude uncertainty criterion for selecting star identification method
- Flag set by the EXEC process to indicate initialization or a normal processing mode

3.7.1.2 Output

The following items are output from the STRACK process:

- Track point group average data
 - Times
 - Unit vectors in geocentric inertial (GCI) coordinate frame
 - Number of track points in each group average
 - Star magnitudes
- Error messages
- Flag to indicate sufficient tracker groups to attempt star identification

3.7.2 PROCESSING

The STRACK process uses raw star tracker data prepared by the SENSIN process. It acquires all the fresh raw star tracker data according to fresh data pointers. The pointers are updated at the end of processing to reflect the current status of stored raw data.

Each raw star tracker data sample includes data time, H and V counts, star tracker temperature, star intensity, star presence flag, and star tracker power-on/off flag. Data samples with flags that indicate power on and star presence

are subjected to limit checking. Temperature and intensity are converted from counts to volts before being checked. Data samples with flags that indicate power off and no star presence, or that fail limit checking are considered invalid and are discarded. A warning message is issued when consecutive invalid data timespans exceed a tolerance.

Validated data are then separated into groups. Presumably, each group contains tracking data from a single star. The group-forming criteria include the maximum difference between two consecutive H and V counts, maximum and minimum number of tracking points in a group, maximum group time duration, and minimum time separation between two groups. H and V counts are calibrated for temperature and intensity effects and are converted into angles.

The next step is to compute track point unit vectors and to perform group averages. The track point unit vectors in the current tracker coordinate frame are computed from H and V angles. They are rotated to the tracker coordinate frame at the nearest gyro-rate times and then to the GCI coordinate frame using gyro-propagated attitude at the gyro-rate times. Track point unit vectors in the GCI coordinate frame and corresponding star intensities in each group are averaged. The group clump sizes are then computed. The averaged star intensities are converted to magnitudes.

If the number of track points used in a group average exceed minimum requirements and the group clump size is within tolerance, the group is considered acceptable for star identification. Current attitude uncertainty is computed from the first three diagonal elements of the state covariance matrix. The attitude uncertainty determines the number of groups required to be formed before attempting star identification and the star identification method to be used. When the number of the acceptable groups satisfies minimum

requirements for the identification method selected, a flag is set to inform the EXEC to invoke the star identification (STARID) process.

Table 3-7 contains the descriptions of STRACK process components to be used with Figures 3-23 and 3-24. Figure 3-23 illustrates data flow within the process and Figure 3-24 illustrates the relationship among process components.

Table 3-7. STRACK Process Component Descriptions

<u>Component</u>	<u>Description</u>
STRACK	Driver of star data processing function
INITST	Initializes star tracker data processing working parameters
STREAD	Edits raw star tracker data, forms track group, and calibrates raw data
MAGN	Edits star tracker data
TGROUP	Forms track group
CALIB	Driver to calibrate raw star track data for temperature and intensity correction
SYNVEC	Computes the track point unit vector; synchronizes unit vector to nearest gyro-propagated quaternion time; rotates unit vector to GCI coordinate frame
SGMPRD	Multiplies two matrices.
AVGVEC	Computes average of track point unit vectors in a track group; determines star identification method and sets a flag when there are enough groups to attempt star identification
FCALIB	Calibrates raw star tracker data for temperature and intensity correction
TCON21	Converts time in seconds from September 1, 1957, to the calendar format (YYMMDD.HHMMSS)
ANGLED	Computes angle between two unit vectors
UNVEC	Unitizes a vector
DATVAX	Computes calendar date corresponding to a given Julian day
/TRACK/	Local COMMON area

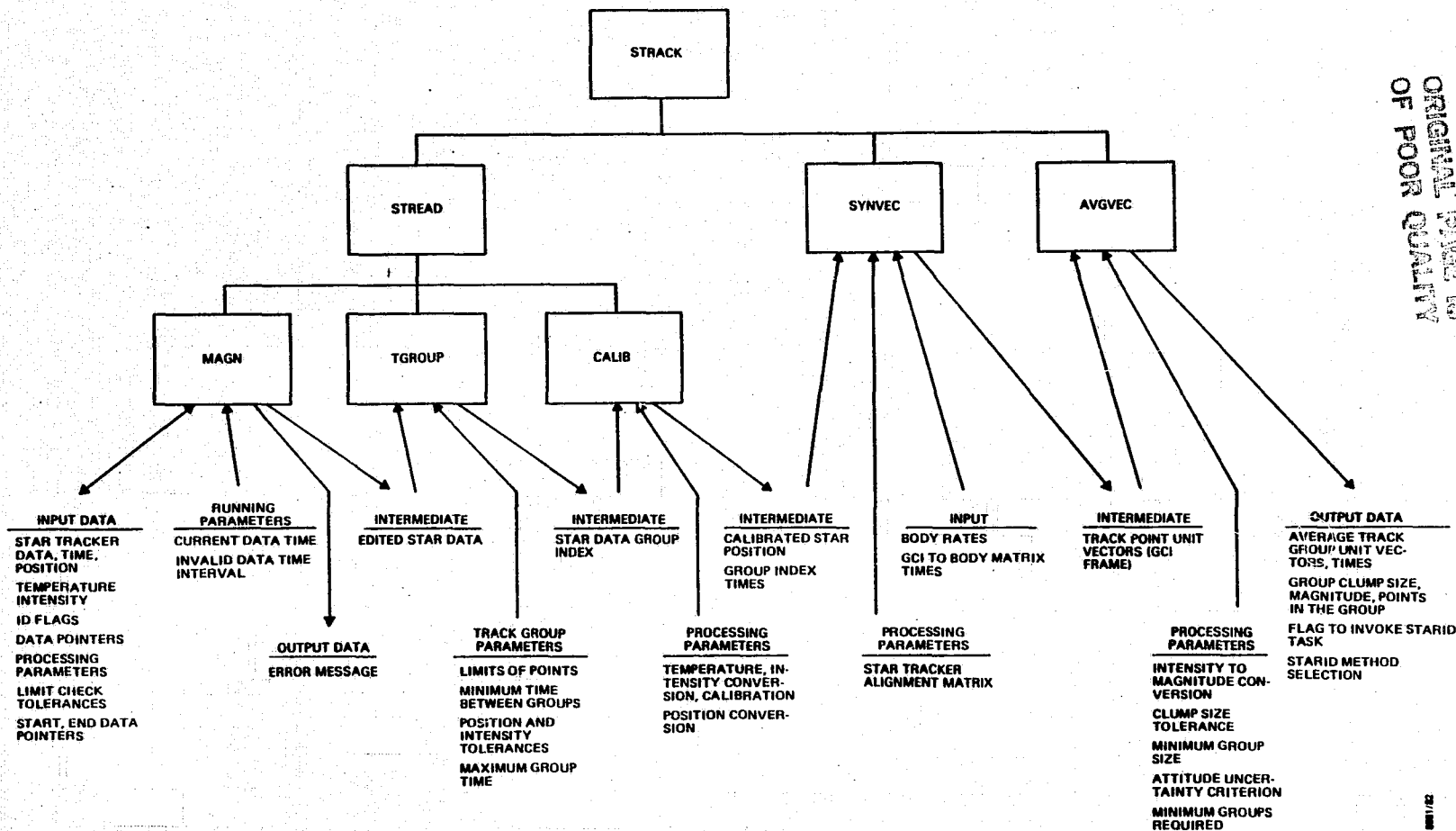


Figure 3-23. STRACK Process Data Flow

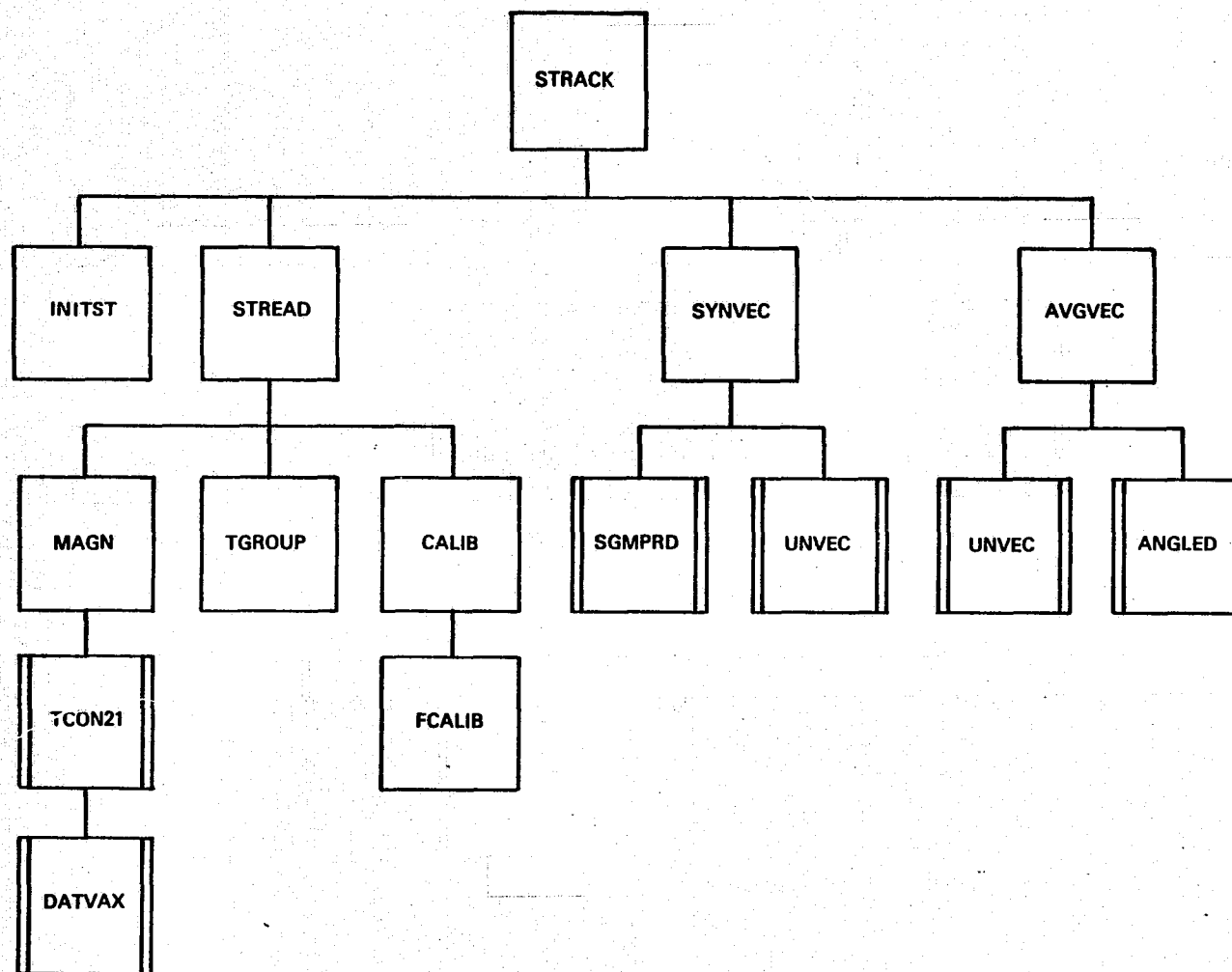


Figure 3-24. STRACK Process Baseline Diagram

6031/22

3.8 STAR IDENTIFICATION PROCESS

This section describes the star identification (STARID) process.

3.8.1 INPUT/OUTPUT

The following subsections describe the input and output parameters for the STARID process. Figure 3-25 illustrates the external interfaces.

3.8.1.1 Input

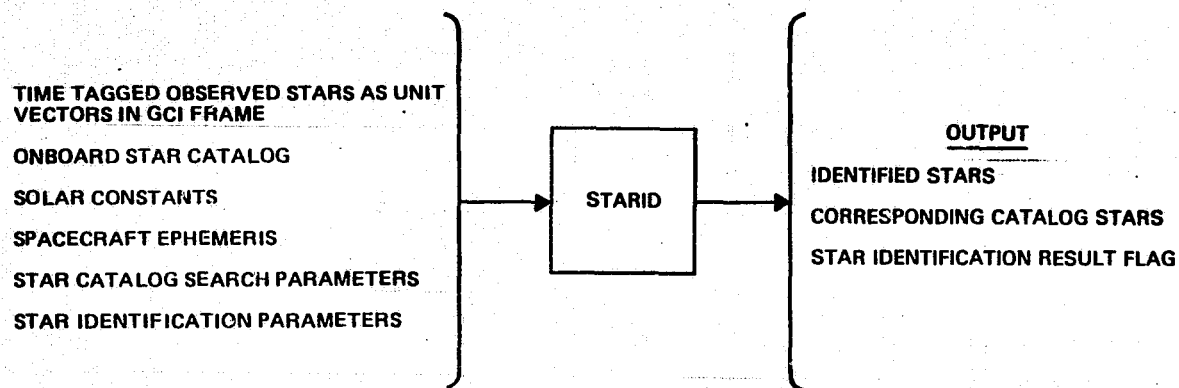
The following items are required as input:

- Track point group average data
 - Times
 - Unit vectors in the GCI coordinate frame
 - Number of track points in each group average
 - Star magnitudes
- Earth ephemeris
- Solar ephemeris computation parameters
- Star catalog and star catalog search parameters
- Star identification parameters
- Flag set by the EXEC process to indicate initialization or a normal processing mode

3.8.1.2 Output

The following items are output from the STARID process:

- Averaged star data corrected for stellar aberration
 - Unit vectors in the GCI coordinate frame
 - Number of track points in each group (average)
 - Star magnitudes



8581/82

Figure 3-25. STARID Process External Interfaces

- Catalog star data (corrected for precession and nutation)
 - Unit vectors in the GCI coordinate frame
 - Star magnitudes
 - Star identification numbers
- Flag to indicate success or failure of star identification process

3.8.2 PROCESSING

The STARID process matches the observed star vectors with star vectors given in a star catalog. The observed star unit vectors are corrected for the stellar aberration effect. The spacecraft velocities with respect to the Sun are calculated for this purpose. The observed stars are temporarily rotated to the epoch time (year 2000) of the AADS star catalog during the search for candidate stars.

The star catalog is searched to find candidate stars that fall within a certain range of the observed star positions and that have comparable magnitudes depending on the tolerances. The observed stars for which candidate stars have not been found are flagged. Those with one or more candidate stars are identified using direct matching or pairwise matching techniques. If star identification is successful, then the resulting candidate stars are rotated to the current time from year 2000 (star catalog epoch time). The observed stars that are successfully identified will be used in the UPDATE process to update the state vector.

Table 3-8 contains the descriptions of STARID process components to be used with Figures 3-26 and 3-27. Figure 3-26 illustrates data flow within the process and Figure 3-27 illustrates the relationship among process components.

Table 3-8. STARID Process Component Description

<u>Component</u>	<u>Description</u>
STARID	Driver for star identification process
INITID	Initializes star identification working parameters
ABER	Corrects star unit vector in inertial coordinate frame for stellar aberration
SMPOS	Computes solar ephemeris
CATLG	Driver to perform star catalog search for candidate stars
PRENUT	Gives precessional and nutational correction for star position
SUBCAT	Gives starting location and number of stars in a star subcatalog
DESTAR	Decodes a star catalog entry to give right ascension and declination angles and star magnitude
ANGLED	Gives angle between two vectors
/IDSTAR/	Local COMMON area
IDENT	Subdriver for star identification
IDENT1	Identifies observed stars using direct matching method
IDENT2	Identifies observed stars using pairwise matching method
PAIR2	Identifies remaining observed stars using a pairwise matching method after a pair of stars has been identified
INTERP	Interpolates Earth ephemeris for the current time
RADECM	Computes right ascension, declination, and magnitude for a given vector
VEC	Computes unit vector from right ascension and declination

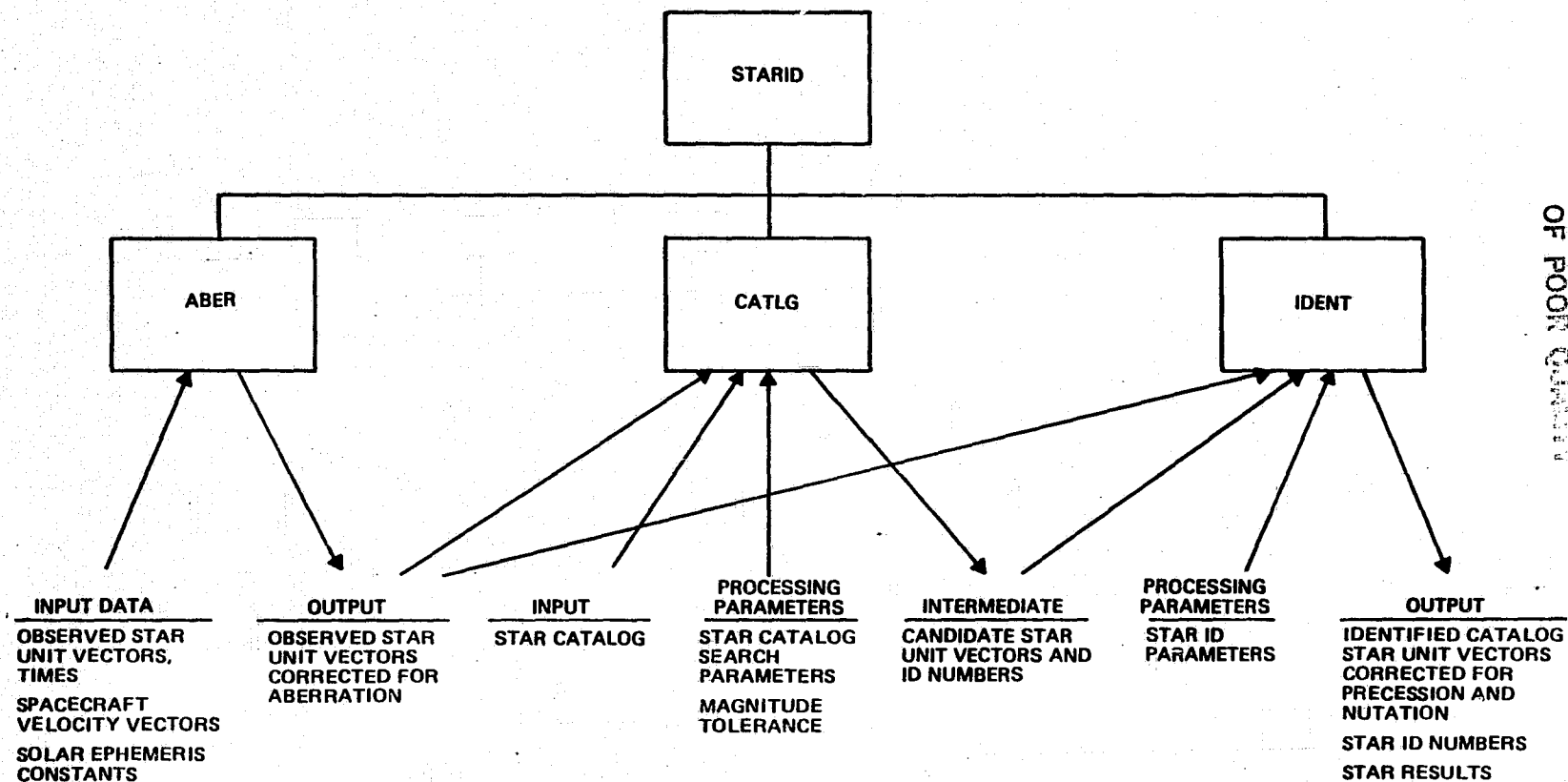


Figure 3-26. STARID Process Data Flow

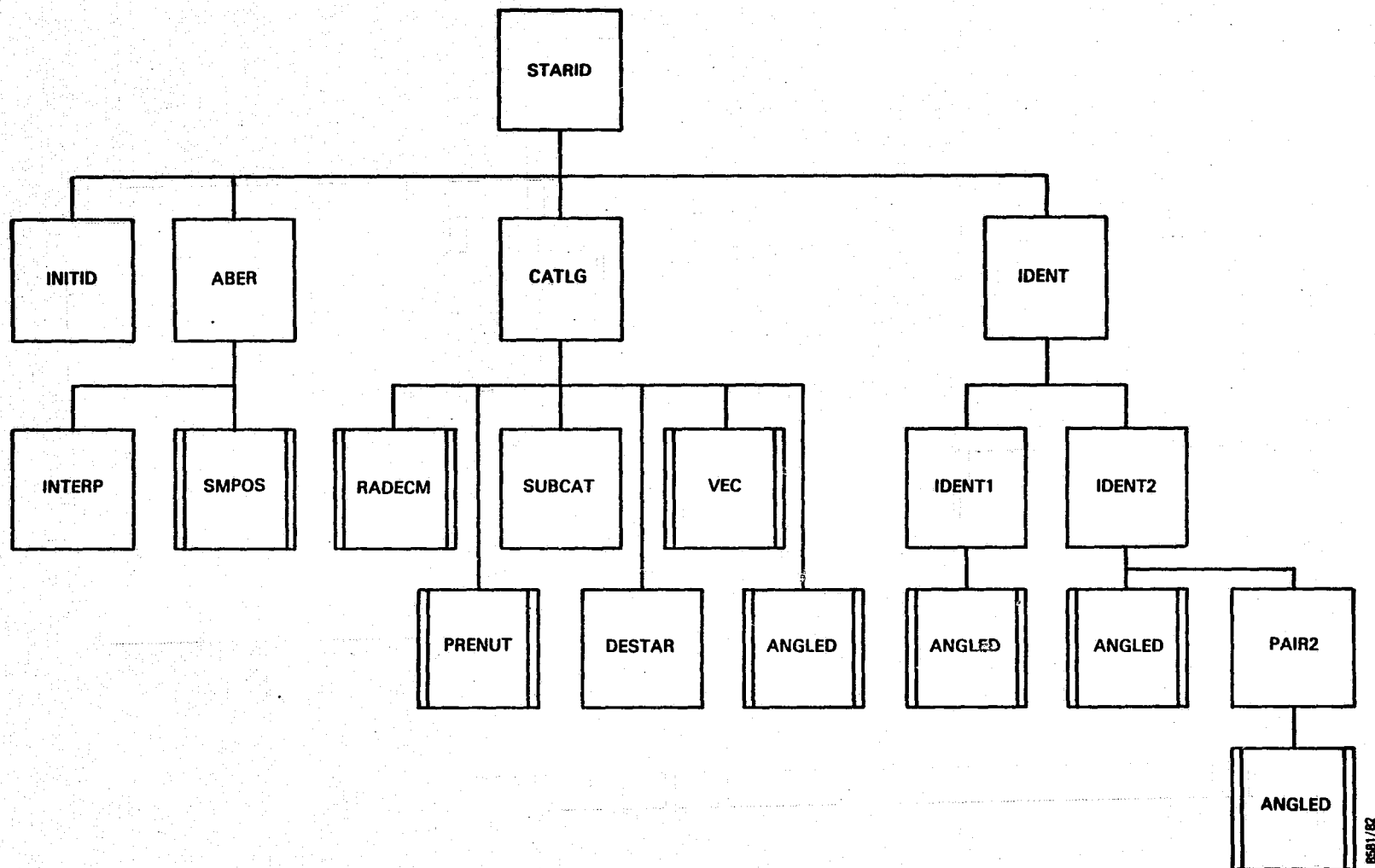


Figure 3-27. STARID Process Baseline ID

3.9 STATE UPDATE PROCESS

This section describes the state update (UPDATE) process.

3.9.1 INPUT/OUTPUT

The following subsections describe the input and output parameters for the UPDATE process. Figure 3-28 illustrates the external interfaces.

3.9.1.1 Input

The following items are required as input:

- Averaged observed star data
 - Unit vectors in the GCI coordinate frame
 - Number of track points in each average
 - Clump size of the group
- Catalog star unit vector in the GCI coordinate frame
- Observation noise
- GCI to body transformation and star tracker alignment matrices
- Current state vector
- Current covariance matrix
- Input parameters set by the EXEC process for initialization on nominal processing

3.9.1.2 Output

The following items are output from the UPDATE process:

- Updated state vector
- Updated state covariance matrix

3.9.2 PROCESSING

The UPDATE process uses a least-squares algorithm to minimize the distances between observed and predicted star positions; in the process, it updates the gyro-propagated state

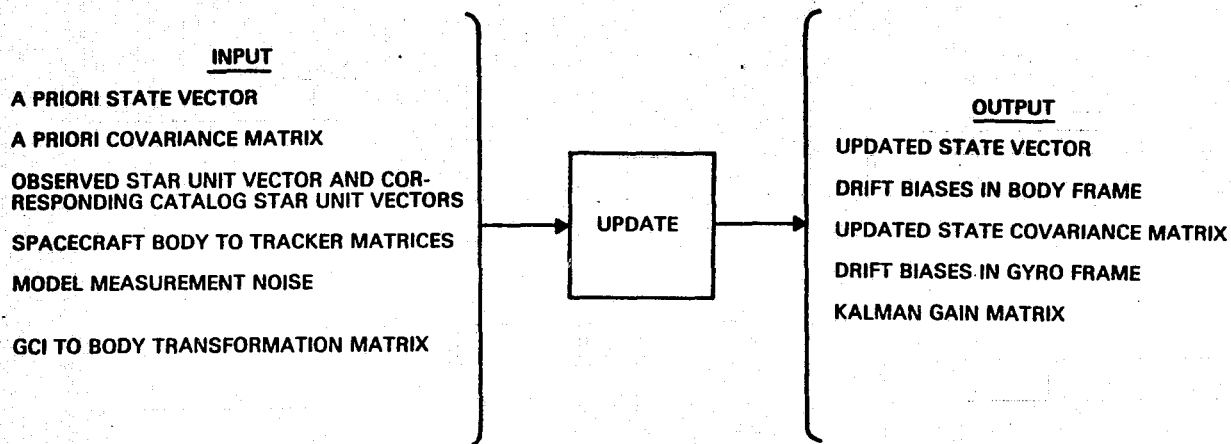


Figure 3-28. UPDATE Process External Interfaces

vector that contains the spacecraft attitude quaternion and gyro drift biases.

The observed star unit vectors in the GCI coordinate frame are rotated into a tracker coordinate frame at the current gyro-propagated attitude. The model observations are computed from unit vectors in the tracker coordinate frame. The observation noise is computed, using the maximum of the model noise and the group clump size. The model predictions are computed using catalog star unit vectors and the current gyro-propagated attitude. Observation residuals and the partials are computed. A least-squares algorithm is used to minimize the differences between observed and predicted values (observation residuals) and to compute corrections to the gyro-propagated state vector and covariance matrix. Finally, the state vector is updated with the quaternion part renormalized.

Table 3-9 contains the descriptions of UPDATE process components to be used with Figure 3-29 and 3-30. Figure 3-29 illustrates data flow within the process and Figure 3-30 illustrates the relationship among process components.

Table 3-9. UPDATE Process Component Descriptions

<u>Component</u>	<u>Description</u>
UPDATE	Driver for state update process
INITUP	Initializes state vector, state covariance matrix, and differential state vector
KALMAN	Driver for Kalman filter algorithm
PARTL	Computes measurement partials
PROBVC	Computes predicted observation vector in body coordinate frame using the star reference vector and a priori attitude
RECUR	Computes gain matrix, correction to state vector, and covariance matrix using a recursive estimation algorithm
SGMPRD	Matrix product routine
/UPDAT/	Local COMMON area

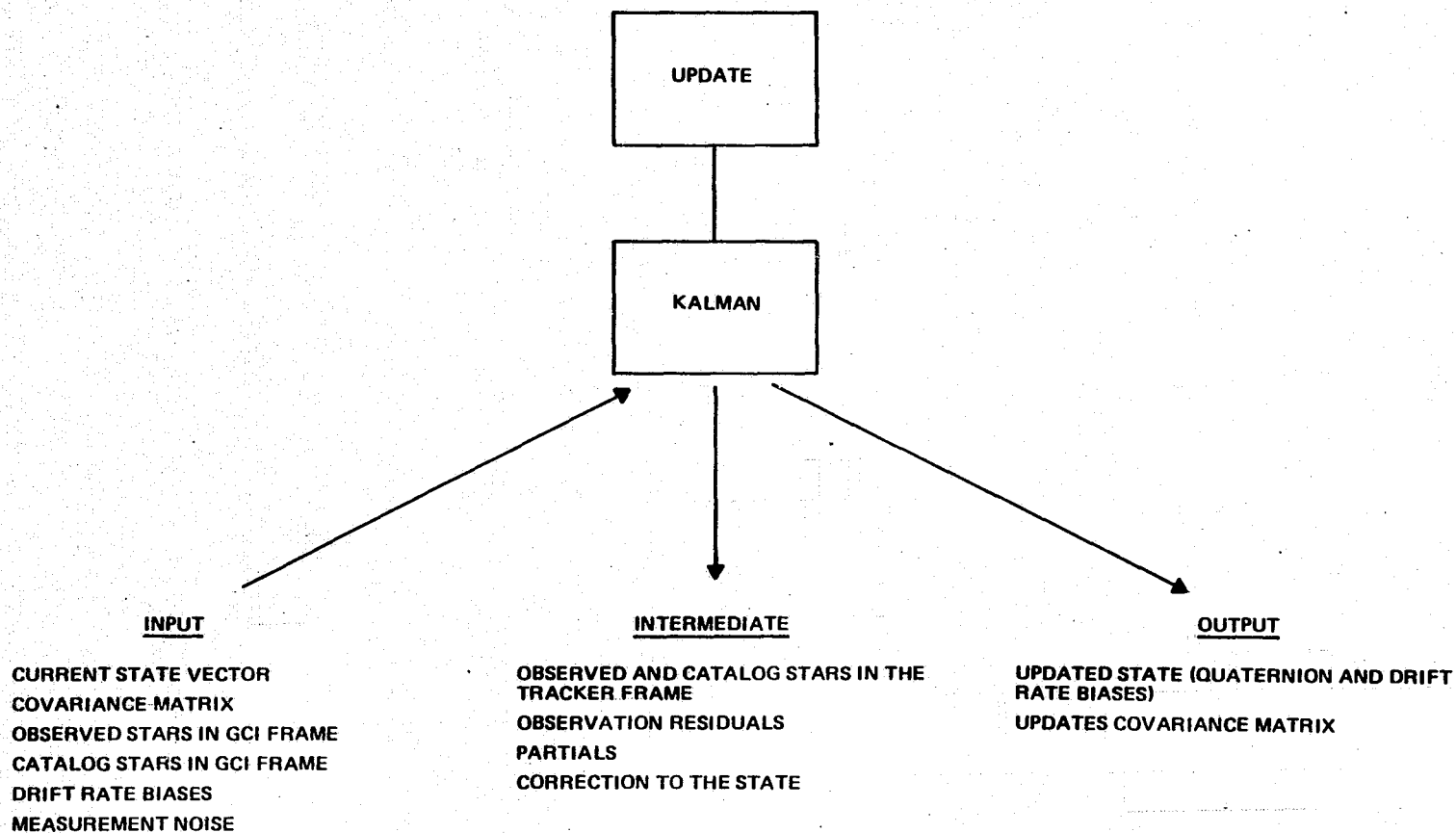


Figure 3-29. UPDATE Process Data Flow

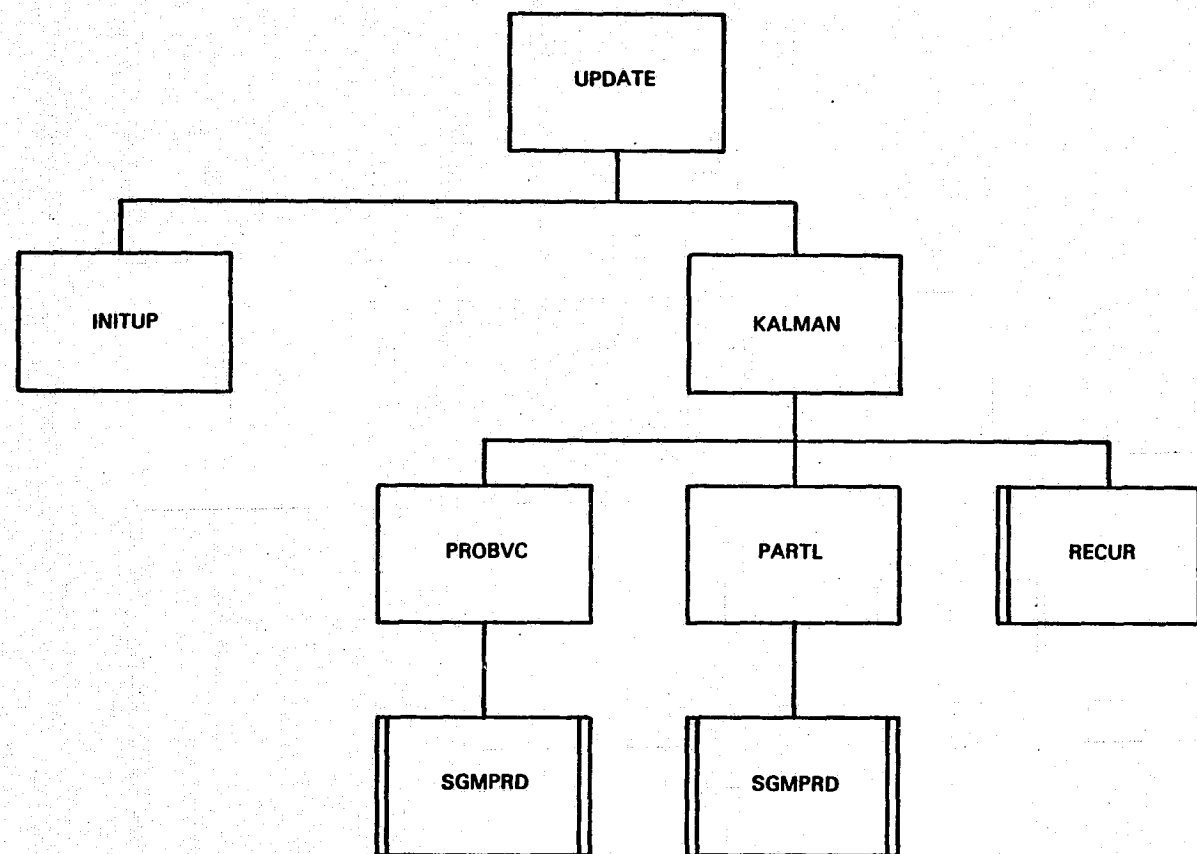


Figure 3-30. UPDATE Process Baseline Diagram

3.10 AADS STAR CATALOG GENERATION UTILITY (CATGEN)

This section describes the generation of the AADS star catalog that is used during star identification. The AADS star catalog is a subset of the main SKYMAP star catalog (Reference 16), and contains stars with visual magnitude in the range 2.0 to 6.5. The star right ascension and declination angles are stored with an accuracy of 0.00005 degree or 0.18 arc-second. The visual magnitudes are stored with an accuracy of 0.05. The AADS star catalog contains approximately 8700 stars and requires approximately 52 kilobytes of memory space. Further information about the structure of the star catalog is available in the file [AADS.STAR]CATADS.FOR and in Section 4.2.2.

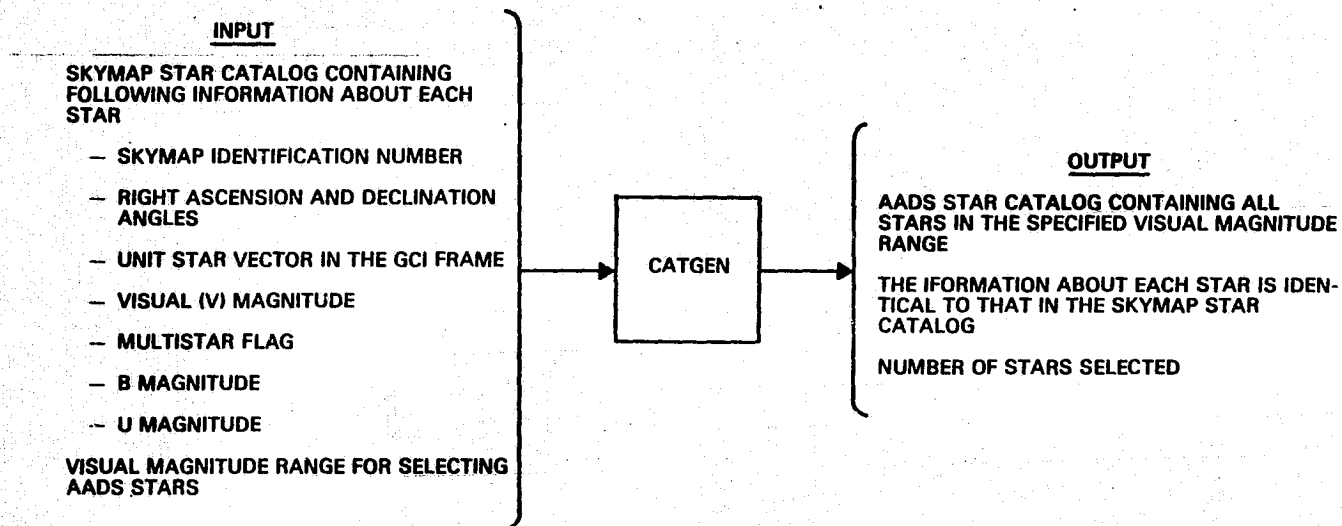
3.10.1 INPUT/OUTPUT

The following subsections describe the input and output parameters for the CATGEN utility. Figure 3-31 illustrates the external interfaces.

3.10.1.1 Input

The following items are required as input:

- SKYMAP star catalog information for each star
 - SKYMAP identification number
 - Right ascension and declination angles and unit star vector in the GCI frame (star position is for epoch 2,000)
 - Visual (V) magnitude
 - Multistar flag
 - B magnitude
 - U magnitude
- Visual magnitude range to select the stars for the AADS star catalog



8581/82

Figure 3-31. CATGEN Utility External Interfaces

3.10.1.2 Output

The following items are required as output:

- All SKYMAP stars that fall in specified visual magnitude range
- SKYMAP star catalog information
 - SKYMAP identification number
 - Right ascension and declination angles and unit star vector in the GCI frame (star position is for epoch 2,000)
 - Visual (V) magnitude
 - Multistar flag
 - B magnitude
 - U magnitude
- Number of stars selected

3.10.2 PROCESSING

The CATGEN utility comprises four separate standalone programs. The SKYMAP star catalog is stored on several tapes accessible from the IBM S/360-95 in Building 3 at Goddard Space Flight Center (GSFC). The SKYCAT program executes on the IBM S/360-95. SKYCAT reads the SKYMAP catalog and selects the stars that fall in the specified visual magnitude range. Pertinent information about each selected star is written to an output tape. This tape is carried to Building 23 at GSFC. The program IBMVAX executes on the VAX-11/780 computer to read this tape, which is in the IBM internal format, and converts it to the VAX internal format. See Guide for Converting Software From System 360 to VAX 11/780 by J. Watson of NASA/GSFC (April 1981) for further information about the internal formats used by the two computers and the conversion procedures. The program IBMVAX

reads each star entry on the input tape, performs the conversion from IBM to the VAX internal format, and writes the output to a data set MATCAT.DAT on the VAX-11/780. Next, program SORT reads all star entries in the data set MATCAT.DAT and sorts all entries in terms of ascending right ascension angles. The sorted output is written to a data set SORTED.SAV.

Finally, the AADS star catalog is created when the STARID process is invoked for the first time during AADS execution. As part of the initialization, STARID calls subroutine CATADS. CATADS reads the star data from data set SORTED.SAV, encodes the right ascension and declination angles and the visual star magnitude information for each star, and stores the information in a COMMON area /CATLOG/. The STARID process calls routines SUBCAT and DESTAR, which decode star information from the star catalog and return the right ascension and declination angles and the visual magnitude.

When AADS is implemented on the target microcomputer, it may be desirable to use an initialized COMMON area /CATLOG/ instead of re-creating the star catalog in /CATLOG/ each time AADS is started. In this case, the routine CATADS will have to be modified to save the AADS star catalog after it is created. The AADS star catalog will be saved as an object module CATLOG\$DATA.OBJ, which will be used during the link step to create the executable image for the STARID process.

Table 3-10 contains the descriptions of CATGEN utility components to be used with Figures 3-32 and 3-33. Figure 3-32 illustrates the data flow within the utility and Figure 3-33 illustrates the relationship among components. The star identification process STARID shown in these figures is not a part of the CATGEN utility. STARID uses the star catalog

Table 3-10. CATGEN Utility Component Descriptions

<u>Component</u>	<u>Description</u>
SKYCAT	Main routine reads the SKYMAP star catalog on IBM S/360-95 and outputs information for selected stars in specified visual magnitude range to a tape
IBMVAX	Main routine converts the tape in the IBM internal format to the VAX-11/780 internal format
FTIO	Input/output routines to mount a tape, read from a tape data set, and so forth
SET_UP	Subroutine sets up output buffer used to store output in VAX internal format
CONVERT	Converts single variables from IBM to VAX internal format
SORT	Main routine reads data set created by IBMVAX and sorts stars in ascending order of right ascension angles
STARID	AADS star identification process; calls routine CATADS to create AADS star catalog during initialization; calls routines SUBCAT and DESTAR to decode star information in star catalog
/CATLOG/	The COMMON area which contains the AADS star catalog
CATADS	Reads direct access, sorted data set created by SORT, encodes star information, and creates AADS star catalog
SUBCAT	Subroutine locates starting address of a given subcatalog and determines number of stars in that subcatalog
DESTAR	Subroutine decodes a single star entry and returns star right ascension and declination angles, and visual magnitude

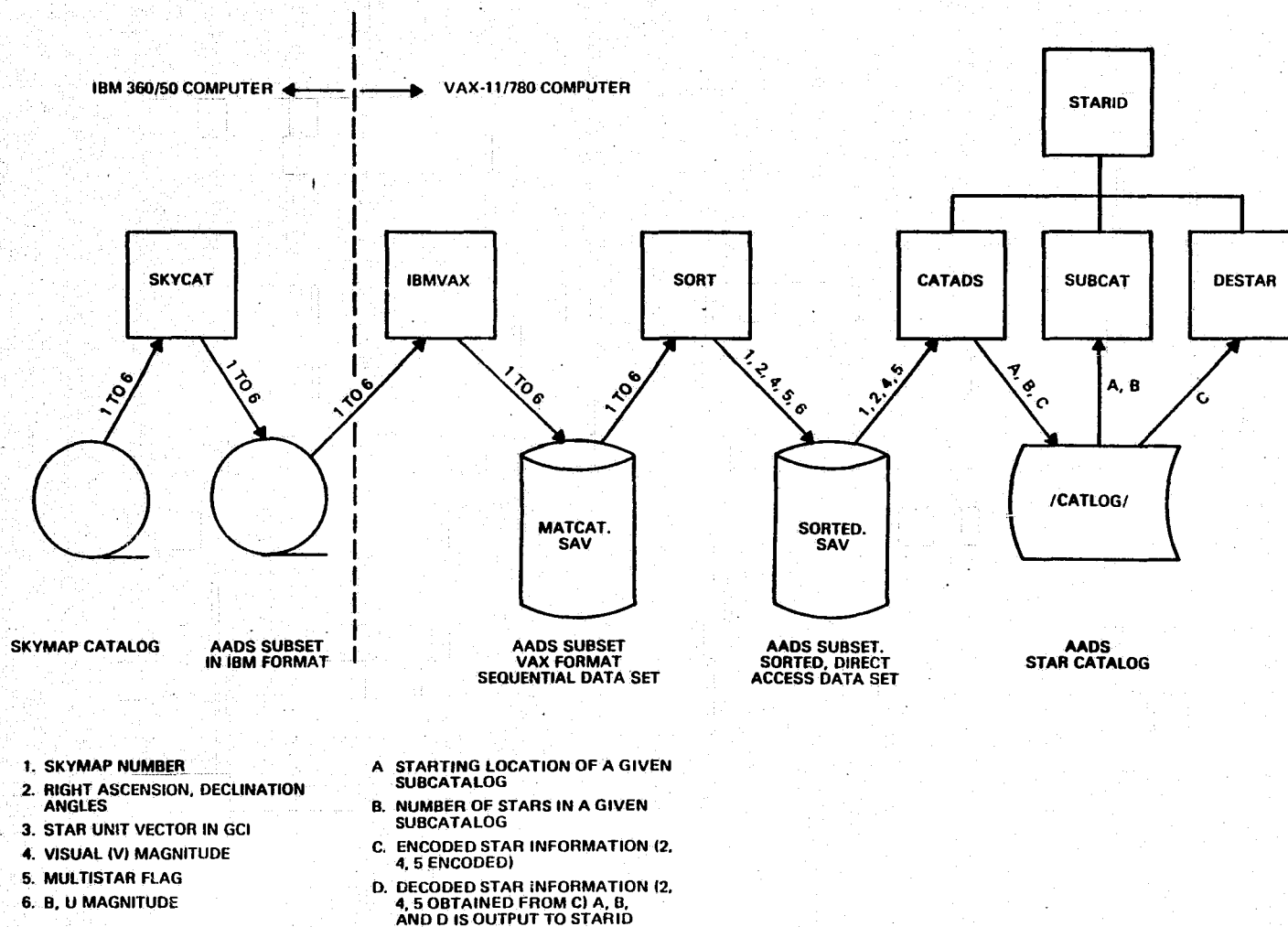


Figure 3-32. CATGEN Utility Data Flow

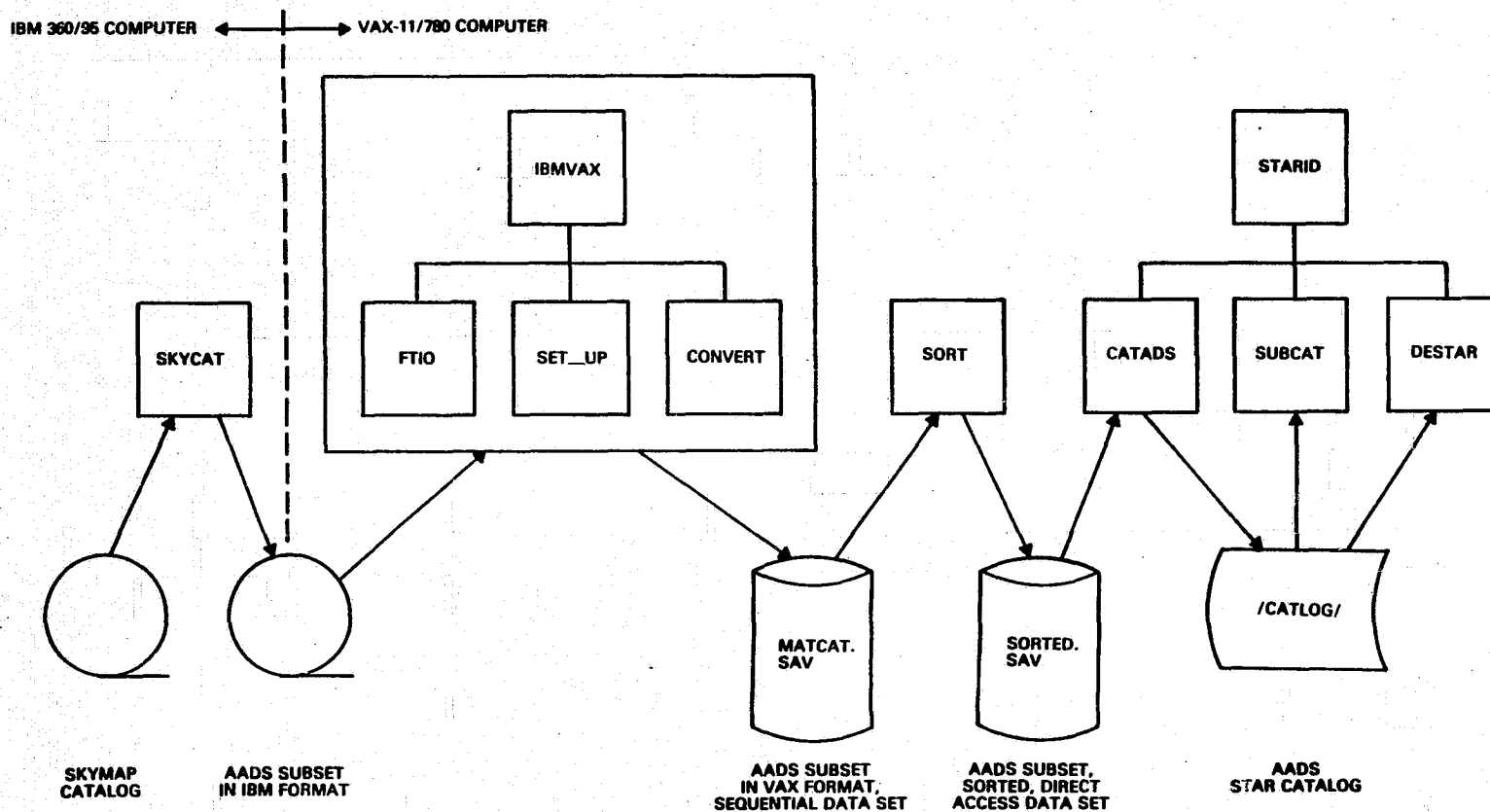


Figure 3-33. CATGEN Utility Baseline Diagram

created by CATGEN utility and is shown here to demonstrate the use of CATADS, DESTAR, and SUBCAT routines during star identification.

SECTION 4 - REQUIREMENTS FOR EXECUTION

4.1 GENERAL INFORMATION

AADS runs on the VAX-11/780 computer or runs in a VAX-Intel 8086 system configuration. In both cases AADS runs in real time and is supported by the AADS simulator residing on the VAX computer.

AADS runs under the VAX/VMS operating system on the VAX computer and under a custom-made operating system (References 3 and 10) on the Intel computer. The operating system provides timing services and interrupts; multiprocessing capability; data management and input/output services; and error recovery at system level. AADS also operates under a system executive (EXEC), which works as a mini-operating system that interfaces with the main operating system. EXEC determines schedules for subprocesses, control of execution sequence, and handling of specific AADS contingencies and error conditions.

The operator interfaces with AADS via the ground support simulator (GSS), which resides on the VAX computer. The system startup procedure includes allocation of resources and assignment of input/output devices. Next, the AADS simulator processes (GSS, SDS, and OSS) are activated and the system executive in AADS is activated, which, in turn, activates other processes in AADS at appropriate times. This complex sequence of events forms a digital command language (DCL) procedure that should be used during the startup to eliminate setup errors.

AADS runs in real time and uses a memory region size of approximately 200 kilobytes when all system capabilities are exercised. AADS needs a cathode ray tube (CRT), disk space and a line printer in the minimum system configuration. The simulator data segments used for AADS testing are normally

stored on disk data sets, but the segments must be stored on tape data sets when the span of data exceeds 4 to 8 hours. In this case, a tape drive is required for AADS testing.

The analyst must be able to initiate system execution (allocation of resources, assignment of devices, and use of DCL procedures), and must be familiar with the commands and control parameters that control AADS execution. These commands and the control parameters are available for operator selection on menu and parameter displays shown by GSS. GSS also receives various reports and messages from AADS and prints or displays these reports for review by the analyst. Besides the analyst, the operator and the maintenance programmer must be familiar with system execution requirements and other computer resources.

Section 4.2 describes computer resources and Section 4.3 describes generation and execution of AADS and other associated systems. Sections 4.4 and 4.5 describe AADS control parameters and error messages, respectively.

4.2 RESOURCES

AADS system generation and execution require resources in the following situations:

- Both AADS and the AADS simulator on the VAX computer
- AADS on an Intel 8086 and the AADS simulator on VAX
- AADS on an Intel microcomputer onboard a spacecraft

During initial development, both AADS and the AADS simulator reside on the VAX computer in the form of executable images. There is no overlay requirement and the VAX/VMS operating system controls the VAX memory space by paging in and out the image segments that are required for execution at a given time. AADS requires approximately 200 kilobytes of memory space. A CRT, a line printer, or a hardcopy terminal is the required resource for a typical test run.

Various reports generated by AADS and any optional diagnostic test output is usually routed to disk data sets for later hardcopy printing and review.

System services required for AADS execution are briefly described here. For additional information, see Reference 15.

VAX/VMS user privileges required for execution of AADS and the AADS simulator follow.

<u>Privilege</u>	<u>Description</u>
GRPNAM	Inserting a name in group logical name table
GROUP	Controlling other processes in same group
ALTPRI	Altering execution priority values for processes
TMPMBX	Creating temporary mailbox
NETMBX	Creating network device

The following is a list of quota values used during AADS system testing.

<u>Quota</u>	<u>Description</u>	<u>Value</u>
PQL\$_ASTLM	AST limit	50
PQL\$_BIOLM	Buffered input/output limit	50
PQL\$_BYTLM	Buffered input/output byte count	128,000
PQL\$_DIOLM	Direct input/output	50
PQL\$_FILLM	Open file	200
PQL\$_PGFLQUOTA	Paging file	20,000
PQL\$_PRCLM	Subprocess	80
PQL\$_TQELM	Timer queue entry	80
PQL\$_WSDEFAULT	Default working set size	300
PQL\$_WSQUOTA	Working set size	300

The following VAX/VMS system services are used by the AADS.

<u>Name</u>	<u>Description</u>
SYS\$_ASSIGN	Assigning input/output channel
SYS\$_CLREF	Clearing event flag
SYS\$_CREPRC	Creating process (start process execution)
SYS\$_GETTIM	Getting time in 64-bit format
SYS\$_HIBER	Hibernating
SYS\$_NUMTIM	Converting binary time to numeric time
SYS\$_QIO	Queuing input/output request
SYS\$_READEF	Reading event flags
SYS\$_SETIMR	Setting timer
SYS\$_CANTIM	Canceling timer
SYS\$_SETPRI	Setting priority
SYS\$_WAITFR	Waiting for single event flag
SYS\$_WAKE	Waking up hibernating process
SYS\$_WFLOR	Waiting for logical OR of event flags
SYS\$_ASCEFC	Associating common event flag cluster
SYS\$_CRMPSC	Creating and mapping section
SYS\$_MGBLSC	Mapping global section
SYS\$_TRNLOG	Translating logical name

NOTE: The last four system services are not used when AADS executes on the Intel computer.

In addition, the VAX FORTRAN library routines SECNDS and IDATE are used. AADS uses the event flag cluster that contains event flags event flag numbers 64 through 95, with the name 'AADS'. The following is a list of the event flags and their purposes.

<u>Event Flag Number</u>	<u>Referencing Processes</u>	<u>Purpose</u>
64	DCAPIN, EXEC	Notification of command received
65	DCAPIN, EXEC	Notification of DCAPIN error condition
66	DCAPIN, EXEC	Resumption of uplink data or ephemeris data validation

<u>Event Flag Number</u>	<u>Referencing Processes</u>	<u>Purpose</u>
67	DCAPIN, EXEC	Notification of completion of uplink data or ephemeris data validation
68	SENSIN, EXEC	Notification of SENSIN process completion
70	GYPROC, EXEC	Notification of GYPROC process completion
72	PROPAG, EXEC	Notification of PROPAG process completion
74	STRACK, EXEC	Notification of STRACK process completion
76	STARID, EXEC	Notification of STARID process completion
78	UPDATE, EXEC	Notification of UPDATE process completion
80	OUTPUT, EXEC	Notification of OUTPUT process completion
82	EXEC	Timer event flag
84	DCAPIN	QIO read event flag
86	SENSIN	/RIU/ access synchronization

AADS uses several COMMON areas during execution. COMMON areas STAT1, STAT2, STAT3, and STAT4 contain AADS control parameters. COMMON areas FLIGHT, GLOBAL, SENSOR, and STATE are used by AADS processes for interprocess communication. These COMMON areas are contained in a global section named 'AADS-COMMON'.

The COMMON area /ADSGBL/, from which the simulation time offset is obtained, is contained in the global section named 'ADSGBL'. EXEC is the only process of AADS to reference this global section. The COMMON area /RIU/, from which the raw sensor data are obtained, is contained in the global section name 'RIU'. SENSIN is the only process of AADS to reference this global section.

The following is a list of the process priorities used during AADS system testing.

<u>Process</u>	<u>Priority</u>
DCAPIN	15
SENSIN	14
GYPROC	10
PROPAG	10
STRACK	6
STARID	6
UPDATE	6
OUTPUT	8
EXEC	12

Mailboxes are used to accomplish the input/output between AADS and the AADS simulator. The logical names of these mailboxes are as follows.

<u>Mailbox Logical Name</u>	<u>Function</u>
UPLINK	Uplink data from OSS to AADS
DNLINK	Downlink data from AADS to OSS

AADS will be transferred to the Intel 8086 computer during the final stage of AADS prototype development. The AADS simulator will still reside on the VAX computer. In addition, the Intel executable version of AADS source will be created on an Intel development system (Reference 18) and will be stored in a disk data set on the VAX computer. The executable images will be transferred into the Intel memory before execution because the Intel computer (the final target computer) will not have any peripheral devices for storage. All AADS process images, the AADS star catalog, the new Intel operating system (Reference 10), and the mathematical library will reside in the Intel memory. The current estimates indicate that AADS will require approximately

315 kilobytes. These are conservative estimates and assume an expansion factor of 2.0 for the object code between the FORTRAN compilers for the VAX and the Intel computers.

AADS execution in the VAX-Intel configuration requires the resources mentioned for the VAX computer, except the SYS\$_ASCEFC, SYS\$_CRMPSC, SYS\$_MGBLSC, and SYS\$_TRNLOG system services. A thorough description of requirements and system services can be found in References 10 and 17. In addition, the following resources are required:

- Intel development system with edit, compile, and link facilities; communication lines and supporting software for transmitting data between VAX and Intel (development) computers
- Intel (target) computer with 8087 NDP chip; custom-made operating system and other required system software; communication lines and supporting hardware/software for transmitting data and timing information between VAX and Intel (target) computers

It should be noted that AADS is initially developed on the VAX computer. During the second stage, AADS is implemented on an Intel computer that is called the target computer. However, any modifications to the AADS code required for the implementation on the target computer, are made on the Intel microprocessor development system (MDS). The Intel MDS is called the development computer and has facilities for editing, compiling, and linking Intel executable images. The target computer does not have any peripheral devices (no disk storage); therefore, the Intel executable images are stored on the VAX, and downloaded to the target computer when required.

In its final operational phase, AADS will reside in an executable image form on an Intel microcomputer onboard a spacecraft. AADS will have access to the spacecraft clock

for timing information, will access data from the onboard sensors with the remote interface unit (RIU), and will provide output via the onboard computer (OBC) in the spacecraft telemetry. GSS will still interface with AADS by providing control information to AADS, and by displaying AADS output. Computer resources should be similar to those required for the VAX-Intel configuration. The resources will depend on the specific mission, and therefore, are not described here.

4.2.1 FILE STRUCTURE ON THE VAX

AADS source code, executable images, and other support data sets are stored on disk as VAX files. A VAX file is specified as DEVICE:[DIRECTORY]filename.type;version. At the present time, the disk device DBA0 and the default directory [AADS] are used for AADS files. The file type describes the type of data in the file. The following file types are used:

<u>File Type</u>	<u>Description</u>
FOR	FORTTRAN source code
MAR	Assembly language source code
INC	INCLUDE files
OBJ	Object module output from a compiler or an assembler
EXE	Executable program image
DIR	Subdirectory
COM	DCL procedure
LD	Files for simulator data
TXT	Text files (for example, AADS simulation case descriptions)
SAV	AADS simulation cases and other permanent data sets
INP	Input to AADS or AADS simulator

AADS was developed in four builds. The executive, input/output functions, and interfaces with the AADS simulator were tested during builds 1 and 2. The mathematical subsystems for gyro data processing and star data processing were

implemented in builds 3 and 4, respectively. The files for these builds are stored under separate subdirectories as follows:

<u>Subdirectory</u>	<u>Description</u>
EX2	Build 2 EXEC source, object modules, and compile and link DCL procedures
IO2	Build 2 SENSIN, DCAPIN, and OUTPUT sources
GY3	Build 3 GYPROC and PROPAG sources
ST4	Build 4 STRACK, STARID, and UPDATE sources
STAR	Star catalog utility source
UT	Utility routines and object modules
BTn	Executable images, execution DCL procedures, simulation cases, and test data segments used for the nth build (n = 2, 3, or 4)
CMn	INCLUDE files and BLOCK DATA modules

Build 4 represents the final AADS version on the VAX computer. Therefore, the subdirectory [AADS.BT4] contains the executable images for the completed AADS.

Some source changes are required for the VAX-Intel version of AADS due to the differences between the FORTRAN compilers and the operating systems used for the two computers. For example, the FORTRAN source, object code, and executable images for the VAX-Intel configuration will be stored under separate subdirectories. These subdirectories have not yet been defined.

The file structure definition for AADS data sets is given above. See Reference 15 for more details on the file structure.

4.2.2 DATA SET DEFINITIONS

AADS creates and uses a star catalog during execution. The AADS star catalog is a subset of the SKYMAP star catalog (Reference 16). Utility programs described in Sections 3.10 and 4.3.4 read the SKYMAP catalog on the IBM S/360-95 computer and create the AADS star catalog data set used by AADS. Table 4-1 shows the format of an intermediate data set (MATCAT.DAT) that contains the SKYMAP stars in the visual magnitude range 2.0 to 6.5. Table 4-2 shows the format of the data set (SORTED.SAV) that contains star entries in terms of ascending star right ascension angle. This is a direct access data set with a 28-byte logical record size, and has 8,666 records.

SORTED.SAV is accessed by GSS to extract the SKYMAP identification number of the stars identified by AADS. SORTED.SAV is also used by AADS to create the AADS star catalog. The format of the AADS star catalog is described in the file DBA0:[AADS.STAR]CATADS.FOR. The AADS star catalog is not a data set; it is a data structure internal to AADS, and is stored in the global COMMON area /CATLOG/.

4.2.3 MEMORY REQUIREMENTS

Memory requirements can be broken down into global COMMON areas, library routines, and code and data.

On the VAX 11/780 computer, the AADS global COMMON areas require 67.4 kilobytes, including the 55.9 kilobytes for the star catalog. VAX/VMS requires that each global COMMON area be page aligned (256-byte boundary). On the Intel 8086/8087 microprocessor, each global COMMON area must be paragraph aligned (16-byte boundary), saving a small amount of memory on the Intel.

Table 4-1. Format of the Data Set MATCAT.DAT

<u>Variable¹</u>	<u>Byte Number</u>	<u>Type</u>	<u>Description</u>
BUFVAX(1)	1-4	I*4	SKYMAP number
RLARR(2)	5-8	R*4	Right ascension
RLARR(3)	9-12	R*4	Declination
RLARR(4)	13-16	R*4	X coordinate (GCI frame)
RLARR(5)	17-20	R*4	Y coordinate (GCI frame)
RLARR(6)	21-24	R*4	Z coordinate (GCI frame)
RLARR(7)	25-28	R*4	B magnitude
RLAAR(8)	29-32	R*4	U magnitude
RLARR(9)	33-36	R*4	V magnitude
BUFVAX(1)	37-40	I*4	Multistar flag

¹The arrays BUFVAX and RLARR occupy the same location in memory (they are equivalent).

Table 4-2. Format of the Sorted Data Set SORTED.SAV

<u>Variable¹</u>	<u>Byte Number</u>	<u>Type</u>	<u>Description</u>
IOUTBF (1)	1-4	I*4	SKYMAP number
ROUTBF (2)	5-8	R*4	Right ascension
ROUTBF (3)	9-12	R*4	Declination
ROUTBF (4)	13-16	R*4	B magnitude
ROUTBF (5)	17-20	R*4	U magnitude
ROUTBF (6)	21-24	R*4	V magnitude
IOUTBF (7)	25-28	I*4	Multistar flag

¹The two arrays IOUTBF AND ROUTBF describe the same memory locations (they are equivalent).

On the VAX, library routines include FORTRAN library, record management service (RMS) routines, and system vectors, taking an average 110 kilobytes per process. On the Intel, a stack takes one-to-two kilobytes (with the system vectors adding slightly more). A two-to-three kilobyte math library will be shared between all processes.

The memory used by each process for code and data is summarized in Table 4-3. The figures for the VAX routines include code used to generate diagnostic test output. The figures for the Intel routines were determined by multiplying the figures for the VAX routines by a conservative expansion factor of 2.0 and adding two kilobytes for the stack.

The current estimates for the Intel operating system, math library, and the AADS COMMON areas are included to help give an overall picture of the total memory requirements. The fourth column in Table 4-3 gives the original memory requirements estimated during the preliminary analysis for AADS. The figure for the executive process was obtained from Reference 5. All other figures were obtained from Reference 11. These figures are included for comparison purposes. The fifth column gives the memory space requirements, for example, format statements and literals. Thus, approximately seven kilobytes of total memory space are required for diagnostic test output in the VAX version. This translates to approximately 15 kilobytes for the Intel version. The final operational version of AADS will not contain the diagnostic test code, and will save the 15 kilobytes of memory space.

4.2.4 TIMING STUDY

AADS timing study has not yet been conducted. This information should be added to the document when the timing study has been completed.

Table 4-3. Memory Requirement Summary

TASK	CURRENT VAX MEMORY (KILOBYTES)	PREDICTED INTEL MEMORY (KILOBYTES)	ORIGINAL MEMORY ESTIMATES (KILOBYTES)	\$PDATA SEC- TIONS FOR VAX (KILOBYTES)
EXEC LESS OCCULTATION PREDICTION	10.6	23.2	22.0	0.2
OCCULTATION PREDICTION	4.1	8.2		
SENSIN	3.2	8.4	2.0	0.0
DCAPIN	8.9	19.8		0.8
OUTPUT	9.5	21.0		0.1
GYPROC	6.2	14.4	4.0	0.7
PROPAG	11.6	25.2	4.0	1.1
STRACK	16.5	35.0	8.0	1.8
STARID	29.8	59.6		2.0
UPDATE	6.9	15.8	4.0	0.7
OPERATING SYSTEM		12	16.0	
MATH LIBRARY		3		
AADS GLOBAL COMMON AREAS INCLUDING STAR CATALOG	67.4	67.4	STAR CATALOG COMMON AREA ONLY 142K	
TOTALS	174.7	313.0	213.0	7.4

8581/82

4.3 EXECUTION AND SYSTEM GENERATION

The following sections describe in detail the DCL procedures used in system generation and execution. See Reference 15 for additional details.

4.3.1 COMMAND FILES

Command files exist for AADS generation and execution. These files contain DCL statements to compile the source, link the object modules and run the AADS system or utility programs. Command files reduce the amount of typing for system generation and execution, and eliminate setup errors. There are nine AADS processes under four subdirectories, each directory containing a compile and a link command file. In addition, there is a run command file for AADS and the AADS simulators. The utility program TMPCOMCRE has a similar command file. A HELP data set is available for most of the command files. The following sections will discuss in detail the use of these command files for system generation and execution.

4.3.2 SYSTEM EXECUTION

Both AADS and the AADS simulator reside on the VAX computer during initial testing (configuration 1), because better software development facilities are available on the VAX computer. AADS is eventually implemented on the Intel 8086 computer and communicates with the AADS simulator using special communication lines between the VAX and the Intel computers. The VAX-Intel configuration is referred to as configuration 2.

To simulate the communication between AADS and the AADS simulator for configuration 1, mailboxes are used (uplink and downlink messages). The AADS simulator transfers the raw sensor data to AADS through a mailbox. A separate process stores the data in a global COMMON area, which is accessed

by AADS on a regular basis depending on the sensor data collection schedule. The mailbox and the associated process simulate the direct memory access (DMA) method that will be used for configuration 2 to store sensor data for AADS. The following sections describe these two cases in more detail.

4.3.2.1 VAX System Configuration

Figure 4-1 shows the command file used for a simulation run of AADS and the AADS simulator during build 2 testing. The command files used for build 3 and build 4 are quite similar to the build 2 command file and the differences, if any, will be transparent to the user. The final version of the command procedure should be available in the VAX file DAB0:[AADS.BT4]BLD4.COM. The command procedure prompts the user for the location of test data, start and end times for test data, and the wall clock time at which the simulation should start. The AADS and AADS simulator processes are then initiated and the control is transferred to GSS. GSS puts up several menu and parameter displays that are used to specify AADS execution during the current simulation run. Finally, GSS stops the simulation run on request and allows selective printing of the AADS output reports and any optional diagnostic test output.

4.3.2.2 VAX-Intel Configuration

The executable form of AADS software is downloaded to the Intel 8086 microprocessor from disk files on the VAX-11/780 computer. Creation of this executable form is described in Section 4.3.3. More information on the downloading procedure can be found in Section 1.3.1 of Reference 10. Two data communication channels are used between the VAX-11/780 computer and the Intel 8086 microprocessor. The first is used for communication between AADS and the OSS. The second is used by the SDS to send simulated sensor data, which the Intel operating system receives and stores in a global

ORIGINAL PAGE IS
OF POOR QUALITY

```

100  $ !
200  $ ! THIS IS THE COMMAND FILE FOR AADS/AADSIM BUILD 2 EXECUTION.
300  $ ! TO INVOKE, TYPE: @BLD2.
400  $ !
500  $ SET DEF LAADS,BT2J
600  $ SET ON
700  $ SET CONTROL_Y
800  $ WRITE SYSSOUTPUT "AADS/AADSIM SIMULATION STARTING - BUILD 2"
900  $                                     ! ENABLE ERROR TRANSFERS AND CONTROL_Y
1000 $                                     ! INTERRUPTS
1100 $ !
1200 $ INQUIRE ALDEV ENTER OUTPUT DEVICE FOR ACTIVITY LOG
1300 $ !
1400 $                                     ! ASK USER TO ENTER DEVICE NAME
1500 $ !
1600 $ IF ALDEV .EQS. "" THEN ALDEV := "DBAU"
1700 $ !
1800 $                                     ! IF NULL STRING ENTERED THEN SET TO
1900 $                                     ! DEFAULT
2000 $ !
2100 $ INLEN = 'F$LENGTH(ALDEV)           ! FIND LENGTH OF INPUT STRING
2200 $ UNDER = 1+'F$LOCATE("_",ALDEV)     ! LOOK FOR UNDERSCORE
2300 $ IF UNDER .EQ. INLEN+1 THEN UNDER = 0
2400 $ !
2500 $                                     ! IF NONE FOUND SET UNDER TO 0
2600 $ !
2700 $ COLON = 'F$LOCATE(":", ALDEV) ! LOOK FOR COLON
2800 $ DEVLEN = COLON - UNDER           ! NUMBER OF CHARACTERS AFTER _ AND
2900 $                                     ! BEFORE !
3000 $ !
3100 $ DEVNAM := 'F$EXTRACT(UNDER,DEVLEN,ALDEV)
3200 $ !
3300 $                                     ! EXTRACT PROPER DEVICE NAME
3400 $ !
3500 $ IF DEVNAM .EQS. "DBAU" .OR. DEVNAM .EQS. "DBB1" THEN GOTO SKIP
3600 $ !
3700 $                                     ! DO NOT ALLOCATE IF DISK IS SPECIFIED
3800 $ !
3900 $ ALLOCATE 'ALDEV'                   ! ALLOCATE DEVICE
4000 $ ASSIGN 'ALDEV' ACTLOG
4100 $ SET NOON
4200 $ SKIP:
4300 $ SKIP:
4400 $ !
4500 $ ! SENSOR DATA SUBSYSTEM (SDS)-----
4600 $ !
4700 $ WRITE SYSSOUTPUT "SENSOR DATA SUBSYSTEM STARTING"
4800 $ SET NOON
4900 $ STIME := 'F$TIME()
5000 $ BLANK := 'F$LOCATE(" ",STIME)
5100 $ STIME[BLANK,1] :=:
5200 $ ON CONTROL_Y THEN GOTO RESTART
5300 $ RUN MB2GS/PROC=MB2GS/PRIORITY=12
5400 $ RESTART:
5500 $ INQUIRE HDEFNM "ENTER HEADER FILE NAME STRING"
5600 $ INQUIRE FNM "ENTER DATA FILE NAME STRING"
5700 $ INQUIRE REFFILE "ENTER REPORT FILE NAME STRING (IF KNOWN)"
5800 $ IF REFFILE.EQS."" THEN GOTO NORANGE
5900 $ @LOCFILE 'REFFILE' 'FNM'
6000 $ NORANGE:
6100 $ OPEN/WRITE SDS_USERIN SDSIN.DAT

```

Figure 4-1. DCL Procedure for AADS Simulation Run (1 of 3)

ORIGINAL PAGE IS
OF POOR QUALITY.

```

0200 $ WRITE SDS_USERID " ",PWW," "
0300 $ CLOSE SDS_USERID
0400 $ INQUIRE STDATA "ENTER START TIME OF DATA (SIM:YYMMDD,HHMMSSMMM)"
0500 $ INQUIRE ETDATA "ENTER END TIME OF DATA (SIM:YYMMDD,HHMMSSMMM)"
0600 $ WRITE SYS$OUTPUT "CURRENT WALL TIME IS:"
0700 $ SHOW TIME
0800 $ INQUIRE STWALL "ENTER START TIME OF DATA (WALL:YYMMDD,HHMMSSMMM)"
0900 $ OPEN/WRITE AADSTIMES AADSTIMES.DAT
1000 $ WRITE AADSTIMES STDATA
1100 $ WRITE AADSTIMES ETDATA
1200 $ WRITE AADSTIMES STWALL
1300 $ CLOSE AADSTIMES
1400 $ ASSIGN /GROUP AADSTIMES.DAT FUR032
1500 $ ASSIGN /GROUP 'HURFNM' HDRFILE
1600 $ ASSIGN /GROUP SDSREPORT.DAT SDSREPORT
1700 $ ASSIGN /GROUP SDSREPORT.DAT FUR037
1800 $ ON CONTROL_Y THEN GOTO QUIT
1900 $ RUN SDSYS/PROCESS_NAME=SDSYS/INPUT=SDSIN.DAT/OUTPUT=SDSOUT.DAT
0000 $ WAIT 00:00:15
0100 $ !
0200 $ ! ONBOARD SUPPORT SUBSYSTEM (OSS)-----
0300 $ !
0400 $ WRITE SYS$OUTPUT "ONBOARD SUPPORT SUBSYSTEM STARTING"
0500 $ RUN OSSIM/PROCESS_NAME=OSSIM/PRIORITY=15/OUTPUT=OSS
0600 $ !
0700 $ ! AUTONOMOUS ATTITUDE DETERMINATION SYSTEM (AADS)-----
0800 $ !
0900 $ WRITE SYS$OUTPUT "AADS STARTING"
1000 $ RUN EXEC.EXE/2/PROCESS_NAME=AADS/PRIORITY=15/OUTPUT='ALDEV'
1100 $ !
1200 $ ! GROUND SUPPORT SUBSYSTEM (GSS)-----
1300 $ !
1400 $ WRITE SYS$OUTPUT "GROUND SUPPORT SUBSYSTEM STARTING"
1500 $ WRITE SYS$OUTPUT "THIS SUBSYSTEM IS RUN INTERACTIVELY FROM THIS TERMINAL."
1600 $ WRITE SYS$OUTPUT "CURRENT WALL TIME IS:"
1700 $ SHOW TIME
1800 $ WRITE SYS$OUTPUT "START TIME OF SIM. IS:"
1900 $ WRITE SYS$OUTPUT STWALL
10000 $ ASSIGN TI SYSSINPUT
10100 $ SHOW TRANS SYSSINPUT
10200 $ WAIT 00:00:15
10300 $ RUN GSS
10400 $ !
10500 $ ! TERMINATE ALL PROCESSES -----
10600 $ !
10700 $ QUIT:
10800 $ ON CONTROL_Y THEN GOTO FIN
10900 $ STOP GYROSIM
11000 $ STOP STARSIM
11100 $ STOP MBZGS
11200 $ STOP SDSYS
11300 $ STOP OSSIM
11400 $ STOP AADS
11500 $ STOP GSOPER
11600 $ STOP GSSIM
11700 $ AGAIN: WAIT 00:00:15
11800 $ SHOW SYSTEM
11900 $ INQUIRE CLEARED ALL AADS PROCESSES OUT OF SYSTEM? (Y/N)
12000 $ IF .NOT. CLEARED THEN GOTO AGAIN
12100 $ DEALLOCATE 'ALDEV'
12200 $ INQUIRE WANT OUTPUT TO THE PRINTER? (Y/N)

```

Figure 4-1. DCL Procedure for AADS Simulation Run (2 of 3)

ORIGINAL PAGE IS
OF POOR QUALITY

```

12300 $ IF .NOT. WANT THEN GOTO FIN
12400 $ !
12500 $ ! PRINT OUTPUT -----
12600 $ !
12700 $ OUTPUT:
12800 $ DIR /SINCE='TIME'/SIZE
12900 $ WRITE SYSSOUTPUT "IF EXCESSIVE OUTPUT, GO TO 2ND TERMINAL AND RENAME
13000 $ WRITE SYSSOUTPUT "SELECTED DAT FILES TO ANOTHER TYPE."
13100 $ INQUIRE OK IS AMOUNT OF OUTPUT OK? (Y/N)
13200 $ IF .NOT. OK THEN GOTO OUTPUT
13300 $ RENAME FOR008.DAT;* FOR008.BIN;*
13400 $ RENAME FOR009.DAT;* FOR009.BIN;*
13500 $ RENAME FOR013.DAT;* FOR013.BIN;*
13600 $ RENAME FOR014.DAT;* FOR014.BIN;*
13700 $ RENAME FOR015.DAT;* FOR015.BIN;*
13800 $ RENAME FOR033.DAT;* FOR033.BIN;*
13900 $ RENAME FOR034.DAT;* FOR034.BIN;*
14000 $ PRINT /FLAG_PAGE BLD2.COM
14100 $ DIR /SINCE='TIME'/SIZE/PRINTER
14200 $ PRINT /FLAG_PAGE *.DAT;*
14300 $ DUMP FOR008.BIN;1/PRINTER/BLUCKS=(START:1,END:5)
14400 $ DUMP FOR008.BIN;2/PRINTER/BLUCKS=(START:1,END:5)
14500 $ DUMP FOR009.BIN;1/PRINTER/BLUCKS=(START:1,END:5)
14600 $ DUMP FOR009.BIN;2/PRINTER/BLUCKS=(START:1,END:5)
14700 $ DUMP FOR013.BIN/PRINTER/BLUCKS=(START:1,END:5)
14800 $ DUMP FOR014.BIN/PRINTER/BLUCKS=(START:1,END:5)
14900 $ DUMP FOR015.BIN/PRINTER/BLUCKS=(START:1,END:5)
15000 $ DUMP FOR033.BIN/PRINTER/BLUCKS=(START:1,END:5)
15100 $ DUMP FOR034.BIN;1/PRINTER/BLUCKS=(START:1,END:5)
15200 $ DUMP FOR034.BIN;2/PRINTER/BLUCKS=(START:1,END:5)
15300 $ FIN:
15400 $ EXIT

```

Figure 4-1. DCL Procedure for AADS Simulation Run (3 of 3)

COMMON area using DMA. More information on the communication between the VAX-11/780 computer and the Intel 8086 microprocessor can be found in Reference 10. The command file used to transfer AADS software to the Intel 8086 microprocessor and run AADS and the AADS simulator has not been defined at the present time.

4.3.3 SYSTEM GENERATION

DCL procedures to compile source code and to create executable images for all AADS processes are available in command files. Compilation and image creation is described in detail for the input/output and executive processes. Compilation and image creation procedures for gyro and star data processes are similar in most respects.

Figures 4-2 through 4-5 show the procedure for compiling the source for the input/output (SENSIN, DCAPIN, and OUTPUT) processes. The first command file, [AADS.IO2]COMP.COM references files SENSIN.COM, DCAPIN.COM, or OUTPUT.COM in the [AADS.IO2] subdirectory, depending on the process name selected. Compilation of diagnostic test lines (D_LINES), and the amount of compiler output depends on the options selected during the execution of the procedure. Figures 4-6 through 4-9 show the command files under the subdirectory [AADS.IO2] that are used for creating executable images for SENSIN, DCAPIN, and OUTPUT processes. Again, the command file LINK.COM references SENSIN1.COM, DCAPIN1.COM, or OUTPUT1.COM, depending on the process selected. Generation of the LINK map is optional.

All the link-editor input files include [AADS.CM2]COMMON.OPT as an options file. Figure 4-10 shows this file. This file includes the object modules for the global COMMON areas using the cluster option, creating the virtual address space and forcing the page alignment necessary for the creation of temporary global COMMON areas on the VAX-11/780 computer.

```
100  $!  
200  $! COMMAND FILE FOR COMPILING ROUTINES FOR AADS IO PROCESSES,  
300  $! [AADS.IO2]COMP.COM  
400  $!  
500  $SET DEFAULT [AADS.IO2]  
600  $ASSIGN [AADS.CM2] LOGGL  
700  $ASSIGN [AADS.CM2] LOGIO  
800  $ASSIGN [AADS.CM2] LOGRIU  
900  $WRITE SYS$OUTPUT "PROCESS NAMES: SENSIN, DCAPIN, OUTPUT"  
1000 $INQUIRE PROCESS "ENTER PROCESS NAME"  
1100 $INQUIRE LISTOPT "DO YOU WANT A LISTING? (Y/N)"  
1200 $INQUIRE DOPT "DO YOU WANT DEBUG LINES? (Y/N)"  
1300 $IF .NOT. LISTOPT THEN GOTO B1  
1400 $IF .NOT. DOPT THEN GOTO A2  
1500 $A1:  
1600 $FOR/D_LINES/LIST=LPO: @'PROCESS'  
1700 $GOTO END  
1800 $A2:  
1900 $FOR/LIST=LPO: @'PROCESS'  
2000 $GOTO END  
2100 $B1:  
2200 $IF .NOT. DOPT THEN GOTO B2  
2300 $FOR/D_LINES @'PROCESS'  
2400 $GOTO END  
2500 $B2:  
2600 $FOR @'PROCESS'  
2700 GOTO END  
2800 $END: EXIT
```

Figure 4-2. Compilation of Input/Output Functions
File [AADS.IO2]COMP.COM

```
100  SENSIN,SIGYRO,SIFHST,TIMOBT,B8ADD
```

Figure 4-3. Compile Input File [AADS.IO2]SENSIN.COM

```
100  DCAPIN,DCAP,QLONG,STOEPH,DCAPIP,INPUT,INEPH,INFULL,-  
200  HEADER,STCDAT
```

Figure 4-4. Compile Input File [AADS.IO2]DCAPIN.COM

```
100  OUTPUT,OBOUT,HIPRI,REPORT,UPDRPT,OUTACT,RAWDAT,RPY,-  
200  TIMOBT,B8ADD
```

Figure 4-5. Compile Input File [AADS.IO2]OUTPUT.COM

ORIGINAL PAGE 19
OF POOR QUALITY

```
100  $!  
200  $! COMMAND FILE TO LINK-EDIT AADS IO PROCESSES,-  
300  [AADS.IO2]LINK.COM  
400  $! OPENGL IS A MACRO ROUTINE  
500  $!  
600  $SET DEFAULT [AADS.IO2]  
700  $WRITE SYS$OUTPUT "PROCESS NAMES: SENSIN, DCAPIN,-  
800  OUTPUT"  
900  $INQUIRE PROCESS "ENTER PROCESS NAME"  
1000 $INQUIRE MAPOPT "DO YOU WANT A LINK MAP? (Y/N) "  
1100 $IF .NOT. MAPOPT THEN GOTO B1  
1200 $A1:  
1300 $LINK/EXE=[AADS.BT2]'PROCESS'.EXE/MAP=LP0: @'PROCESS'1  
1400 $GOTO END  
1500 $B1:  
1600 $LINK/EXE=[AADS.BT2]'PROCESS'.EXE @'PROCESS'1  
1700 $END: EXIT
```

Figure 4-6. Link Command File [AADS.IO2]LINK.COM

```

100  SENSIN,SIGYRO,SIFHST,TIMOBT,B8ADD,OPENGL,-
200  [AADS.CM2]COMMON/OPT,COMMON/OPT

```

Figure 4-7. Link-Editor Input File [AADS.IO2]SENSIN1.COM

```

100  DCAPIN,DCAP,QLONG,STOEPH,DCAPIP,INPUT,INEPH,INFULL,-
200  HEADER,STCDAT,-
300  [AADS.CM2]COMMON/OPT

```

Figure 4-8. Link-Editor Input File [AADS.IO2]DCAPIN1.COM

```

100  OUTPUT,OBOUT,HIPRI,REPORT,UPDRPT,OUTACT,RAWDAT,-
200  TRPY,B8ADD,-
300  [AADS.EX2]TIMOBT,[AADS.CM2]COMMON/OPT

```

Figure 4-9. Link-Editor Input File [AADS.IO2]OUTPUT1.COM

ORIGINAL PAGE IS
OF POOR QUALITY

```
100 CLUSTER=GLOBAL,,, [AADS.CM2]GLOBAL
200 CLUSTER=ACTVLG,,, [AADS.CM2]ACTVLG
300 CLUSTER=CATLOG,,, [AADS.CM2]CATLOG
400 CLUSTER=SENSOR,,, [AADS.CM2]SENSOR
500 CLUSTER=FLIGHT,,, [AADS.CM2]FLIGHT
600 CLUSTER=STATE,,, [AADS.CM2]STATE
700 CLUSTER=STAT1,,, [AADS.CM2]STAT1
800 CLUSTER=STAT2,,, [AADS.CM2]STAT2
900 CLUSTER=STAT3,,, [AADS.CM2]STAT3
1000 CLUSTER=STAT4,,, [AADS.CM2]STAT4
```

Figure 4-10. Link-Editor Input Options File
[AADS.CM2]COMMON.OPT

The link-editor input file for the SENSIN process includes [AADS.IO2]COMMON.OPT as a second options file. This file is shown in Figure 4-11. This file includes the object module for the /RIU/ global COMMON using the cluster option.

Figures 4-12 through 4-16 show the command procedures used to compile the source code and to create the executable image for the EXEC process. All procedures are under the default directory [AADS.EX2]. The procedure in file COMP.COM (Figure 4-12) is used to compile the source code for the EXEC process and uses the input file EXEC.COM (Figure 4-13). Figure 4-14 shows the link command procedure in the file LINK.COM, which refers to the link input file EXECL.COM (Figure 4-15). The link procedure uses two input options file. The first file is [AADS.CM2]COMMON.OPT shown earlier in Figure 4-10. The second options file [AADS.EX2]COMMON.OPT (Figure 4-16) includes the object module for the /ADSGBL/ global COMMON area using the cluster option.

System creation of AADS, where AADS is to be executed on the Intel 8086 target microprocessor, is done by transferring the source code to the Intel MDS (development computer). There the code is compiled and link edited as modules in the Intel absolute object file format. The code, as modules, is then transferred back to the VAX-11/780 computer, from which it can be downloaded to the Intel 8086 microprocessor. The code transmission is done through 1200-band dialup lines. Figure 4-17 shows the system configuration used for system creation.

4.3.4 EXECUTION AND CREATION OF UTILITY PROGRAM

Figure 4-18 shows the command file used to link edit and execute the TMPCOMCRE utility program. This program is used to initialize the global COMMON areas used by AADS with block data values. TMPCOMCRE does this by creating the data

100 CLUSTER=RIU,,, [AADS.CM2]RIU

Figure 4-11. SENSIN Link-Editor Input Options File
[AADS.IO2]COMMON.OPT

```
00100 $!
00200 $! COMMAND FILE FOR COMPILING THE AADS EXECUTIVE,-
00300 [AADS.EX2]COMP.COM
00400 $!
00500 $SET DEFAULT [AADS.EX2]
00600 $ASSIGN [AADS.CM2] LOGGL
00700 $ASSIGN [AADS.CM2] LOGEX
00800 $ASSIGN [AADS.BT2] LOGBLD
00900 $WRITE SYS$OUTPUT "ROUTINES FOR PROCESS EXEC-
01000 WILL BE COMPILED"
01100 $INQUIRE LISTOPT "DO YOU WANT A LISTING? (Y/N)"
01200 $INQUIRE DOPT "DO YOU WANT DEBUG LINES? (Y/N)"
01300 $IF .NOT. LISTOPT THEN GOTO B1
01400 $IF .NOT. DOPT THEN GOTO A2
01500 $A1:
01600 $FOR/D-LINES/LIST=LP0: @EXEC,OCCULT
01700 $GOTO END
01800 $A2:
01900 $FOR/LIST-LP0: @EXEC,OCCULT
02000 $GOTO END
02100 $B1:
02200 $IF .NOT. DOPT THEN GOTO B2
02300 $FOR/D-LINES @EXEC,OCCULT
02400 $GOTO END
02500 $B2:
02600 $FOR @EXEC,OCCULT
02700 $END: EXIT
```

Figure 4-12. Compile Command File [AADS.EX2]COMP.COM

ORIGINAL PAGE IS
OF POOR QUALITY

```
00100  ACTLOG,CMDPRO,ENQUE,EXEC,INITL,SCHED,HMS,TIMEUP,TSKDNE,-
00200  UPSTAT,TASKS,BLDLOG,MAIN,FLAGS,CANCEL,TIMOB,T,B8ADD
```

Figure 4-13. Compile Input File [AADS.EX2]EXEC.COM

```
00100  $!
00200  $! COMMAND FILE TO LINK-EDIT THE AADS EXECUTIVE,-[AADS.EX2]LINK.COM
00300  [AADS.EX2]LINK.COM
00400  $! OPENGL IS A MACRO ROUTINE WHICH IS ALREADY COMPILED
00500  $!
00600  $SET DEFAULT [AADS.EX2]
00700  $WRITE SYS$OUTPUT "PROCESS EXEC WILL BE LINKED"
00800  $INQUIRE MAPOPT "DO YOU WANT A LINK MAP? (Y/N)"
00900  $IF .NOT. MAPOPT THEN GOTO B1
01000  $A1:
01100  $LINK/EXE=[AADS.BT2]EXEC.EXE/MAP=LP0: @EXEC1
01200  $GOTO END
01300  $B1:
01400  $LINK/EXE=[AADS.BT2]EXEC.EXE @EXEC1
01500  $END: EXIT
```

Figure 4-14. Link Command File [AADS.EX2]LINK.COM

ORIGINAL PAGE IS
OF POOR QUALITY

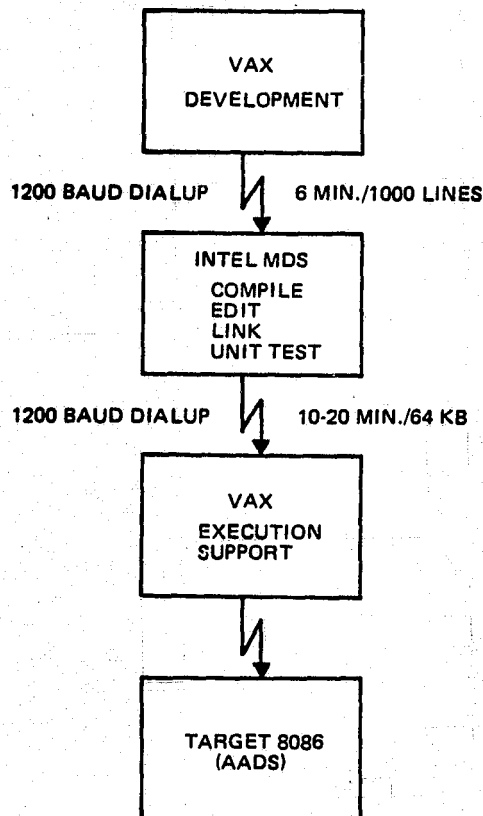
```
00100  ACTLOG,CMDPRO,ENQUE,EXEC,INITL,SCHED,HMS,TIMEUP,TSKDNE,-  
00200  UPSTAT,TASKS,BLDLOG,MAIN,FLAGS,CANCEL,TIMOB,T,B8ADD,-  
00300  OPENGL,OCCULT,-  
00400  [AADS,UT]MATMPY,UNVEC,SMPOS,[AADS.ST]INTERP,-  
00500  [AADS.CM2]COMMON/OPT,[AADS.EX2]COMMON/OPT
```

Figure 4-15. Link Input File [AADS.EX2]EXEC1.COM

```
00100  CLUSTER=ADSGBL,,,[AADS.CM2]ADSGBL
```

Figure 4-16. Link Input Options File [AADS.EX2]COMMON.OPT

ORIGINAL PAGE IS
OF POOR QUALITY



8581/82

Figure 4-17. VAX-Intel System Configuration Used for System Generation of Intel Executable Image of AADS

```
100 $ASSIGN [AADS.CM2] LOGGL
200 $FOR TMPCOMCRE
300 $LINK TMPCOMCRE,[AADS.EX2]OPENGL,[AADS.CM2]COMMON/OPT
400 $RUN TMPCOMCRE
```

**Figure 4-18. DCL Procedure To Link and To Execute
TMPCOMCRE Utility Program on VAX**

set [AADS.GL]COMMON.DAT. Block data values that were link edited into TMPCOMCRE are written to this data set. This data set is then used in the creation of the temporary global COMMON areas.

No command files exist for the star catalog generation utilities. Creation of these utilities is accomplished with a simple compilation and link edit. The remainder of this section contains execution instructions for these utilities.

The IBMVAX utility program converts a tape containing the star catalog from a IBM S/360-95 computer to a data set on the VAX computer.

Before running this utility program, the tape is mounted and the following DCL commands are executed:

```
MOUNT/FOREIGN/DENSITY=1600 MTA0
ASSIGN MTA0: FOR001
```

This utility program will create the direct access data set, MATCAT.DAT, on the logical unit FOR039. Additional output, formatted for viewing purposes, will be created on the logical unit FOR006.

The SORT utility program sorts the direct access star catalog data set created by IBMVAX. The star catalog entries are sorted by ascending right ascension angle values. Before running this utility program, the input data set is assigned to the logical unit FOR038, as in the following example:

```
ASSIGN MATCAT.DAT FOR038
```

Output will go to the logical unit FOR039. Additional output, formatted for viewing purposes, will go to logical unit FOR006. The output, FOR039.DAT, should be renamed to SORTED.SAV, if it is satisfactory. This data set is accessed later by the STARID process of AADS on unit FOR039 to initialize the star catalog COMMON area. This data set is

also used by the GSS process to obtain the SKYMAP star identification numbers for the stars identified by AADS.

4.4 AADS CONTROL PARAMETERS

AADS control parameters are divided into four broad types, as follows:

- Scheduling frequencies and output parameters
- Initial state, gyro data processing, and state propagation parameters
- Star data processing, star identification, and state update parameters
- Occultation prediction parameters

The parameter descriptions are taken from BLOCK DATA routines stored under the subdirectory [AADS.CM2]. The files STAT1.FOR, STAT2.FOR, STAT3.FOR, and STAT4.FOR should be consulted for the current descriptions of these variables.

In addition to the control parameters, AADS execution can be controlled (although at a higher level) by using such AADS commands as START, STOP, HALT, RESUME, and STOP. These commands are described in Section 2.

4.4.1 SCHEDULES AND PRIORITIES

VARIABLE -----	TYPE	DEFAULT	DESCRIPTION -----
UPDELT	R*4	29.75	TIME DIFFERENCE (SECONDS) BETWEEN THE SCHEDULED TIME OF THE LAST UPDATE AND THE START OF THE STAR DATA PROCESSING FOR THE NEXT STAR UPDATE.
GYDELT(2)	R*4		GYRO DATA PROCESSING TASK PERIOD(SECOND
(1)		0.512	NOMINAL PROCESSING PERIOD
(2)		0.256	PERIOD DURING A MANEUVER.
STDELT	R*4	0.512	SCHEDULING PERIOD FOR THE STAR DATA DATA PROCESSING TASK (SECONDS).
UPLIM	R*4	420.0	MAXIMUM TIME PERIOD WITHOUT A STATE UPDATE. AADS GENERATES AN ERROR MESSAGE IF THERE IS NO UPDATE FOR UPLIM SECONDS.

ORIGINAL PAGE IS
OF POOR QUALITY

SNDLT(2)	R*4		SCHEDULING PERIOD FOR THE SENSOR DATA COLLECTION TASK (SECONDS)
(1)		0.064	GYRO DATA SAMPLING PERIOD.
(2)		0.100	STAR DATA SAMPLING PERIOD. DATA FROM BOTH THE TRACKERS IS COLLECTED AT THE SAME TIME
IGYRP(2)	I*4		NUMBER OF TIMES GYRO DATA PROCESSING TASK INVOKED BEFORE SCHEDULING GYRO PROPAGATION TASK.
		1	DURING NOMINAL PROCESSING
		1	DURING MANEUVER
NPROUT(2)	I*4	1	NUMBER OF STATE PROPAGATIONS BEFORE REQUESTING DATA ANNOTATION REPORT.
(1)		1	DURING NORMAL PROCESSING
(2)		1	DURING MANEUVER
NUPOUT	I*4	1	NUMBER OF UPDATES BEFORE REQUESTING UPDATE REPORT
NATOUT	I*4	120	NUMBER OF PROPAGATIONS BEFORE REQUESTING ACTIVITY LOG OUTPUT
NRWOUT	I*4	120	NUMBER OF PROPAGATIONS BEFORE REQUESTING RAW SENSOR DATA OUTPUT
TSKPRI(10)	I*4		TASK PRIORITY INDICATOR PRIORITIES RANGE FROM 1 TO 31 WHERE 1 IS THE LOWEST AND 31 THE HIGHEST
		15	(1) - DCAP
		15	(2) - INPUT
		14	(3) - SENSIN
		10	(4) - GYPROC
		10	(5) - PROPAG
		6	(6) - STRACK
		6	(7) - STARID
		6	(8) - UPDATE
		8	(9) - OUTPUT
		12	(10) - EXEC
LEVDBG	I*4	10*0	LEVEL OF DEBUG OUTPUT FOR EACH TASK INDEXES ARE THE SAME AS TSKPRI ABOVE 0 MEANS NO DEBUG OUTPUT 10 MEANS MAXIMUM DEBUG OUTPUT

ORIGINAL PAGE 12
OF POOR QUALITY

4.4.2 INITIAL STATE AND TIME, AND GYRO PROCESSING PARAMETERS

VARIABLE	TYPE	DEFAULT	DESCRIPTION
-----	----	-----	-----
AQTIME	R*8	811001.03	TIME OF THE INITIAL STATE (YYMMDD,HHMMSS THE EXECUTIVE TASK WILL START SCHEDULING OTHER AADS TASKS AT THIS TIME.
AQUATR(4)	R*4		INITIAL STATE IN QUATERNION FORM AT TIME QTIME.
(1)		.062885	QUATERNION 1
(2)		-.168893	QUATERNION 2
(3)		-.055597	QUATERNION 3
(4)		.982054	QUATERNION 4
ASTCOV(6,6)	R*4		INITIAL COVARIANCE MATRIX AT AQTIME (UNCERTAINTY)**2 IN UNITS OF RADIAN**2
(1,1)		4.E-4	UNCERTAINTY**2 FOR QUATERNION 1
(2,2)		4.E-4	UNCERTAINTY**2 FOR QUATERNION 2
(3,3)		4.E-4	UNCERTAINTY**2 FOR QUATERNION 3
(4,4)		4.E-12	UNCERTAINTY**2 FOR DRIFT BIAS 1
(5,5)		4.E-12	UNCERTAINTY**2 FOR DRIFT BIAS 2
(6,6)		4.E-12	UNCERTAINTY**2 FOR DRIFT BIAS 3
			ALL OTHER ELEMENTS ARE ZERO.
ADBIAS(3)	R*4	3*0.0	INITIAL GYRO DRIFT RATE BIASES IN THE BODY FRAME AT QTIME (RADIAN/SEC)
(1)			X-AXIS BIAS
(2)			Y-AXIS BIAS
(3)			Z-AXIS BIAS
AGBIAS(6)	R*8	6*0.0	INITIAL GYRO DRIFT RATE BIASES AT THE TIME QTIME (DEG/SEC)
(I)			BIAS FOR THE I-TH CHANNEL, I=1 TO 6 CORRESPONDS TO CHANNELS A1, A2, B1, B2, C1, C2, RESPECTIVELY.
GC(3,3,8)	R*4		GYRO ALIGNMENT MATRICES FOR THE GYRO TO SPACECRAFT FRAME.
(I,J,K)			I=1,3; J=1,3 IS THE MATRIX FOR THE K-TH GYRO CONFIGURATION, K=1,8 ALL MATRICES ARE IDENTITY MATRICES.
GNOISE(2,6)	R*4		NOISE ASSOCIATED WITH THE GYRO MEASUREMENTS FOR ALL SIX CHANNELS.
(I,J)		3.E-9	GYRO RATE NOISE FOR THE J-TH CHANNEL (RADIAN**2/SECOND)
(2,J)		5.E-21	GYRO DRIFT RATE-RATE NOISE FOR THE J-TH CHANNEL (RADIAN**2/SECOND**3) THE CHANNEL ASSIGNMENT IS FOR J=1 TO 6, THE CHANNEL ARE A1, A2, B1, B2, C1, C2
DRPTOL	R*4		DATA DROPOUT TIME INTERVAL (SECONDS) FOR ACCUMULATING GYRO DATA. NOT USED.

ORIGINAL PAGE IS
OF POOR QUALITY

IGYLM(2,2)	I*4		LIMIT FOR GYRO COUNTS (COUNTS)
(1,1)		600	LOWER LIMIT, NOMINAL(LOW RATE) MODE.
(1,2)		800	UPPER LIMIT, NOMINAL(LOW RATE) MODE.
(2,1)		600	LOWER LIMIT, MANEUVER(HIGH RATE) MODE.
(2,2)		800	UPPER LIMIT, MANEUVER(HIGH RATE) MODE.
REDN(3)	R*4		TOLERANCES FOR REDUNDANT GYRO CHANNEL OUTPUT(RADIAN/SECOND). AADS SENDS AN ERROR MESSAGE IF THE DIFFERENCE IN RATE BETWEEN TWO SEPARATE CHANNELS ON THE SAME AXIS EXCEEDS THE TOLERANCE.
(1)		.48E-5	X AXIS: A1, C2 CHANNELS
(2)		.48E-5	Y AXIS: B1, A2 CHANNELS
(3)		.48E-5	Z AXIS: C1, B2 CHANNELS
			A,B,C ARE THE THREE, TWO AXIS GYROES A1,B1,C1 IS THE PRIMARY CONFIGURATION C2,A2,B2 IS THE BACKUP CONFIGURATION
IROLL	I*4		NOT USED.
MAXGYR	I*4	8388667	MAXIMUM GYRO COUNTS
IGYCON	I*4	1	GYRO CONFIGURATION
			1 = A1, B1, C1 : PRIMARY CONFIG.
			2 = A1, B1, B2
			3 = A1, A2, C1
			4 = A1, A2, B2
			5 = C2, B1, C1
			6 = C2, B1, B2
			7 = C2, A2, C1
			8 = C2, A2, B2
ICOTAB(3,9)	I*2	1 3 5	GYRO CHANNEL COMBINATION TABLE
		1 3 4	REPRESENTS THE 8 POSSIBLE CONFIGURATIONS
		1 2 5	LET THE CHANNELS A1, A2, B1, B2, C1, AND
		1 2 4	C2 BE REPRESENTED BY THE NUMBERS 1 THRU
		6 3 5	6, RESPECTIVELY. THEN THE GIVEN DEFAULTS
		6 3 4	REPRESENT THE 8 POSSIBLE CONFIGURATIONS
		6 2 5	WITH THE PRIMARY CONFIGURATION GIVEN BY
		6 2 4	IGYCON. THE BACKUP CONFIGURATION IS THE
		7 5 9	DIFFERENCE BETWEEN THE 9TH SUBARRAY AND
			THE PRIMARY CONFIGURATION.
			THIS PARAMETER IS INTERNALLY USED BY
			AADS. IT IS NOT A CONTROL PARAMETER
			AND SHOULD NOT BE CHANGED.
CGYRO(6,2)	R*4		CONVERSION CONSTANTS FOR CONVERTING
(I,1)			THE GYRO COUNTS TO ANGLE (DEGREE/COUNT)
			CONVERSION FACTOR FOR THE I-TH CHANNEL
			FOR THE LOW RATE MODE.
			0.1388889E-4 DEGREE/COUNT OR
			0.05 ARC-SECOND/SECOND
(T,2)			CONVERSION FACTOR FOR THE I-TH CHANNEL
			FOR THE HIGH RATE MODE.
			.2222222E-4 DEGREE/COUNT OR
			0.8 ARC-SECOND/SECOND.
			THE CHANNEL ASSIGNMENT IS I=1 TO 6
			FOR CHANNELS A1, A2, B1, B2, C1, C2,
			RESPECTIVELY.

4.4.3 STAR DATA PROCESSING AND STAR IDENTIFICATION PARAMETERS

VARIABLE -----	TYPE	DEFAULT -----	DESCRIPTION -----
RN(2,2)	R*8		MODEL MEASUREMENT NOISE(RADIAN). THIS IS THE LOWER BOUND ON THE NOISE. ACTUAL NOISE IS COMPUTED AS THE MAX OF RN AND THE GROUP CLUMP SIZE. SEE GCLUMP BELOW.
(1,J)		0.0001	MEASUREMENT NOISE FOR THE J-TH TRACKER(J=1,2) THAT IS USED FOR THE (X/Z) MODEL DURING UPDATE. HERE, X,Y,Z ARE THE 3 COMPONENTS OF THE OBSERVED UNIT STAR VECOTR.
(2,J)		0.0001	MEASUREMENT NOISE FOR THE J-TH TRACKER(J=1,2) THAT IS USED FOR THE (Y/Z) MODEL DURING STATE UPDATE.
TR(3,3,2) (I,J,1)	R*4		SPACECRAFT TO FHST ROTATION MATRICES (I=1,3), (J=1,3) ARE THE 9 ELEMENTS OF THE FHST1 TO BODY ROTATION MATRIX -.7865963 .6174676 -.3751331E-06 .1656866 .2110699 .9633263 .5948228 .7577488 -.2683329
(I,J,2)			(I=1,3), (J=1,3) ARE 9 ELEMENTS OF OF THE SPACECRAFT TO FHST2 MATRIXX .7865963 .6174676 0.0 .1656866 -.2110699 -.9633263 .5948228 .7577488 -.2683329
IHLIM(2,2)	I*4		UPPER/LOWER LIMITS FOR STAR H POSITION (COUNTS)
(1,1)		2047	UPPER LIMIT FOR FHST1
(2,1)		-2047	LOWER LIMIT FOR FHST1
(1,2)		2047	UPPER LIMIT FOR FHST2
(2,2)		-2047	LOWER LIMIT FOR FHST2
IVLIM(2,2)	I*4		UPPER/LOWER LIMITS FOR STAR V POSITION (COUNTS)
(1,1)		2047	UPPER LIMIT FOR FHST1
(2,1)		-2047	LOWER LIMIT FOR FHST1
(1,2)		2047	UPPER LIMIT FOR FHST2
(2,2)		-2047	LOWER LIMIT FOR FHST2
VILIM(2,2)	R*4		UPPER/LOWER LIMITS FOR STAR INTENSITY (VOLTS)
(1,1)		999.0	UPPER LIMIT FOR FHST1
(2,1)		.0001	LOWER LIMIT FOR FHST1
(1,2)		999.0	UPPER LIMIT FOR FHST2
(2,2)		.0001	LOWER LIMIT FOR FHST2

VTIIM(2,2) R*4

(1,1) 999.0
(2,1) .0001
(1,2) 999.0
(2,2) .0001

UPPER/LOWER LIMIT FOR STAR TEMPERATURE
(VOLTS)

UPPER LIMIT FOR FHST1
LOWER LIMIT FOR FHST1
UPPER LIMIT FOR FHST2
LOWER LIMIT FOR FHST2

CALHVT(19,2,2)R*4

(I,1,1)

I=1 TO 19, ARE 19 ELEMENTS FOR FHST1
FOR H COORDINATE.

CALHVT(2,1,1)=1., OTHER ELEMENTS ZERO

(I,1,2)

I=1,19 ARE THE 19 PARAMETERS FOR FHST2
FOR H COORDINATE.

CALHVT(2,1,2)=1., OTHER ELEMENTS ZERO

(I,2,1)

I=1,19 ARE 19 PARAMETERS FOR FHST1
FOR V COORDINATE

CALHVT(3,2,1)=1., OTHER ELEMENTS ZERO

(I,2,2)

I=1,19 ARE 19 PARAMETERS FOR FHST2
FOR V COORDINATE

CALHVT(3,2,2)=1., OTHER ELEMENTS ZERO.

CALHVI(19,2,2)R*4

STAR POSITION CALIBRATION PARAMETERS TO
CORRECT FHST DATA FOR INTENSITY

(I,1,1)

I=1 TO 19, ARE 19 ELEMENTS FOR FHST1
FOR H COORDINATE.

CALHVI(2,1,1)=1., OTHER ELEMENTS ZERO

(I,1,2)

I=1,19 ARE THE 19 PARAMETERS FOR FHST2
FOR H COORDINATE.

CALHVI(2,1,2)=1., OTHER ELEMENTS ZERO

(I,2,1)

I=1,19 ARE 19 PARAMETERS FOR FHST1
FOR V COORDINATE

CALHVI(3,2,1)=1., OTHER ELEMENTS ZERO

(I,2,2)

I=1,19 ARE 19 PARAMETERS FOR FHST2
FOR V COORDINATE

CALHVI(3,2,2)=1., OTHER ELEMENTS ZERO.

CTOVT(2,2) R*4

(1,J) 1.0
(2,J) 0.0

COUNT TO VOLT CONVERSION FOR TEMP-
RATURE ASSUMING

VOLTS = K1*COUNTS + K2, WHERE K1, AND
K2 ARE CONSTANTS

K1 IN (VOLT/COUNT), K2 IN (VOLT)
CONSTANT K1 FOR THE J-TH TRACKER, J=1,2
CONSTANT K2 FOR THE J-TH TRACKER, J=1,2

CTOVI(2,2) R*4

(1,J) 1.0

COUNT TO VOLT CONVERSION FOR STAR
INTENSITY ASSUMING

VOLTS = K1*COUNTS * K2, WHERE K1, AND
K2 ARE CONSTANTS.

K1 IN (VOLT/COUNT), K2 IN VOLT
CONSTANT K1 FOR THE J-TH TRACKER, J=1,2

ORIGINAL PAGE 11
OF POOR QUALITY

(2,J)	0.0	CONSTANT K2 FOR THE J-TH TRACKER, J=1,2
THETAH(2,2)	R*4	STAR CAMERA H POSITION COUNTS TO DEGREE CONVERSION CONSTANTS ASSUMING DEGREE = K1*COUNT + K2 K1 IN (DEGREE/COUNT), K2 IN (DEGREE)
(1,J)		CONSTANT K1 FOR THE J-TH TRACKER, J=1,2 (1,J) = 3.410525E-5
(2,J)	0.0	CONSTANT K2 FOR THE J-TH TRACKER, J=1,2
THETA V(2,2)	R*4	STAR CAMERA V POSITION COUNTS TO DEGREE CONVERSION CONSTANTS ASSUMING DEGREE = K1*COUNTS + K2
(1,J)		CONSTANT K1 FOR THE J-TH TRACKER, J=1,2 K1 IN (DEGREE/COUNT), K2 IN (DEGREE)
(2,J)	0.0	(1,J) = 3.410525E-5 CONSTANT K2 FOR THE J-TH TRACKER, J=1,2
TOLI(2)		TOLERANCE FOR THE SUCCESSIVE DIFFERENCE IN INTENSITY (VOLTS) DURING STAR GROUP FORMATION.
(1)	5.0	FHST1
(2)	5.0	FHST2
		NOT USED.
TOLP(2,2)	R*4	TOLERANCE FOR THE SUCCESSIVE DIFFERENCE IN POSITION (COUNTS) A NEW GROUP IS STARTED WHEN THE DIFF. EXCEEDS THE TOLERANCE NOTE THAT NEW GROUPS ARE ALWAYS STARTED AFTER AN UPDATE, EVEN IF THE TOLP TOLERANCE CAN BE SATISFIED.
(1,J)	50.0	TOLERANCE FOR H COUNTS FOR THE J-TH TRACKER, J=1,2
(2,J)	50.0	TOLERANCE FOR V COUNTS FOR THE J-TH TRACKER, J=1,2.
TTRK(2)	R*4	MAXIMUM TRACK GROUP TIME (SECONDS) NEW OBSERVATIONS FROM THE STAR ARE ARE REJECTED UNTIL NEXT UPDATE IF SAME STAR CONTINUES TO BE TRACKED AFTER TTRK SECONDS.
(1)	20.0	FHST1
(2)	20.0	FHST2
ITRK(2,2)	I*4	MINIMUM/MAXIMUM NUMBER OF TRACK POINTS NECESSARY FOR A TRACK GROUP FORMATION WHEN THE MAXIMUM POINTS ARE COLLECTED FOR THE CURRENT GROUP, AADS DISCARDS THE NEW STAR OBSERVATIONS IF SAME STAR CONTINUES TO BE TRACKED. SEE TOLP, TTRK ABOVE.
(1,J)	10.0	MINIMUM FOR THE J-TH TRACKER, J=1,2
(2,J)	80.0	MAXIMUM FOR THE J-TH TRACKER, J=1,2
TGMIN(2)	R*4	MINIMUM TIME BETWEEN TWO TRACKER GROUPS TO SEPERATE OBSERVED STARS (SECONDS).
(1)	0.3	TRACKER 1.
(2)	0.3	TRACKER 2.
TRS(2)	R*4	TIME INTERVAL OF CONSECUTIVE

ORIGINAL PAGE 15
OF POOR QUALITY

			UNACCEPTABLE RAW STAR TRACKER DATA BEYOND WHICH ERROR MESSAGE WILL BE SENT (SECONDS)
(1)		120.0	FHST1
(2)		120.0	FHST2
GCLUMP(2)	R*4		TOLERANCE FOR TRACK GROUP CLUMP SIZE A STAR GROUP IS NOT VALID IF THE OBSERVED CLUMP SIZE EXCEEDS THE TOLE- RANCE. THE OBSERVED CLUMP SIZE IS AN INDICATION OF THE TOTAL NOISE IN THE AVERAGE OBSERVED STAR VECTOR. THIS IS THE NOISE(ERROR) DUE TO STATE PROPAGA- TION PLUS THE NOISE IN THE STAR DATA. (DEG)
(1)		0.5	FHST1
(2)		0.5	FHST2
UNCID	I*4	0.2	ATTITUDE UNCERTAINTY CRITERIA FOR CHOOSING STAR IDENTIFICATION METHOD (DEG) COMPUTED UNCERTAINTY LESS THAN OR EQUAL TO UNCID, THEN USE DIRECT STAR MATCHING METHOD COMPUTED UNCERTAINTY GREATER THAN UNCID, THEN USE THE PAIRWISE STAR MATCHING METHOD
ISTRMN	I*4	1	MINIMUM NUMBER OF TRACK GROUPS FOR ATTEMPTING LAST-MINUTE STAR IDENTIFICATION OR UPDATE
ISTRDS(2)	I*4		DESIRED NUMBER OF TRACK GROUPS FOR ATTEMPTING IDENTIFICATION
(1)		1	FOR DIRECT IDENTIFICATION METHOD
(2)		3	PAIRWISE IDENTIFICATION METHOD
TOLINT(2)			STAR IDENTIFICATION MAGNITUDE TOLERANCE TO COMPARE THE STAR MAGNITUDES FROM THE STAR CATALOG AND OBSERVATIONS
(1)		0.5	TRACKER 1
(2)		0.5	TRACKER 2
SMAG(2,2)	R*4		STAR INTENSITY TO MAGNITUDE CONVERSION ASSUMING MAGNITUDE = K1*LOG(INTENSITY) + K2.
(1,J)		-1.0	CONSTANT K1 FOR THE J-TH TRACKER, J=1,2
(2,J)		5.0	CONSTANT K2 FOR THE J-TH TRACKER, J=1,2
DMWIND	R*4	0.05	DIRECT MATCH OBSERVATION TOLERANCE OR WINDOW SIZE (DEGREES). ALL STARS IN A WINDOW CENTERED ON THE AVERAGE OBSERVED STAR VECTOR ARE CANDIDATES
DTSUN	R*4	1.0	INTERVAL FOR COMPUTING AN AVERAGE SUN VECTOR FOR ABERRATION CORRECTION(SECONDS) SV(T) = SUN VECTOR AT TIME T IS GIVEN BY SV(T) = (SV(T+DTSUN)+SV(T-DTSUN))/2*DTSUN
DISTP	R*4	0.5	MINIMUM SEPARATION BETWEEN THE INITIAL PAIR DURING PAIRWISE MATCHING(DEGREE).

ORIGINAL PAIRING
OF POOR QUALITY

A PAIR OF OBSERVED STARS WILL BE INITIALLY SELECTED AND IDENTIFIED. THE REMAINING OBSERVED STARS WILL THEN BE MATCHED AGAINST THESE TWO STARS. DISTP IS THE MINIMUM SEPERATION REQUIRED FOR THE FIRST PAIR.

PMWIND	R*4	2.0	WINDOWSIZE FOR PAIRWISE MATCHING(DEG) SEE DMWIND ABOVE.
MDMW	I*4	2	MULTIPLICATION FACTOR FOR COMPUTING DIRECT MATCH WINDOW SIZE USING THE CURRENT ATTITUDE UNCERTAINTY (NO UNITS)
MPMW	I*4	2	MULTIPLICATION FACTOR FOR COMPUTING PAIR-WISE MATCH WINDOW SIZE(PMWIND) USING ATTITUDE UNCERTAINTY (NO UNITS)
PTOL	R*4	0.2	PAIRWISE MATCHING ARC-LENGTH TOLERANCE. ARC-LENGTH SEPARATION IS COMPUTED FOR A PAIR OF OBSERVED STARS. ARC-LENGTH IS ALSO COMPUTED FOR A CANDIDATE PAIR CORRESPONDING TO THE OBSERVED PAIR. THE CANDIDATE PAIR IS REJECTED IF THE DIFFERENCE IN ARC LENGTH EXCEEDS THE TOLERANCE. (DEGREE)
IUNIQD	I*4	0	UNIQUE MATCH REQUIREMENT FLAG FOR DIRECT MATCH IDENTIFICATION(NO UNIT) =0, UNIQUE MATCH NOT REQUIRED =1, UNIQUE MATCH REQUIRED
IUNIQP	I*4	0	UNIQUE MATCH REQUIREMENT FLAG FOR PAIR-WISE MATCH IDENTIFICATION(NO UNIT) =0, UNIQUE MATCH NOT REQUIRED =1, UNIQUE MATCH REQUIRED
IDOUBD	I*4	0	FLAG FOR USING DOUBLET STARS(NO UNIT) DURING STAR IDENTIFICATION =0, NO =1, YES
MNUMD	I*4	1	MINIMUM NUMBER OF IDENTIFIED STARS FOR DIRECT IDENTIFICATION TO BE SUCCESSFUL (NO UNIT)
MNUMP	I*4	3	MINIMUM NUMBER OF IDENTIFIED STARS FOR PAIR-WISE IDENTIFICATION TO BE SUCCESSFUL (NO UNIT)
PCTD	R*4	500	MINIMUM PERCENTAGE OF IDENTIFIED STARS FOR DIRECT MATCH TO BE SUCCESSFUL
PCTP	R*4	70.0	MINIMUM PERCENTAGE OF IDENTIFIED STARS FOR PAIR-WISE MATCH TO BE SUCCESSFUL ISTRDS(2)=3, MNUMP=3, PCTD=70.0 THAT AADS WILL USE A MINIMUM OF 3 STARS FOR PAIRWISE MATCH

4.4.4 OCCULTATION PREDICTION PARAMETERS

VARIABLE	TYPE	DEFAULT	DESCRIPTION
CS'IN(8)	R*8		CONSTANTS FOR SOLAR EPHEMERIS CALCULATION
CMOON(8)	R*8		CONSTANTS FOR LUNAR EPHEMERIS
RADIUS(2)	R*4		RADII (KILOMETERS)
(1)		6378.4	EARTH
(2)		1736.2	MOON
OBIAS(2)	R*4		BIAS ON THE RADIUS (KILOMETERS)
(1)		100.0	EARTH
(2)		20.0	MOON
FOVST(2)	R*4		FIELD OF VIEW OF THE STAR CAMERA (DEGREE FOR THE I-TH STAR CAMERA, I=1,2
(1)		4.0	
BIAS(2)	R*4		BIASES ON THE FIELD OF VIEW (DEGREE BIAS FOR THE I-TH STAR TRACKER, I=1,2
(1)		0.2	
MODEGY	I*4	1	DUMMY VARIABLE USED IN THE CREATION OF TEMPORARY GLOBAL COMMON AREAS. THIS VARIABLE MUST BE THE LAST ONE IN THE CLUSTER FOR PROPER MAPPING

4.5 AADS ERROR MESSAGES

AADS may generate error messages during execution. These error messages are of two types: critical and noncritical. In both cases, AADS takes a predefined action (error handling) and continues execution. For critical errors (for example, star identification not possible), ground action in the form of uplink of new values for AADS control variables is expected. The errors are transmitted to the ground as messages. The message header portion contains the error number and the data portion may contain one or more numbers, depending on the specific error. Error text is stored on the ground in a data set on the VAX to save AADS space and limit the message length. GSS decodes the error message, selects error text corresponding to the error number from the error message data set, and displays it on the output device. The error text explains the nature of the error and

the optional numbers contained in the error message give specific information about the error.

For example, the error message 601 contains the numbers 720.01 and 2. The error text printed by GSS explains that the star tracker 2 did not see any stars for the last 720.01 seconds. If the ground control repeatedly receives an error message indicating that stars cannot be identified, the operator uplinks a new estimate of the spacecraft attitude because the current estimate of the spacecraft attitude is incorrect for some reason.

AADS error message numbers are allocated by processes as follows:

<u>Error Number</u>	<u>Process</u>
000 to 099	EXEC
100 to 199	DCAP
200 to 299	INPUT
300 to 399	SENSIN
400 to 499	GYPROC
500 to 599	PROPAG
600 to 699	STRACK
700 to 799	STARID
800 to 899	UPDATE
900 to 999	OUTPUT

4.5.1 SYSTEM EXECUTIVE MESSAGES

Number 000: In routine UPSTAT. A command was received and was considered to be invalid. The AADS was already in the processing mode directed by the command, or the command number is out of range. This is a noncritical error. The executive process will continue in the current processing mode.

<u>Byte</u>	<u>Type</u>	<u>Description</u>
1	I*4	Error number
5	R*8	Current simulation time (YYMMDD.HHMMSS)
13	I*4	Command number. Valid numbers are: 1--START 2--STOP 3--HALT 4--RESUME

Number 001: In routine TIMEUP. The time limit for an attitude update was exceeded. This is a critical error. The executive process will continue attempting an attitude update.

<u>Byte</u>	<u>Type</u>	<u>Description</u>
1	I*4	Error number
5	R*8	Current simulation time (YYMMDD.HHMMSS)

Number 002: In routine TIMEUP. A process scheduled to execute at this time has not completed or returned from its previous execution. This is a noncritical error. The executive will schedule the process for its next execution.

<u>Byte</u>	<u>Type</u>	<u>Description</u>
1	I*4	Error number
5	R*8	Current simulation time (YYMMDD.HHMMSS)
13	I*4	Number of scheduled process: 3--SENSIN 4--GYPROC 5--PROPAG 6--STRACK 7--STARID 8--UPDATE

Number 003: In routine TSKDNE. A completion of a maneuver was detected and the star data processing is in a halted mode. This is a noncritical error. The executive will continue normal gyro processing and propagation.

<u>Byte</u>	<u>Type</u>	<u>Description</u>
1	I*4	Error number
5	R*8	Time or error

4.5.2 INPUT/OUTPUT MESSAGES

Number 100: In routine DCAP. Data with an invalid synchronization (sync) byte(s) was read. This is a noncritical error. The data capture process ignores the data and performs another read on the line.

<u>Byte</u>	<u>Type</u>	<u>Description</u>
1	I*4	Error number
5	R*8	Time tag on erroring record (YYMMDD.HHMMSS)
13	I*4	Record number in transmission from header of erroring record
17	L*1	Value of first sync byte
18	L*1	Value of second sync byte

Number 101: In routine QLONG. Uplinked data was lost due to a full queue. This is a noncritical error. The input process will validate and store the data that was successfully queued.

<u>Byte</u>	<u>Type</u>	<u>Description</u>
1	I*4	Error number
5	R*8	Time tag on lost record (YYMMDD.HHMMSS)
13	I*2	Record number in block
15	I*2	Record number in transmission
17	I*2	Block number in transmission

Number 102: In routine DCAP. An unidentified command or data was read. The erroring record has an invalid INTYPE value (and valid sync byte values). This is a noncritical error. The data capture process ignored the data and performs another read on the line.

<u>Byte</u>	<u>Type</u>	<u>Description</u>
1	I*4	Error number
5	R*8	Time tag on erroring record (YYMMDD.HHMMSS)
13	I*4	Record number in transmission from header of erroring record
17	L*1	Erroring INTYPE value

Number 103: In routine STOEPH. Ephemeris data with invalid header variables was read. Either the record number in the transmission, or the number of ephemeris points, is invalid. This is a noncritical error. This record will be ignored and any data successfully queued prior to this will be validated by the input process.

<u>Byte</u>	<u>Type</u>	<u>Description</u>
1	I*4	Error number
5	R*8	Time tag on erroring record (seconds since 9-1-57)
13	I*4	Record number in transmission from header of erroring record
17	I*4	Number of ephemeris points

Number 201: In routine INEPH. The magnitude of ephemeris data that was read is invalid. This is a noncritical error. Ephemeris elements that passed validation prior to this will be kept. The erroring ephemeris element and any following ephemeris elements will be discarded.

<u>Byte</u>	<u>Type</u>	<u>Description</u>
1	I*4	Error number
5	R*8	Time of erroring ephemeris element (in seconds since 9-1-57)
13	R*4	Position vector magnitude
17	R*4	Velocity vector magnitude

Number 202: In routine HEADER. The running record or block number in the transmission is invalid. This is noncritical error. The uplink data is discarded.

<u>Byte</u>	<u>Type</u>	<u>Description</u>
1	I*4	Error number
5	R*8	Time tag on erroring record (YYMMDD.HHMMSS)
13	I*2	Expected block number
15	I*2	Expected record number in transmission
17	I*2	Received block number
19	I*2	Received record number in transmission

Number 203: In routine HEADER. The running record number in the block is invalid. This is a noncritical error. The uplink data is discarded.

<u>Byte</u>	<u>Type</u>	<u>Description</u>
1	I*4	Error number
5	R*8	Time tag on erroring record (YYMMDD.HHMMSS)
13	I*2	Expected record number in block
15	I*2	Received record number in block
17	I*2	Received running block number in transmission
19	I*2	Received running record number in transmission

Number 204: In routine HEADER. The number of bytes of data indicated in the header exceeds the maximum. This is a noncritical error. The uplink data is discarded.

<u>Byte</u>	<u>Type</u>	<u>Description</u>
1	I*4	Error number
5	R*8	Time tag on erroring record (YYMMDD.HHMMSS)
13	I*2	Number of bytes of data
15	I*2	Running record number in block
17	I*2	Running block number in transmission
19	I*2	Running record number in transmission

Number 205: In routine HEADER. The time tags on received records are not sequential. This is a noncritical error. The uplinked data is discarded.

<u>Byte</u>	<u>Type</u>	<u>Description</u>
1	I*4	Error number
5	R*8	Time tag of erroring record (YYMMDD.HHMMSS)
13	R*8	Time tag of previous record (YYMMDD.HHMMSS)

Number 206: In routine INFULL. A COMMON area was overfilled with uplinked data. This is a noncritical error. The common area following the one indicated may be damaged.

<u>Byte</u>	<u>Type</u>	<u>Description</u>
1	I*4	Error number
5	R*8	Time tag on last record (YYMMDD.HHMMSS)
13	I*4	Number indicating the last overfilled STATIC COMMON area: 1--STAT1 2--STAT2 3--STAT3 4--STAT4

4.5.3 GYRO DATA PROCESSING MESSAGES

Number 401: In routine GYPROC. Power off detected in some channels. This is a critical error if more than one channel power is off. Otherwise, GYPROC will continue execution.

<u>Byte</u>	<u>Type</u>	<u>Description</u>
1	I*4	Error number
5	R*8	Error time (YYMMDD.HHMMSS)
13	I*2	Error code: = 1, channel 1 off = 2, channel 2 off = 3, channel 3 off = 5, channels 1,2 off = 6, channels 1,3 off = 9, channels 2,3 off = 18, channels 1,2,3 off

Number 402: In routine GYPROC. Saturation detected for channels. GYPROC will use the saturation value to compute body rates and continue execution.

<u>Byte</u>	<u>Type</u>	<u>Description</u>
1	I*4	Error number
5	R*8	Error time (YYMMDD.HHMMSS)
13	I*2	Number of saturations in first given channel
15	I*2	Number of saturations in second given channels
17	I*2	Number of saturations in third given channel

Number 403: In routine GYPROC. Glitch (large noise pulse) detected. This is a noncritical error. GYPROC will not use the noisy data, but will continue execution.

<u>Byte</u>	<u>Type</u>	<u>Description</u>
1	I*4	Error number
5	R*8	Error time (YYMMDD.HHMMSS)
13	I*2	Error code. See the error codes given for error 401

Number 404: In routines GYPROC and PROPAG. Invalid gyro configuration number detected. This is a critical error. GYPROC/PROPAG will not process data. Uplink new valid gyro configuration number between 1 and 9.

<u>Byte</u>	<u>Type</u>	<u>Description</u>
1	I*4	Error number
5	I*4	Gyro configuration number

Number 501: In routine PROPAG. The difference in rate between the primary and the secondary channels exceeds the specified tolerance. PROPAG will continue processing data but the state propagation may give incorrect attitude.

<u>Byte</u>	<u>Type</u>	<u>Description</u>
1	I*4	Error number
5	R*8	Error time (YYMMDD.HHMMSS)
13	I*2	Gyro channel number

Number 502: In routine PROPAG. The incremental time for the given channel was zero, probably due to invalid gyro data for that channel. PROPAG will use the last valid rate and continue execution.

<u>Byte</u>	<u>Type</u>	<u>Description</u>
1	I*4	Error number
5	R*8	Error time (YYMMDD.HHMMSS)
13	I*2	Channel number

Number 503: In routine PROPAG. The propagation time is zero. PROPAG will continue execution, but will use the last valid gyro rates.

<u>Byte</u>	<u>Type</u>	<u>Description</u>
1	I*4	Error number
5	R*8	Error time (YYMMDD.HHMMSS)
13	R*8	Propagation time in seconds

4.5.4 STAR DATA PROCESSING MESSAGES

Number 601: In routine MAGN. Star tracker 1 or 2 invalid data time interval exceeds specified tolerance. This is not a critical error and AADS will continue processing. The error has probably occurred because there are no stars in the field of view (most likely), or the star data are failing limit checks. Check the limits on H and V counts and observed intensity.

<u>Byte</u>	<u>Type</u>	<u>Description</u>
1	I*4	Error number
5	R*8	Last valid data time (YYMMDD.HHMMSS)
13	R*4	Invalid data interval (seconds)

There are no error messages in the STARID process.

Number 801: In routine UPDATE. Error detected during state update. Execution will continue, but the current star observation will not be used.

<u>Byte</u>	<u>Type</u>	<u>Description</u>
1	I*4	Error number
5	R*8	Error time (YYMMDD.HHMMSS)
13	I*2	= 1, Denominator zero in computing model residuals = 2, Number of elements in the state vector is incorrect

4.6 AADS MESSAGE FORMATS

AADS communicates with the external world by means of messages. A message usually contains a header portion followed by a data portion. The header contains flags describing data that follow. In addition, the header contains a sync (synchronization) byte and other accounting information to enable detection of errors errors in message transmission.

The majority of messages is 256 bytes in length. However, the annotation report and the ephemeris request are 40 bytes long and do not have a standard header. The uplink messages are transmitted by GSS to OBC(OSS) and then retransmitted by OBC(OSS) to AADS. AADS output also goes to OBC(OSS); OBC(OSS) then retransmits the AADS messages to the ground, except for the ephemeris request, which is used by OBC(OSS) to generate the requested ephemeris. GSS decodes the AADS output and generates formatted, hardcopy output for operator evaluation. The following terms are used to describe message formats:

- Header--Most significant bytes of the message (usually 20) containing the data type information, sync bytes, and other accounting information.

- Record--A 20-byte header plus data plus filler to comprise 256 bytes. Words record and message are used interchangeably. Filler bytes contain binary 1 bits.
- Block--A collection of records. For example, the update report is sent as a block of three records.
- Transmission--A collection of two or more blocks in which the last block is the end-of-transmission block (EOT). EOT contains binary 1 bits.

The following pages describe the various message formats in detail.

4.6.1 GENERAL HEADER (UPLINK/DOWNLINK) FORMAT

The header size is 20 bytes long and is used for all uplink messages. It is also used for the raw sensor data report, the update report, and the activity log report. The general header format is given below. Byte 1 means the most significant byte.

<u>Variable</u>	<u>Byte Number</u>	<u>Type</u>	<u>Value</u>	<u>Description</u>
SYNC1	1	Byte	88	Sync byte (later spacecraft ID)
INTYPE	2	Byte		Type of input: = 1, data = 2, command
INDATA	3	Byte		Type of data or command
NBLOCK	4	Byte		Running number of this record in block
IAREA	5	Byte		Data area number for insertion or report
SYNC2	6	Byte	103	Second sync byte
NTRAN	7 to 8	I*2		Running number of this record in transmission
IBLOCK	9 to 10	I*2		Running number of this block in transmission

<u>Variable</u>	<u>Byte Number</u>	<u>Type</u>	<u>Value</u>	<u>Description</u>
NSIZE	11 to 12	I*2		Number of bytes used in the data portion of the record
TTRAN	13 to 20	R*8		Time of transmission (GMT) YYMMDD.HHMMSS

4.6.2 COMMANDS AND UPLINK DATA

GSS sends commands to AADS with a standard header (see Section 4.6.1) with INTYPE = 2 and INDATA = 3. The command number is contained in the first four bytes (I*4 variable ICMD) following the header. The following commands are specified:

ICMD = 1--START
 ICMD = 2--STOP
 ICMD = 3--HALT
 ICMD = 4--RESUME

OBC(OSS) receives all uplink messages from GSS, and retransmits appropriate messages to AADS. GSS uplinks new tables (new values for AADS control parameters), using a standard header with INTYPE = 1. The new values are stored in AADS COMMON areas using INDATA and NSIZE values in the header. For example, INDATA = 2 and NSIZE = 236 means to store 236 bytes from the data area of the message in the COMMON area /STAT3/. Because the whole COMMON area must be modified during a single transmission, there will be a block of three consecutive records in the transmission with INDATA = 2. Records within the block must be in the proper order to ensure a correct update. The data area from the last record in the block may be partially filled (NSIZE \leq 236). If all four AADS COMMON areas are updated, the transmission to AADS will consist of four blocks, each block containing one or more records. The header for each record will have INTYPE = 1 and the proper value of INDATA as given

below. Finally, the record with INDATA = 5 is used by OBC(OSS), and not retransmitted to AADS.

<u>INDATA</u>	<u>INTYPE</u>	<u>Description</u>
1	1	Gyro processing parameters (/STAT2/ global COMMON values)
2	1	Star processing parameters (/STAT3/ global COMMON values)
3	2	Commands to AADS
4	1	Process scheduling parameters and priorities (/STAT1/ global COMMON values)
5	1	Orbital elements (to OSS only)
6	1	Occultation prediction parameters (/STAT4/ global COMMON values)
20	1	Error messages from AADS to GSS

4.6.3 SPACECRAFT EPHEMERIS FROM OSS TO AADS

AADS periodically makes requests for spacecraft ephemeris to OBC(OSS). The format of this request is described later in this section. OBC(OSS) generates the requested ephemeris and sends it to AADS as a transmission containing six records followed by the EOT record. Each record has a header with INTYPE = 5. In addition, the header for each record contains the SYNC1 byte and the running number of the record (ENTRAN), but does not contain other variables in the standard header. Instead, the header contains the ephemeris start time, number of points, and time interval between points for verification. Nominally, the ephemeris transmission contains 20 position and velocity vectors each, and can contain a maximum of 29 of these vectors. The data area of records 1 to 3 contain the spacecraft position vector components and the data area of records 4 to 6 contain the velocity vector components. Again, the last (seventh) record of the transmission is the EOT record containing all binary 1 bits. The detailed format of the ephemeris transmission is given below.

<u>Variable</u>	<u>Byte Number</u>	<u>Type</u>	<u>Value</u>	<u>Description</u>
SYNC1	1	Byte	88	Sync number (later space- craft ID)
INTYPE	2	Byte	5	Type of input, = 5 for ephemeris
ENTRAN	3 to 4	I*2		Running number of this rec- ord in transmission
ST	5 to 12	R*8		Start time of ephemeris (seconds since 9/1/57)
NOPTS	13 to 16	I*4		Number of data points in a record
INTERVAL	17 to 20	R*4		Time between data points (seconds)

<u>Variable</u>	<u>Byte Number</u>	<u>Type</u>	<u>Value</u>	<u>Description</u>
<u>Record 1</u>				
X(1)	21 to 28	R*8		X coordinate of first data point
X(2)	29 to 36	R*8		X coordinate of second data point
⋮				
X(N)		R*8		X coordinate of nth data point
<u>Record 2</u>				
Y(1)	21 to 28	R*8		Y coordinate of first data point
Y(2)	29 to 36	R*8		Y coordinate of second data point
⋮				
Y(N)		R*8		Y coordinate of nth data point

Record 3

Z(1)	21 to 28	R*8	Z coordinate of first data point
Z(2)	29 to 36	R*8	Z coordinate of second data point
⋮			
Z(N)		R*8	Z coordinate of nth data point

Record 4

X(1)	21 to 28	R*8	X component of velocity of first point
X(2)	29 to 36	R*8	X component of velocity of second point
⋮			
X(N)		R*8	X component of velocity of nth point

Record 5

Y(1)	21 to 28	R*8	Y component of velocity of first point
Y(2)	29 to 36	R*8	Y component of velocity of second point
⋮			
Y(N)		R*8	Y component of velocity of nth point

Record 6

Z(1)	21 to 28	R*8	Z component of velocity of first point
Z(2)	29 to 36	R*8	Z component of velocity of second point
⋮			
Z(N)		R*8	Z component of velocity nth point

4.6.4 EPHEMERIS REQUEST FROM AADS TO OBC(OSS)

AADS periodically makes ephemeris requests to AADS. The ephemeris request is 40 bytes long and is nominally generated by the OUTPUT process every 20 minutes. The request has a header with a SYNC1 byte and has INTYPE = 4 for ephemeris request. In addition, the message contains the requested start time for the ephemeris, number of points requested, and time interval between points. Nominally, 20 points are requested at 1-minute intervals. A maximum of 29 points can be requested. The detailed format for the ephemeris request is given below.

<u>Variable</u>	<u>Byte Number</u>	<u>Type</u>	<u>Value</u>	<u>Description</u>
SYNC1	1	Byte	88	Sync byte (later spacecraft ID)
INTYPE	2	Byte	4	Type of data, = 4 for ephemeris request
ST	3 to 10	R*8		Start time in seconds since 9/1/57
NOPTS	11 to 14	I*4		Number of data points in a record
INTERVAL	15 to 18	R*4		Time between data points in seconds
Filler	19 to 40	22 bytes		Filler

4.6.5 DATA ANNOTATION REPORT

This report contains current time and the spacecraft attitude in quaternion and pitch, roll, and yaw forms. AADS downlinks this report at a high frequency (highest frequency once per 0.512 seconds). The record length is 40 bytes. It contains the SYNC1 byte and has INTYPE = 3, but does not contain other variables from a standard header. The detailed format of the annotation report is given below.

<u>Variable</u>	<u>Byte Number</u>	<u>Type</u>	<u>Value</u>	<u>Description</u>
SYNCl	1	Byte	88	Sync number (later spacecraft ID)
INTYPE	2	Byte	3	Type of data, = 3 for annotation data
TIME	3 to 10	R*8		Time in seconds since 9/1/57
Q1	11 to 14	R*4		First quaternion
Q2	15 to 18	R*4		Second quaternion
Q3	19 to 22	R*4		Third quaternion
Q4	23 to 26	R*4		Fourth quaternion
P	27 to 30	R*4		Pitch (degrees)
R	31 to 34	R*4		Roll (degrees)
Y	35 to 38	R*4		Yaw (degrees)
Filler	39 to 40	2 bytes		Filler

4.6.6 UPDATE REPORT

This report is generated after one or more updates and gives the current attitude in quaternion form and the drift rate biases. It gives a measure of star availability by the number of stars used for the update and the total elapsed time between the first and the last identified star. The identification numbers for the identified star can be used to verify the star identification algorithm on the ground. Finally, the report gives the last state covariance matrix and Kalman gain matrix elements. These matrices provide a measure of the filter performance.

The update report consists of three records, each of which is 256 bytes long. The standard header for each record has INTYPE = 1, and INDATA = 1, 2, or 3, indicating the record number. The detailed update report format is given below.

<u>Variable</u>	<u>Byte Number</u>	<u>Type</u>	<u>Value</u>	<u>Description</u>
<u>Record 1</u>				
Header	1 to 20			As explained above
Q1	21 to 28	R*8		First quaternion
Q2	29 to 36	R*8		Second quaternion
Q3	37 to 44	R*8		Third quaternion
Q4	45 to 52	R*8		Fourth quaternion
DB1	53 to 60	R*8		Gyro drift rate bias in body frame, X-axis
DB2	61 to 68	R*8		Gyro drift rate bias in body frame, Y-axis
DB3	69 to 76	R*8		Gyro drift rate bias in body frame, Z-axis
IGYCON	77 to 80	I*4		Gyro configuration: = 1, A1, B1, C1 = 2, A1, B1, B2 = 3, A1, A2, C1 = 4, A1, A2, B2 = 5, C2, B1, C1 = 6, C2, B1, B2 = 7, C2, A2, C1 = 8, C2, A2, B2
STCOV	81 to 256	22 x R*8		First 22 elements of the latest covariance matrix
<u>Record 2</u>				
Header	1 to 20			As explained above
STCOV	21 to 132	14 x R*8		Last 14 elements of the latest covariance matrix
Filler	133 to 256			Filler
<u>Record 3</u>				
Header	1 to 20			As explained above
KLGAIN	21 to 44	6 x R*4		Kalman gain matrix after the update

<u>Variable</u>	<u>Byte Number</u>	<u>Type</u>	<u>Value</u>	<u>Description</u>
DELTA	45 to 48	R*4		Elapsed time between first and last identified stars
ELAPST	49 to 52	R*4		Elapsed time since last update
QTIME	53 to 60	R*8		Time of state (seconds since 9/1/57)
NSTARS	61 to 64	I*4		Number of stars used for update
ISTARS	65 to 68	I*4		Number of stars identified since last update
KSTAR	69 to 256	Up to 47 x I*4		Internal star catalog numbers (NSTARS x I*4)

NOTE: Gyro channels 1 through 6 are A1, A2, B1, B2, C1, C2, respectively. A1, B1, C1 is usually the primary channel configuration.

4.6.7 RAW DATA REPORT

The raw data report consists of four records. Each record is 256 bytes long and consists of a 20-byte standard header and 236-byte data portion. The value of INTYPE = 1 and INDATA = 5, 6, 7, or 8, respectively, for the four records. The report contains raw gyro and star tracker data used by AADS, and is generated after n state propagations or on a severe error in AADS. The detailed format of the raw data report is given below.

<u>Variable</u>	<u>Byte Number</u>	<u>Type</u>	<u>Value</u>	<u>Description</u>
<u>Record 1</u>				
Header	1 to 20			Standard header with INTYPE = 1 and INDATA = 5
STIME	21 to 84	8 x R*8		Time of last eight observations of star camera data in seconds since 9/1/57

<u>Variable</u>	<u>Byte Number</u>	<u>Type</u>	<u>Value</u>	<u>Description</u>
ATIME	85 to 88	R*4		Time offset for TRACKER2; data is offset from corre- sponding TRACKER1 time by a constant amount ATIME; is negative if TRACKER2 is sampled before TRACKER1
IHPOS	89 to 152	16 x I*4		Raw star tracker output for H coordinate (alter- nating trackers)
IVPOS	153 to 216	16 x I*4		Raw star tracker output for V coordinate (alter- nating trackers)
IPRES	217 to 248	16 x I*2		Star presence indicator: = 0, stars not present in the field of view = 1, star present in the field of view
Filler	249 to 256			Filler

Record 2

Header	1 to 20			Standard header with INTYPE = 1 and INDATA = 6
SINTEN	21 to 84	16 x R*4		Star intensity (counts); last eight observations (alternating star trackers)
TEMP	85 to 148	16 x R*4		Star camera temperature (counts) (alternating trackers)
STATUS	149 to 212	16 x I*4		Status flag for star trackers (alternating trackers)
Filler	213 to 256			Filler

Record 3

Header	1 to 20			Standard header with INTYPE = 1, and INDATA = 7
--------	------------	--	--	--

<u>Variable</u>	<u>Byte Number</u>	<u>Type</u>	<u>Value</u>	<u>Description</u>
GTIME	21 to 84	8 x R*8		Time in seconds since 9/1/57 for last 8 obser- vations
NGYRO	85 to 252	7 x 6 x I*4		Most recent seven samples from six gyro channels (counts)
Filler	253 to 256			Filler
<u>Record 4</u>				
Header	1 to 20			Standard header with INTYPE = 1 and INDATA = 8
NGYRO	21 to 44	6 x I*4		Raw gyro counts of eighth sample from six gyro channels
NGYST	45 to 140	8 x 6 x I*2		Gyro power on/off flag: = 0, power off = 1, power on For last eight samples from each of the six channels
MGYST	141 to 236	8 x 6 x I*2		Gyro rate mode flag: = 1, low rate mode = 2, high rate or maneuver mode For last eight samples from each of the six channels
Filler	237 to 256			Filler

NOTE: Gyro channels 1 to 6 are A1, A2, B1, B2, C1, and C2.

4.6.8 ACTIVITY LOG REPORT

The activity log consists of a single record 256 bytes long. The record consists of a 20-byte header and 236-byte data portion. The value of INTYPE = 1 and INDATA = 9. This report is generated after n propagations, or on a critical

error. It gives a brief summary of AADS activities and errors encountered during processing. The detailed format of the activity log is given below.

<u>Variable</u>	<u>Byte Number</u>	<u>Type</u>	<u>Value</u>	<u>Description</u>
Header	1 to 20			Standard header with INTYPE = 1 and INDATA = 9
STTIME	21 to 28	R*8		Time and date of START command in the form YYMMDD.HHMMSS
Spare	29 to 36	R*8		Not currently used
UPTRYS	37 to 40	I*4		Total number of state up- dates attempted
UPSUCC	41 to 44	I*4		Total number of successful state updates
EPHREQ	45 to 48	I*4		Total number of spacecraft ephemeris requests
GYSTATS	49 to 52	I*4		Total number of gyro data saturation
IDSTRS	53 to 56	I*4		Total number of identified stars
Filler	57 to 256			Filler bytes

4.6.9 AADS ERROR MESSAGES

The error message is 256 bytes long. It consists of a standard 20-byte header and 20-byte data portion. The value of INTYPE in the header is 1 and INDATA is 20 for error messages. The first four bytes in the data portion contain the error number. Bytes 25 through 32 usually contain the time when the error was noted. However, the actual formats of bytes 25 through 40 depend on the specific message (see Section 4.5). The error message format is given below.

<u>Variable</u>	<u>Byte Number</u>	<u>Type</u>	<u>Value</u>	<u>Description</u>
Header	1 to 20	-	-	Standard header with INTYPE = 1 and INDATA = 20

<u>Variable</u>	<u>Byte Number</u>	<u>Type</u>	<u>Value</u>	<u>Description</u>
NERROR	21 to 24	I*4	-	Error number
Data	25 to 40	-	-	Data associated with the specific error

4.6.10 SENSOR DATA FORMATS

In the VAX configuration, the SDS stores Landsat-D simulator generated sensor data in a global COMMON area. SDS does not queue these data but overwrites the entire buffer with new data each time SDS is invoked. The SENSIN process accesses this data area and performs initial decoding to obtain gyro/star camera counts and other flags. SENSIN does not perform a "READ" operation, but simply accesses the data from a buffer. This procedure may require changing later for the VAX-Intel configuration or when AADS is installed onboard a spacecraft. The sensor data formats are included here for completeness even though AADS does not perform a "READ" operation to get the data from the sensors.

Gyro data consists of six 4-byte packets. Each packet corresponds to a channel; otherwise, all packets are identical in format. The star camera data consists of two 7-byte packets. Each packet corresponds to a star camera. The formats for gyro and star camera data are given below.

<u>Bit Number^a</u>	<u>Byte Number</u>	<u>Description</u>
<u>Channel 1</u>		
1	1	On/off switch: = 0, off = 1, on
2	1	Mode indicator: = 0, low rate = 1, high rate

^aWithin bytes, bits are numbered from least significant bit (LSB) to most significant bit (MSB).

<u>Bit Number^a</u>	<u>Byte Number</u>	<u>Description</u>
3 to 8	1	Fill
9 to 32	2 to 4	Counts

Channel 2^b

^aWithin bytes, bits are numbered from least significant bit (LSB) to most significant bit (MSB).

^bChannels 2 through 6 continue in the same manner as Channel 1 ending at byte 24, bit 192.

<u>Bit Number^a</u>	<u>Byte Number</u>	<u>Description</u>
<u>Camera 1</u>		
1	1	On/off switch: = 0, off = 1, on
2	1	Presence flag: = 1, star is present = 0, no stars
3 to 8	1	Fill
9 to 20	2 to 3	H coordinate counts
21 to 32	3 to 4	V coordinate counts
33 to 40	5	Intensity counts
41 to 56	6 to 7	Temperature counts

Camera 2^b

^aWithin bytes, bits are numbered from LSB to MSB.

^bCamera 2 continues in the same manner as Camera 1 ending at byte 14, bit 112.

SECTION 5 - BASIC ALGORITHMS

This section describes the Kalman filtering scheme for state propagation and update using 3-axis gyro data and star tracker data. See Reference 8 for further details.

5.1 PROPAGATION

5.1.1 GYRO MODELS

The gyro output rate vector \vec{u} is related to the spacecraft angular velocity \vec{w} according to

$$\vec{u} = \vec{w} + \vec{b} + \vec{\eta}_1 \quad (5-1)$$

where \vec{b} is the drift rate bias and $\vec{\eta}_1$ is the drift rate noise. The vectors \vec{u} , \vec{w} , \vec{b} , and $\vec{\eta}_1$ are in the spacecraft (body) coordinate frame. $\vec{\eta}_1$ is assumed to be a Gaussian white-noise process with constant variance

$$E\{\vec{\eta}_1(t)\} = \vec{0} \quad (5-2)$$

$$E\{\vec{\eta}_1(t) \vec{\eta}_1^T(t')\} = Q_1 \delta(t-t') \\ = \begin{pmatrix} \sigma_{11}^2 & 0 & 0 \\ 0 & \sigma_{12}^2 & 0 \\ 0 & 0 & \sigma_{13}^2 \end{pmatrix} \delta(t-t') \quad (5-3)$$

where E denotes the expectation and T the matrix transpose. The drift-rate bias is driven by a second Gaussian white-noise process, the gyro drift-rate ramp noise, with constant variance

$$\frac{d}{dt} \vec{b} = \vec{\eta}_2 \quad (5-4)$$

with

$$E\{\vec{\eta}_2(t)\} = \vec{0} \quad (5-5)$$

$$E\{\vec{\eta}_2(t) \vec{\eta}_2^T(t')\} = Q_2 \delta(t-t') \quad (5-6)$$

$$= \begin{pmatrix} \sigma_{21}^2 & 0 & 0 \\ 0 & \sigma_{22}^2 & 0 \\ 0 & 0 & \sigma_{23}^2 \end{pmatrix} \delta(t-t')$$

The two noise processes are assumed to be uncorrelated

$$E\{\vec{\eta}_1(t) \vec{\eta}_2^T(t')\} = 0 \quad (5-7)$$

5.1.2 THE STATE EQUATION

The state vector of the system is given by

$$\underline{x} = \begin{bmatrix} \vec{q} \\ \vec{b} \end{bmatrix} \quad (5-8)$$

where \vec{q} is the attitude quaternion. The quaternion and the bias vector satisfy the coupled differential equations

$$\frac{d}{dt} \vec{q} = \frac{1}{2} \Omega(\vec{\omega}) \vec{q} = \frac{1}{2} \Omega(\vec{u} - \vec{b} - \vec{\eta}_1) \vec{q} \quad (5-9)$$

$$\frac{d}{dt} \vec{b} = \vec{\eta}_2 \quad (5-10)$$

with

$$\Omega(\vec{w}) = \begin{bmatrix} 0 & w_3 & -w_2 & w_1 \\ -w_3 & 0 & w_1 & w_2 \\ w_2 & -w_1 & 0 & w_3 \\ -w_1 & -w_2 & -w_3 & 0 \end{bmatrix} \quad (5-11)$$

Noting that the matrix function Ω is linear and homogeneous in its argument, and defining the 4×3 matrix function $S(\vec{q})$ by

$$\Omega(\vec{b}) \vec{q} = S(\vec{q}) \vec{b} \quad (5-12)$$

Equation (5-9) may be rewritten as

$$\frac{d}{dt} \vec{q} = \frac{1}{2} \Omega(\vec{u} - \vec{b}) \vec{q} - \frac{1}{2} S(\vec{q}) \vec{n}_1 \quad (5-13)$$

The matrix $S(\vec{q})$ has the explicit form

$$S(\vec{q}) = \begin{bmatrix} q_4 & -q_3 & q_2 \\ q_3 & q_4 & -q_1 \\ -q_2 & q_1 & q_4 \\ -q_1 & -q_2 & -q_3 \end{bmatrix} \quad (5-14)$$

5.2 STATE PROPAGATION

Taking the expectation of Equations (5-10) and (5-13) leads to equations satisfied by the predicted estimate of the state vector.

$$\frac{d}{dt} \hat{\vec{q}} = \frac{1}{2} \Omega(\vec{u} - \hat{\vec{b}}) \hat{\vec{q}} \quad (5-15)$$

$$\frac{d}{dt} \hat{\vec{b}} = \vec{0} \quad (5-16)$$

An approximation has been made such that

$$\Omega(\vec{u}-\hat{\vec{b}}) \approx \hat{\Omega}(\vec{u}-\vec{b}) \quad (5-17)$$

Integrating the equation shows the state estimate is propagated by

$$\hat{\vec{q}}(t) = e^{1/2} \int_{t_0}^t \Omega(\hat{\vec{w}}) dt \hat{\vec{q}}(t_0) \quad (5-18)$$

$$\hat{\vec{b}}(t) = \hat{\vec{b}}(t_0) \quad (5-19)$$

where $\hat{\vec{w}} = \vec{u} - \hat{\vec{b}}$. If the direction of $\hat{\vec{w}}$ is assumed constant over the interval (t, t_0) or if the rotation vector defined by

$$\Delta\vec{\theta} = \int_{t_0}^t \hat{\vec{w}}(t') dt' \quad (5-20)$$

is small, then using average value

$$\tilde{\vec{w}} = \frac{\int_{t_0}^t \hat{\vec{w}} dt'}{t - t_0} \quad (5-21)$$

we get

$$\hat{q}(t) = e^{1/2 \Omega(\tilde{w})(t-t_0)} \hat{q}(t_0) \quad (5-22)$$

or

$$\hat{q}(t) = \left(\cos \frac{w}{2} (t-t_0) I_{4 \times 4} + \frac{\sin \frac{w}{2} (t-t_0)}{w} \Omega(\tilde{w}) \right) \hat{q}(t_0) \quad (5-23)$$

where $w = |\tilde{w}|$ and $I_{4 \times 4}$ is the identity matrix of order 4x4.

Letting $t = t_n$, $t_0 = t_{n-1}$, and $t_n - t_{n-1} = \Delta t$ leads to

$$\hat{q}(t_n) = \left(\cos \frac{w}{2} \Delta t I_{4 \times 4} + \frac{\sin \frac{w}{2} \Delta t}{w} \Omega(\tilde{w}) \right) \hat{q}(t_{n-1}) \quad (5-24)$$

$$\hat{b}(t_n) = \hat{b}(t_{n-1}) \quad (5-25)$$

Taking first-order approximation in the trigonometric functions gives

$$\hat{q}(t_n) \cong \left(I_{4 \times 4} + \frac{\Delta t}{2} \Omega(\tilde{w}) \right) \hat{q}(t_{n-1}) \quad (5-26)$$

$$\hat{b}(t_n) = \hat{b}(t_{n-1}) \quad (5-27)$$

Quaternion should be renormalized after propagation using approximation Equation (5-26).

5.3 STATE-ERROR EQUATIONS

The state-error vector is defined by

$$\Delta \underline{x} = \underline{x} - \hat{\underline{x}} \quad (5-28)$$

or

$$\Delta \underline{q} = \underline{q} - \hat{\underline{q}} \quad (5-29)$$

and

$$\Delta \underline{b} = \underline{b} - \hat{\underline{b}} \quad (5-30)$$

Neglecting terms that are higher than first order in state-error vector and the process noise, the state-error vector satisfies the differential equations

$$\frac{d}{dt} \Delta \underline{q} = \frac{1}{2} \Omega(\hat{\underline{w}}) \Delta \underline{q} - \frac{1}{2} S(\hat{\underline{q}}) \Delta \underline{b} - \frac{1}{2} S(\hat{\underline{q}}) \underline{\eta}_1 \quad (5-31)$$

$$\frac{d}{dt} \Delta \underline{b} = \underline{\eta}_2 \quad (5-32)$$

The solution of these equations are given by

$$\begin{bmatrix} \Delta \underline{q}(t) \\ \Delta \underline{b}(t) \end{bmatrix} = \begin{bmatrix} \phi_{11}(t, t_0) & \phi_{12}(t, t_0) \\ \hline 0_{3 \times 4} & I_{3 \times 3} \end{bmatrix} \begin{bmatrix} \Delta \underline{q}(t_0) \\ \Delta \underline{b}(t_0) \end{bmatrix} + \begin{bmatrix} \underline{f}_1 \\ \underline{f}_2 \end{bmatrix} \quad (5-33)$$

where $\phi_{3 \times 4}$ is a null (zero) matrix of order 3×4 .

$$\begin{aligned} \phi_{11}(t, t_0) = e^{\frac{1}{2} \Omega(\tilde{w})(t-t_0)} &= \cos \frac{w}{2}(t-t_0) I_{4 \times 4} \\ &+ \frac{\sin \frac{w}{2}(t-t_0)}{w} \Omega(\tilde{w}) \end{aligned} \quad (5-34)$$

$$\phi_{12}(t, t_0) = -\frac{1}{2} \int_{t_0}^t \phi_{11}(t, t') S(\hat{q}(t')) dt' \quad (5-35)$$

$$\begin{aligned} f_1(t, t_0) = -\frac{1}{2} \int_{t_0}^t \phi_{11}(t, t') S(\hat{q}(t')) \\ \times [f_2(t', t_0) + \vec{\pi}_1(t')] dt' \end{aligned} \quad (5-36)$$

$$f_2(t, t_0) = \int_{t_0}^t \vec{\pi}_2(t') dt' \quad (5-37)$$

5.4 COVARIANCE PROPAGATION

The covariance matrix is defined by

$$P(t) = E\{\Delta \underline{X}(t) \Delta \underline{X}^T(t)\} \quad (5-38)$$

It follows from the state-error equations that the covariance matrix is propagated by

$$P(t_n) = \begin{pmatrix} \phi_{11} & \phi_{12} \\ \hline 0_{3 \times 4} & I_{3 \times 3} \end{pmatrix} P(t_{n-1}) \begin{pmatrix} \phi_{11} & \phi_{12} \\ \hline 0_{3 \times 4} & I_{3 \times 3} \end{pmatrix}^T + \begin{pmatrix} Q_{11} & Q_{12} \\ \hline Q_{21} & Q_{22} \end{pmatrix} \quad (5-39)$$

where

$$\phi_{11} = \phi_{11}(t_n, t_{n-1}) \quad (5-40)$$

$$\phi_{12} = \phi_{12}(t_n, t_{n-1}) \quad (5-41)$$

$$Q_{11} = E\{f_1 f_1^T\} = \frac{1}{4} \int_{t_{n-1}}^{t_n} \int_{t_{n-1}}^{t_n} \phi_{11}(t_n, t') S(\hat{q}(t')) \quad (5-42)$$

$$\times E\{(f_2(t', t_{n-1}) + \bar{\eta}_1(t'))(f_2(t'', t_{n-1}) + \bar{\eta}_1(t''))^T\} S^T(\hat{q}(t'')) \phi_{11}^T(t_n, t'') dt' dt''$$

$$Q_{12} = E\{f_1 f_2^T\} = -\frac{1}{2} \int_{t_{n-1}}^{t_n} \int_{t_{n-1}}^{t_n} \phi_{11}(t_n, t') S(\hat{q}(t')) \quad (5-43)$$

$$\times E\{(f_2(t', t_{n-1}) + \bar{\eta}_1(t')) \bar{\eta}_2^T(t'')\} dt' dt''$$

$$Q_{21} = E\{f_2 f_1^T\} = -\frac{1}{2} \int_{t_{n-1}}^{t_n} \int_{t_{n-1}}^{t_n} E\left\{\pi_2(t') \left(f_2(t'', t_{n-1}) + \pi_1(t'')^T\right)\right\} S^T(\hat{q}(t')) \phi_{11}^T(t_n, t'') dt' dt'' \quad (5-44)$$

$$Q_{22} = E\{f_2 f_2^T\} = \int_{t_{n-1}}^{t_n} \int_{t_{n-1}}^{t_n} E\left\{\pi_2(t') \pi_2^T(t'')\right\} dt' dt'' \quad (5-45)$$

The covariance matrix for the seven-dimensional state vector is singular. This follows from the constraint on the quaternion norm, that is

$$q^T q = 1 \quad (5-46)$$

The singularity is difficult to maintain numerically due to the accumulation of round-off error. In fact, $P(t_n)$ may even develop a negative eigenvalue. The simplest way to maintain the singularity is to represent $P(t_n)$ by a matrix of smaller dimension.

A 6x6 representation of the covariance matrix is derived below.

It can be shown that

$$\phi_{11}(t, t_0) S(\hat{q}(t_0)) = S(\hat{q}(t)) \tilde{\phi}_{11}(t, t_0) \quad (5-47)$$

and

ORIGINAL PAGE IS
OF POOR QUALITY

$$\begin{aligned} \phi_{11}(t, t_0) &= S(\hat{q}(t)) \tilde{\phi}_{11}(t, t_0) S^T(\hat{q}(t_0)) \\ &+ \hat{q}(t) \hat{q}^T(t_0) \end{aligned} \quad (5-48)$$

where

$$\tilde{\phi}_{11}(t, t_0) = e^{\int_{t_0}^t \tilde{\Omega}(\hat{w}) dt} \approx e^{\tilde{\Omega}(\hat{w})(t-t_0)} \quad (5-49)$$

is the 3x3 attitude rotation matrix and

$$\tilde{\Omega}(\hat{w}) = \begin{bmatrix} 0 & w_z & -w_y \\ -w_z & 0 & w_x \\ w_y & -w_x & 0 \end{bmatrix} \quad (5-50)$$

Substituting into Equations (5-41), (5-42), (5-43), and (5-44) we get

$$\phi_{12}(t_n, t_{n-1}) = S(\hat{q}(t_n)) \tilde{\phi}_{12}(t_n, t_{n-1}) \quad (5-51)$$

$$Q_{11} = S(\hat{q}(t_n)) \tilde{Q}_{11} S^T(\hat{q}(t_n)) \quad (5-52)$$

$$Q_{12} = S(\hat{q}(t_n)) \tilde{Q}_{12} \quad (5-53)$$

$$Q_{21} = \tilde{Q}_{21} s^T(\hat{q}(t_n)) \quad (5-54)$$

$$Q_{22} = \tilde{Q}_{22} \quad (5-55)$$

where

$$\tilde{\phi}_{12}(t_n, t_{n-1}) = -\frac{1}{2} \int_{t_{n-1}}^{t_n} \tilde{\phi}_{11}(t_n, t') dt' \quad (5-56)$$

$$\begin{aligned} \tilde{Q}_{11} = & \frac{1}{4} \int_{t_{n-1}}^{t_n} \int_{t_{n-1}}^{t_n} \tilde{\phi}_{11}(t_n, t') E\left\{ (f_2(t', t_{n-1}) \right. \\ & + \tilde{\eta}_1(t')) (f_2(t'', t_{n-1}) \\ & + \tilde{\eta}_1(t''))^T \} \tilde{\phi}_{11}^T(t_n, t'') dt' dt'' \end{aligned} \quad (5-57)$$

$$\begin{aligned} \tilde{Q}_{12} = & -\frac{1}{2} \int_{t_{n-1}}^{t_n} \int_{t_{n-1}}^{t_n} \tilde{\phi}_{11}(t_n, t') E\left\{ (f_2(t', t_{n-1}) \right. \\ & + \tilde{\eta}_1(t')) \tilde{\eta}_2^T(t'') \} dt' dt'' \end{aligned} \quad (5-58)$$

$$\begin{aligned} \tilde{Q}_{21} = & -\frac{1}{2} \int_{t_{n-1}}^{t_n} \int_{t_{n-1}}^{t_n} E\left\{ \eta_2(t') (f_2(t'', t_{n-1}) \right. \\ & + \tilde{\eta}_1(t''))^T \} \tilde{\phi}_{11}^T(t_n, t'') dt' dt'' \end{aligned} \quad (5-59)$$

$$\tilde{Q}_{22} = \int_{t_{n-1}}^{t_n} \int_{t_{n-1}}^{t_n} E \left\{ \tilde{\eta}_2(t') \tilde{\eta}_2^T(t'') \right\} dt' dt'' \quad (5-60)$$

Substituting Equations (5-47) and (5-48) into Equation (5-39), the covariance matrix propagation can be written

$$\begin{aligned} P(t_n) = & \mathcal{J}(\hat{q}(t_n)) \tilde{\Phi}(t_n, t_{n-1}) \mathcal{J}^T(\hat{q}(t_{n-1})) P(t_{n-1}) \mathcal{J}(\hat{q}(t_{n-1})) \\ & \times \tilde{\Phi}^T(t_n, t_{n-1}) \mathcal{J}^T(q(t_n)) + \begin{bmatrix} \hat{q}(t_n) \hat{q}^T(t_{n-1}) & 0_{4 \times 3} \\ \hline 0_{3 \times 4} & 0_{3 \times 3} \end{bmatrix} \\ & \times P(t_{n-1}) \begin{bmatrix} \hat{q}(t_n) \hat{q}(t_{n-1}) & 0_{4 \times 3} \\ \hline 0_{3 \times 4} & 0_{3 \times 3} \end{bmatrix}^T \\ & + \mathcal{J}(\hat{q}(t_n)) \tilde{Q} \mathcal{J}^T(\hat{q}(t_n)) \end{aligned} \quad (5-61)$$

where

$$\mathcal{J}(\hat{q}(t)) = \begin{bmatrix} s(\hat{q}(t)) & 0_{4 \times 3} \\ \hline 0_{3 \times 3} & I_{3 \times 3} \end{bmatrix} \quad (5-62)$$

$$\tilde{\Phi}(t_n, t_{n-1}) = \begin{bmatrix} \tilde{\Phi}_{11}(t_n, t_{n-1}) & \tilde{\Phi}_{12}(t_n, t_{n-1}) \\ \hline 0_{3 \times 3} & I_{3 \times 3} \end{bmatrix} \quad (5-63)$$

$$\tilde{Q} = \begin{bmatrix} \tilde{Q}_{11} & \tilde{Q}_{12} \\ \hline \tilde{Q}_{21} & \tilde{Q}_{22} \end{bmatrix} \quad (5-64)$$

The second term of the right member of Equation (5-65) vanishes since

$$\hat{\mathbf{q}} \hat{\mathbf{q}}^T \mathbf{P} = 0 \quad (5-65)$$

because of the normalization condition (Equation (5-46)) on $\hat{\mathbf{q}}$.

Thus, the 6x6 covariance matrix can be defined by

$$\tilde{\mathbf{P}}(t) = \mathcal{J}^T(\hat{\mathbf{q}}(t)) \mathbf{P}(t) \mathcal{J}(\hat{\mathbf{q}}(t)) \quad (5-66)$$

and it is propagated by

$$\tilde{\mathbf{P}}(t_n) = \tilde{\Phi}(t_n, t_{n-1}) \tilde{\mathbf{P}}(t_{n-1}) \tilde{\Phi}^T(t_n, t_{n-1}) + \tilde{\mathbf{Q}} \quad (5-67)$$

This covariance matrix corresponds to the state-error differential equations

$$\frac{d}{dt} \Delta \vec{\theta} = \tilde{\Omega}(\tilde{\mathbf{w}}) \Delta \vec{\theta} - \frac{1}{2} \Delta \vec{b} - \frac{1}{2} \vec{\eta}_1 \quad (5-68)$$

$$\frac{d}{dt} \Delta \vec{b} = \vec{\eta}_2 \quad (5-69)$$

The following gives further derivation of $\tilde{\Phi}$ and $\tilde{\mathbf{Q}}$ from Equations (5-49), (5-56), (5-57), (5-58), (5-59), and (5-60)

$$\begin{aligned} \tilde{\Phi}_{11}(t_n, t_{n-1}) &= e^{\tilde{\Omega}(\tilde{\mathbf{w}})(t_n - t_{n-1})} \\ &= \mathbf{I}_{3 \times 3} + \frac{\sin W \Delta t}{W} \tilde{\Omega}(\tilde{\mathbf{w}}) + \frac{1 - \cos W \Delta t}{W^2} \tilde{\Omega}^2(\tilde{\mathbf{w}}) \end{aligned} \quad (5-70)$$

To the second order in $W\Delta t$

$$\tilde{\Phi}_{11}(t_n, t_{n-1}) = I_{3 \times 3} + \Delta t \tilde{\Omega}(\tilde{W}) + \frac{\Delta t^2}{2} \tilde{\Omega}^2(\tilde{W}) \quad (5-71)$$

$$\begin{aligned} \tilde{\Phi}_{12}(t_n, t_{n-1}) &= -\frac{1}{2} \int_{t_{n-1}}^{t_n} e^{\tilde{\Omega}(\tilde{W})(t_n-t')} dt' \\ &= \frac{\tilde{\Omega}^{-1}(\tilde{W})}{2} \left[I_{3 \times 3} - e^{\tilde{\Omega}(\tilde{W})(t_n-t_{n-1})} \right] \\ &= -\frac{1}{2} \left[\frac{\sin W\Delta t}{W} I_{3 \times 3} + \frac{(1-\cos W\Delta t)}{W^2} \tilde{\Omega}(\tilde{W}) \right] \end{aligned} \quad (5-72)$$

To the second order in $W\Delta t$

$$\tilde{\Phi}_{12}(t_n, t_{n-1}) \approx -\frac{1}{2} \left[\Delta t I_{3 \times 3} + \frac{\Delta t^2}{2} \tilde{\Omega}(\tilde{W}) \right] \quad (5-73)$$

$$\begin{aligned}
 \tilde{\sigma}_{12} &= -\frac{1}{2} \int_{t_{n-1}}^{t_n} \int_{t_{n-1}}^{t_n} \int_{t_{n-1}}^{t'} e^{\tilde{\Omega}(\tilde{W})(t_n-t')} \\
 &\quad \times E\{n_2(t''') n_2^T(t'')\} dt''' dt'' dt' \\
 &= -\frac{Q_2}{2} \int_{t_{n-1}}^{t_n} \int_{t_{n-1}}^{t'} e^{\tilde{\Omega}(\tilde{W})(t_n-t')} dt''' dt' \\
 &= -\frac{Q_2}{2} \int_{t_{n-1}}^{t_n} (t'-t_{n-1}) e^{\tilde{\Omega}(\tilde{W})(t_n-t')} dt' \\
 &= -\frac{Q_2 \tilde{\Omega}^{-2}(\tilde{W})}{2} \left[e^{\tilde{\Omega}(\tilde{W}) \Delta t} - \tilde{\Omega}(\tilde{W}) \Delta t - I_{3 \times 3} \right] \\
 &= -\frac{Q_2 \tilde{\Omega}^{-2}(\tilde{W})}{2} \left[\frac{\sin W \Delta t}{W} \tilde{\Omega}(\tilde{W}) + \frac{(1 - \cos W \Delta t)}{W^2} \tilde{\Omega}^2(\tilde{W}) - \tilde{\Omega}(\tilde{W}) \Delta t \right] \\
 &= -\frac{Q_2}{2W^2} \left[(1 - \cos W \Delta t) I_{3 \times 3} - \left(\frac{\sin W \Delta t}{W} - \Delta t \right) \tilde{\Omega}(\tilde{W}) \right]
 \end{aligned} \tag{5-74}$$

Approximately

$$\tilde{\sigma}_{12} \approx -\frac{Q_2}{4} \left[\Delta t^2 I_{3 \times 3} + \frac{\Delta t^3}{3} \tilde{\Omega}(\tilde{W}) \right] \tag{5-75}$$

$$\begin{aligned}
 \tilde{\sigma}_{21} = \tilde{\sigma}_{12}^T &= -\frac{Q_2}{2W^2} \left[(1 - \cos W \Delta t) I_{3 \times 3} \right. \\
 &\quad \left. + \left(\frac{\sin W \Delta t}{W} - \Delta t \right) \tilde{\Omega}(\tilde{W}) \right]
 \end{aligned} \tag{5-76}$$

Approximately

$$\tilde{\alpha}_{21} \approx -\frac{Q_2}{4} \left[\Delta t^2 I_{3 \times 3} - \frac{\Delta t^3}{3} \tilde{\alpha}(\tilde{w}) \right] \quad (5-77)$$

$$\begin{aligned} \tilde{\alpha}_{11} &= \frac{1}{4} \int_{t_{n-1}}^{t_n} \int_{t_{n-1}}^{t_n} \tilde{\phi}_{11}(t_n, t') \left[E \left\{ f_2(t', t_{n-1}) \right. \right. \\ &\quad \times f_2^T(t'', t_{n-1}) \left. \right\} \\ &\quad + E \left\{ \tilde{\eta}_1(t') \tilde{\eta}_1^T(t'') \right\} \tilde{\phi}_{11}^T(t_n, t'') \, dt' dt'' \\ &= \frac{Q_2}{4} \int_{t_{n-1}}^{t_n} \int_{t_{n-1}}^{t'} \int_{t_2}^{t_n} \int_{t_{n-1}}^{t''} e^{\tilde{\alpha}(\tilde{w})(t_n - t')} \delta(t_1 - t_2) \\ &\quad \times e^{\tilde{\alpha}(\tilde{w})(t'' - t_n)} \, dt_1 dt'' dt_2 dt' + \frac{Q_1 \Delta t}{4} \\ &= \frac{Q_2 \tilde{\alpha}^{-1}(\tilde{w})}{4} \int_{t_{n-1}}^{t_n} \int_{t_{n-1}}^{t'} e^{\tilde{\alpha}(\tilde{w})(t_n - t')} \left[I_{3 \times 3} \right. \\ &\quad \left. - e^{\tilde{\alpha}(\tilde{w})(t_2 - t_n)} \right] dt_2 dt' + \frac{Q_1 \Delta t}{4} \\ &= \frac{Q_2}{4} \left\{ \tilde{\alpha}^{-3}(\tilde{w}) \left[e^{\tilde{\alpha}(\tilde{w}) \Delta t} - e^{-\tilde{\alpha}(\tilde{w}) \Delta t} \right] - 2 \Delta t \tilde{\alpha}^{-2}(\tilde{w}) \right\} + \frac{Q_1 \Delta t}{4} \\ &= -\frac{Q_2}{2w^2} \left[\frac{\sin w \Delta t}{w} - \Delta t \right] + \frac{Q_1 \Delta t}{4} \end{aligned} \quad (5-78)$$

Approximately

$$Q_{11} \approx \frac{\tilde{Q}_2 \Delta t^3}{12} + \frac{Q_1 \Delta t}{4} \quad (5-79)$$

$$\tilde{Q}_{22} = Q_2 \Delta t \quad (5-80)$$

5.5 UPDATE

The measurement vector at time t_k , \underline{z}_k is related to the state vector

$$\underline{x}_k = \begin{pmatrix} \bar{a}_k \\ \bar{b}_k \end{pmatrix} \quad (5-81)$$

by

$$\underline{z}_k = \underline{h}(\underline{x}_k) + \underline{\eta}_{3k} \quad (5-82)$$

where $\underline{\eta}_{3k}$, the measurement noise, is a discrete Gaussian white-noise process

$$E\{\underline{\eta}_{3k}\} = \underline{0} \quad (5-83)$$

$$E\{\underline{\eta}_{3k} \underline{\eta}_{3k'}\} = R \delta_{k,k'} \quad (5-84)$$

The minimum-variance estimate of $\hat{\underline{x}}_k$ immediately following the measurement is given by

$$\hat{\underline{x}}_k (+) = \hat{\underline{x}}_k (-) + K_k [\underline{z}_k - \underline{h}(\hat{\underline{x}}_k (-))] \quad (5-85)$$

where the Kalman gain matrix is given by

$$K_k = P_k(-) H_k^T \left[H_k P_k(-) H_k^T + R_k \right]^{-1} \quad (5-86)$$

and the measurement partial matrix is given by

$$H_k = \frac{\partial h(\underline{x})}{\partial \underline{x}} \bigg|_{\hat{\underline{x}}_k(-)} \quad (5-87)$$

the covariance matrix immediately following the measurement is given by

$$P_k(+) = (I_{7 \times 7} - K_k H_k) P_k(-) \quad (5-88)$$

Notice that $\hat{\underline{x}}_k(-)$ and $P_k(-)$ denote the predicted values of the state vector and state covariance matrix at time t_k , and $\hat{\underline{x}}_k(+)$ and $P_k(+)$ denote the same quantities immediately following a measurement at time t_k .

The update algorithm may also be mechanized using the 6x6 covariance matrix $\tilde{P}(t)$. Define the nx6 matrix \tilde{H}_k , the 6xn matrix \tilde{K}_k , and 6x1 vector $\Delta \tilde{\underline{x}}_k$

$$\Delta \tilde{\underline{x}}_k = \begin{pmatrix} \Delta \vec{\theta}_k \\ \Delta \vec{b}_k \end{pmatrix} \quad (5-89)$$

$$\tilde{H}_k = H_k \mathcal{J}(\hat{\underline{q}}_k(-)) \quad (5-90)$$

$$\tilde{K}_k = \tilde{P}_k(-) \tilde{H}_k^T \left[\tilde{H}_k \tilde{P}_k(-) \tilde{H}_k^T + R_k \right]^{-1} \quad (5-91)$$

$$\hat{\underline{x}}_k(+) = \tilde{K}_k [\underline{z}_k - h(\hat{\underline{x}}_k(-))] \quad (5-92)$$

Then it can be shown that

$$\tilde{P}_k(+) = (I_{6 \times 6} - \tilde{K}_k \tilde{H}_k) \tilde{P}_k(-) \quad (5-93)$$

$$\tilde{K}_k = \mathcal{L}(\hat{q}_k(-)) \tilde{K}_k \quad (5-94)$$

$$\Delta \hat{x}_k \equiv \begin{pmatrix} \Delta \hat{q}_k \\ \Delta \hat{b}_k \end{pmatrix} \equiv \hat{x}_k(+) - \hat{x}_k(-) = \mathcal{L}(\hat{q}_k(-)) \Delta \hat{x}_k \quad (5-95)$$

n is the dimension of the measurement vector. The quaternion part of the state vector should be renormalized after the update, that is

$$\hat{q}_k = \frac{\hat{q}_k(+)}{|\hat{q}_k(+)|} = \frac{\hat{q}_k(-) + \Delta \hat{q}_k}{|\hat{q}_k(-) + \Delta \hat{q}_k|} \quad (5-96)$$

For each star observed by the star tracker, we define two model measurements

$$h = \begin{pmatrix} h_1 \\ h_2 \end{pmatrix} = \begin{pmatrix} X_s/Z_s \\ Y_s/Z_s \end{pmatrix} \quad (5-97)$$

where X_s , Y_s , and Z_s are components of the star unit vector represented in star tracker frame.

$$\vec{V}_s = \begin{pmatrix} X_s \\ Y_s \\ Z_s \end{pmatrix} \quad (5-98)$$

\vec{V}_S is related to the star unit vector in spacecraft body frame \vec{V}_B and inertial frame \vec{V}_I by

$$\vec{V}_S = M \vec{V}_B = M A(\vec{q}) \vec{V}_I \quad (5-99)$$

where M is the spacecraft body to star tracker rotation matrix (misalignment of the star tracker included in this rotation) and A(q) is the attitude matrix given by

$$A(q) = \begin{bmatrix} q_1^2 - q_2^2 - q_3^2 + q_4^2 & 2(q_1q_2 + q_3q_4) & 2(q_1q_3 - q_2q_4) \\ 2(q_1q_2 - q_3q_4) & -q_1^2 + q_2^2 - q_3^2 + q_4^2 & 2(q_2q_3 + q_1q_4) \\ 2(q_1q_3 + q_2q_4) & 2(q_2q_3 - q_1q_4) & -q_1^2 - q_2^2 + q_3^2 + q_4^2 \end{bmatrix} \quad (5-100)$$

The measurement partial matrix is given by

$$H = \frac{\partial \underline{h}}{\partial \underline{x}} = \left[\frac{\partial \underline{h}}{\partial \vec{q}} \mid \frac{\partial \underline{h}}{\partial \vec{b}} \right] = \left[\frac{\partial \underline{h}}{\partial \vec{q}} \mid 0_{2 \times 3} \right] \quad (5-101)$$

$$\frac{\partial \underline{h}}{\partial \vec{q}} = \frac{\partial \underline{h}}{\partial \vec{V}_S} \frac{\partial \vec{V}_S}{\partial \vec{q}} \quad (5-102)$$

$$\frac{\partial \underline{h}}{\partial \vec{V}_S} = \begin{bmatrix} 1/z_s & 0 & -x_s/z_s^2 \\ 0 & 1/z_s & -y_s/z_s^2 \end{bmatrix} \quad (5-103)$$

$$\frac{\partial \vec{V}_S}{\partial \vec{q}} = M \frac{\partial A(\vec{q}) \vec{V}_I}{\partial \vec{q}} \quad (5-104)$$

It can be shown that

$$\left(\frac{\partial A(\vec{q}) \vec{V}_I}{\partial \vec{q}} \right) S(\vec{q}) = -2\tilde{\Omega}(\vec{V}_B) \quad (5-105)$$

Thus, we have the reduced measurement partial matrix given by

$$\tilde{H} = \left[\begin{array}{c|c} -2 \frac{\partial h}{\partial \vec{V}_s} M \tilde{\Omega}(\vec{V}_B) & 0_{2 \times 3} \end{array} \right] \quad (5-106)$$

In general, \underline{z} , \underline{h} , and \underline{n}_3 will be $2m$ -component vectors, \tilde{H} $2m \times 6$ matrix, and \tilde{K} $6 \times 2m$ matrix, where m is the number of stars observed at a certain time..

The matrix Equations (5-93), (5-94), and (5-95) can be decomposed and manipulated to give rise to a set of recursive formulae. Components of the measurement vector can be processed through the recursive formula one at a time. The final result should be the same as that obtained using full matrix equations. The recursive formulae are given by

$$\Delta \hat{\underline{X}}_n = \Delta \hat{\underline{X}}_{n-1} + \tilde{K}_n \left[\underline{z}_n - h_n(\hat{\underline{X}}_0) - \tilde{H}_n(\hat{\underline{X}}_0) \Delta \hat{\underline{X}}_{n-1} \right] \quad (5-107)$$

$$\tilde{K}_n = \tilde{P}_{n-1} \tilde{H}_n^T(\hat{\underline{X}}_0) \left[R_n + \tilde{H}_n(\hat{\underline{X}}_0) \tilde{P}_{n-1} \tilde{H}_n^T(\hat{\underline{X}}_0) \right]^{-1} \quad (5-108)$$

$$\tilde{P}_n = [I_{6 \times 6} - \tilde{K}_n \tilde{H}_n(\hat{\underline{X}}_0)] \tilde{P}_{n-1} \quad (5-109)$$

\underline{z}_n , h_n , and \tilde{H}_n (1×6 matrix) are n th row of \underline{z} , \underline{h} , and \tilde{H} , respectively. $\hat{\underline{X}}_0$ is the initial estimate of the state vector before any component of the measurement vector is processed. $\Delta \hat{\underline{X}}_n$ and $\Delta \hat{\underline{X}}_{n-1}$ are the corrections to the 6-component state vector after processing n components and $(n-1)$ components of the measurement vector, respectively. \tilde{P}_n and \tilde{P}_{n-1} are the corresponding 6×6 state covariance matrices. \tilde{K}_n is the 6×1 Kalman gain matrix after processing n components of the measurement vector.

APPENDIX A - TIMING AND MEMORY ESTIMATES

These estimates were made during the previous design process (Task 92300). These estimates were given in the document entitled, Microprocessor-Based Autonomous Attitude Determination System Design (CSC/TM-81/6085) (Reference 3) and are reproduced here for convenience.

A.1 TIMING AND MEMORY STUDIES

The ability of the design to meet computational requirements was shown earlier. The ability to meet the execution time and memory requirements is described in this section.

Implementation of AADS on the target microprocessor, the Intel 8086/8087 microcomputer, is constrained by the amount of Intel computer memory available to AADS, the timing requirements for AADS execution, and the accuracy requirements for attitude determination. This section describes the analysis performed to demonstrate the feasibility of implementing AADS on the Intel computer.

Estimates of AADS memory and timing requirements were determined from published data on memory and timing needs of similar software systems designed for the NASA Standard Systems Computer (NSSC-1) and the manufacturer's specifications for the Intel hardware capabilities. A study was conducted to identify the AADS parameters that are critically sensitive to computer truncation errors and must be represented in double-precision format to prevent degradation of attitude determination accuracy. Results of the study indicated that the number of these parameters did not significantly alter the AADS memory and timing estimates.

Sections A.1.1 and A.1.2 document the AADS computer memory and timing studies, respectively. Section A.1.3 presents the analysis and results of the floating point precision study.

A.1.1 AADS MEMORY REQUIREMENTS STUDY

In this section, conservative estimates are developed for the memory requirements of AADS. It is shown that AADS can reside on the Intel 8086/8087 without violating the 250K-byte total memory address limitations of the Intel.

The AADS memory estimates are based on memory sizings performed for similar software implemented on the NSSC-1 computer. The increased memory needed to accommodate the large AADS star catalog and data output buffer was taken into account. The AADS design specifications provide for a star catalog that contains all the SKYMAP catalog stars down to a stellar magnitude of 6.4, approximately 8500 stars, and a data output buffer that stores samples of raw and reduced data as well as spacecraft attitude and critical performance parameters. To extrapolate the memory requirements evaluated for an onboard attitude determination system installed on the NSSC-1 computer to the Intel microprocessor, each NSSC-1 18-bit word was associated conservatively with two Intel 16-bit words. The estimated memory requirements of the AADS subsystem are presented in Table A-1.

AADS memory requirements are compatible with the Intel computer operating environment. Although the Intel can handle up to 1 megabyte of memory, a design requirement limits the size of AADS to 256K bytes. The Intel operating system and mathematical package are assumed to use 16K bytes, leaving 240K bytes for AADS code and data storage. All data and code must be stored internally in memory, since the Intel operating environment will not include peripherals. From Table A-1, the memory estimates for each process show that AADS will need a maximum memory size of 175K bytes. This estimate is well within the 240K bytes of memory allocated to AADS for the Intel.

Table A-1. Memory Requirements on the Intel 8086/8087

Process	Memory Required (K Bytes)		
	Program	Data	Total
Input Data Processing	1	1	2
Gyro Data Processing	2	2	4
Gyro Propagation	3	1	4
Star Data Processing and Star Identification	8	142 ^a	150
Attitude Update	3	1	4
Output Data Processing	<u>1</u>	<u>10</u>	<u>11</u>
	18	157	175

^aLarge catalog size because some redundant information is stored to save execution time.

A.1.2 AADS TIMING REQUIREMENTS STUDY

The goal of the timing study was to estimate the execution times of AADS processes and to perform a throughput analysis demonstrating that the Intel 8086/8087 microcomputer can execute a worst-case AADS operational scenario within the specified processing cycle time. AADS process execution times were estimated from the reference times provided by studies of the execution of an onboard attitude determination system on the NSSC-1 computer.

To compare the execution times of the same process on the Intel and the NSSC-1 computers, the relative speeds of the two computers were estimated. The first step in this analysis was to determine the instruction timings of the two computers. The instruction timings are presented in Table A-2. The cycle times of the NSSC-1 and the Intel 8086/8087 computer are 1 microsecond (μs) and 0.5 μs , respectively. The Intel performs a LOAD or STORE operation in 4 μs , and the NSSC-1 requires 9.5 μs to execute the corresponding operations. The faster performance of the Intel is because of its cache memory, which includes six prefetch registers. Analysis of the COMPARE operation indicates that it should require at least twice the time of the respective LOAD operations, namely 19 μs for the NSSC-1 and 8 μs for the Intel.

The Intel floating point instruction times were obtained from the Intel 8086 family user's manual. The corresponding NSSC-1 instruction times represent the times estimated for the NSSC-1 mathematics package to emulate the floating point instructions. Table A-2 also provides ratios of the execution times for each timing instruction on the Intel and NSSC-1 computers. One estimate shows that AADS code will consist of 70 percent load-and-compare (L&C) instructions, 22 percent single-precision (SP) arithmetic instructions,

Table A-2. Comparison of Instruction Timings on the NSSC-1 and Intel 8086/8087 Computers

<u>Instruction</u>	<u>NSSC-1 (μs)</u>	<u>Intel (μs)</u>	<u>Ratio of Intel to NSSC-1</u>
Load-and-Compare			
Load	9.5	4	0.42
Compare	19	8	0.42
SP Arithmetic ¹			
SP Add	18	17	0.94
SP Subtract	18	17	0.94
SP Multiply	53	19	0.36
SP Divide	85	39	0.46
DP Arithmetic ²			
DP Add	63	22	0.35
DP Subtract	83	22	0.26
DP Multiply	233	27	0.12
DP Divide	2500	50	0.02

¹SP = Single Precision

²DP = Double Precision

and 8 percent double-precision (DP) arithmetic instructions. The frequency of use of the basic types of arithmetic instructions in representative applications software is shown in Table A-3. Estimates for the distribution of the different types of instructions in AADS code were used in compute a weighted mean ratio of the instruction execution times of the Intel and NSSC-1 computers. From the estimates provided in Table A-3, weighted means were derived for the Intel/NSSC-1 ratios of single and double-precision arithmetic instruction times. These ratios, in conjunction with the distributions and timing ratios of the remaining instruction types, were then applied to computing the final weighted mean execution time ratio, using the formula:

$$\begin{aligned}
 \text{Intel/NSSC-1 Execution Time Ratio} &= 70\% \text{ L\&C} + 22\% \text{ SP} + 8\% \text{ DP} \\
 &= 0.7 (0.42) + 0.22 (0.85) \\
 &\quad + 0.08 (0.28) \\
 &= 0.5
 \end{aligned}$$

This result predicts a twofold improvement in the performance of the Intel microprocessor over the NSSC-1 computer. Approximate execution times on the Intel computer for AADS processes were calculated by multiplying the times determined on the NSSC-1 for executing comparable processes by the Intel/NSSC-1 timing ratio.

Table A-4 presents the processing times in milliseconds (ms) predicted for AADS processes. Because I/O times could not be translated directly from the NSSC-1 to the Intel, Table 1-5 does not include estimates for the AADS INPUT and OUTPUT processes. Very rough estimates for Intel I/O times were obtained by assuming that the time to input/output a 16-bit word between memory and the Intel 64K buffer and the time (0.5 μ s) to execute the buffer/transmission line interface. Based on an I/O execution rate of 3 μ s/word,

Table A-3. Frequency of Use of Arithmetic Instructions
in Applications Software Cycle

<u>Operator</u>	<u>Percentage</u>
Add	57
Subtract	26
Multiply	13
Divide	4

Table A-4. Timing Estimates for AADS Processes Implemented on the Intel 8086/8087

<u>Process</u>	<u>Processing Time (ms)</u>
Gyro Data Processing	20
Gyro Propagation	35
Star Data Processing	25
Star Identification	100
Attitude Update	<u>90</u>
Total	260 ms

the time to output 5000 words, the maximum number stored in the 10K output data buffer, would be 15 ms. Input processing will required less than 1 ms because of the relatively small amount of AADS input data.

APPENDIX B - QUESTIONS AND ANSWERS

This section contains questions raised by the design team. Each question is numbered to indicate the area of the design to which it pertains. The numbering scheme is as follows:

<u>Code</u>	<u>Area of Design</u>	<u>Page</u>
EX	Executive process	B-2
IO	INPUT, OUTPUT, SENSIN processes	B-25
AT	Attitude processes	B-40

Each question is dated and indicates the person who asked the question and the person who answered the question. When an answer to a question is available, it follows the question in the text. Q: indicates the beginning of a question; A: indicates the beginning of an answer.

The information obtained through these questions has been incorporated in the body of the document; however, the questions and answers are reproduced here to document the evolution of the AADS design. In some cases, the answers are incomplete or inaccurate because sufficient information was not available at the time. Brief notes are added after the answers to indicate this problem.

Sections 1 through 5 of this document should be consulted to resolve ambiguities or discrepancies.

Number EX001

Originator K. Saralkar

Answered by S. Sanders

Date 6/23/81

Date 7/3/81

Q: What sort of Software Clock will be used by AADS? Three possible clocks are described below.

1. The current time will be taken from the system clock. It will be in the form of year, month, day, and milliseconds of the day and will be very easy to implement for the prototype system. This will result in a single source of time for the OSS, SDS and AADS. The orbit program (AODS) uses this type of clock, but AODS is designed for the LSI-11. The operating system and the software clock features for the LSI-11 are similar to those for the PDP-11. AADS will have to change the logic based on the operating system clock if a microprocessor other than the LSI-11 is used.

2. A hardware counter with a full day capacity which is available to all onboard computers. GSS will uplink the start time. AADS will keep a counter for year, month, and day that will be set from the start time. The hardware counter will be set to the milliseconds from the start time. Any delay (offset time) in resetting the clock after the RESET command can be automatically added to the counter.

3. The hardware counter may cycle over a shorter time period; then AADS will keep cumulative time in its memory. The counter may generate an interrupt every N milliseconds; AADS will update the time register and reset the counter after an interrupt. AADS will have to compensate for any delay in or overhead required to perform these actions.
(See question IO003.)

Number EX001 (Cont'd)

A: The time will be obtained from the operating system in the form of year, month, day, and milliseconds of the day. Timers and system event flags based on timers will be available. This arrangement is based on the following proposed scenario.

The microprocessor will be provided with interrupts at regular time intervals, approximately every 10 milliseconds. These interrupts will be generated by the spacecraft clock for the actual onboard system. During simulation testing, these interrupts will be provided by a combination hardware and software configuration on the VAX-11 computer. The operating system in the microprocessor will use these interrupts to maintain the time. A subroutine may be needed for AADS to set the initial time in the operating system. This time would be obtained from the SET CLOCK command. The granularity of the time is not expected to have an adverse effect on AADS.

- NOTE:
1. AADS will use the system clock on the VAX. The SET CLOCK command will not be used. These requirements are TBD by GSFC on the Intel version (8/24/81).
 2. Test runs on the VAX-11/780 computer indicate that the 10-millisecond granularity and scheduling variations result in time tag errors for both gyro and star tracker data. These time tag errors produce an error of several arc seconds in the computed attitude (4/2/82).

Number EX002

Originator G. Klitsch

Answered by F. Snow and
K. Tasaki

Date 6/23/81

Date 7/3/81

Q: Will attitude results be output at regular intervals, at the request of the OSS, or both?

If requested by the OSS, will the request be for the most recent computed or propagated attitude, or will the request be for a previous attitude with a specified time?

Recommendation: If attitude results are requested by the OSS, the requests should be for the most recent attitude. This would eliminate the need to store previous attitudes in some wraparound queue, which would, in turn, reduce memory requirements and simplify logic. The need to flag previous attitudes as questionable when a RESET CLOCK command is processed or when a new state attitude is uplinked would also be eliminated. Moreover, it would be undesirable to have to interpolate between attitudes of enclosing times.

A: The most recent computed or propagated attitude will be output after every nth propagation, where n is an adjustable integer parameter. This output will start when propagation starts and will continue until gyro processing is suspended. Initially, n will be set to one.

Number EX003

Originator G. Klitsch

Answered by F. Snow

Date 6/23/81

Date 7/3/81

Q: Reports containing activity log entries, raw sensor data, global COMMON variables, and so on will be produced for downlink upon command. Will there be a requirement to output periodic reports over a specified time period? This would require commands specifying a start and end time and a STOP PERIODIC REPORTS command if a very large end time was specified.

Recommendation: Although this capability would increase the debugging capabilities in AADS, it would also complicate the scheduling logic and increase the size of the executive. This capability may be desirable during critical events, such as a maneuver. Tradeoffs in memory and time will have to be studied further before a recommendation can be made.

A: There is no requirement to output periodic reports over a specified time interval. The activity log will be downlinked by command or upon a critical error. Update performance parameters and the raw sensor data reports will also be downlinked upon a critical error. In addition, propagation and update performance parameters will be downlinked after every update, and the log of sample, raw sensor data will be downlinked on command.

The raw sensor data report will contain 14 entries with a period of approximately 0.5 minute. Thus, the log will contain samples covering approximately 7 minutes. The log will probably be wraparound, and the exact time between samples will probably be variable. See Table B-1.

Table B-1. Output Logs and Schedules

TYPE OF OUTPUT	FREQUENCY ¹				
	EVERY n^{th} PROPAGATION	EVERY T SECONDS	CRITICAL ERRORS	EVERY UPDATE	ON COMMAND
PROPAGATED ATTITUDE	•				
PROPAGATION AND UPDATE PERFORMANCE PARAMETERS			•	•	•
SAMPLE RAW SENSOR DATA LOG			•		•
ACTIVITY LOG			•		•

W234/81

¹THERE IS NO OUTPUT BY TIME.

¹There is no output by time.

NOTE: See Section 4.6 on message formats for additional information. Propagated attitude, activity log and sensor data reports are downlinked after n propagations where the factor n is separately selected for each report. The update performance report is downlinked after m state updates. In addition, the activity log and the raw sensor data report are downlinked in the event of an error (4/2/82).

Number EX004

Originator G. Klitsch

Answered by

Date 6/23/81

Date

Q: A wraparound activity log will be maintained to record the activities of AADS. A level-of-diagnostics parameter in a global COMMON area may be desirable to select the amount of information that should go into this activity log. This parameter could have the following values and meanings:

1. Everything. This includes routine scheduling, commands received and responses to them, and severe and minor errors.
2. Everything except routine scheduling.
3. Severe and minor errors only.
4. Severe errors only.

Recommendation: This parameter could be implemented easily and would provide greater flexibility in the recording of AADS activities.

NOTE: Activity log format is fixed and given in Section 4 (4/2/82).

Number EX005

Originator G. Klitsch

Answered by F. Snow

Date 7/6/81

Date 7/10/81

Q: Is AADS required to continue gyro data processing and propagation during a maneuver, or should AADS suspend processing and wait for a new state attitude uplink following the maneuver?

If AADS is required to continue processing, should the frequency of gyro data processing and/or propagation be increased, or should the error bounds on the gyro data be increased, or both? If the frequency of processing is increased, would this increase vary according to the maneuver, or would it remain constant? Will the same attitude accuracy requirements be in effect?

A: AADS is required to continue gyro data processing and propagation during a maneuver. The frequency will be increased, and the error bounds will remain unchanged. The gyro data processing and propagation frequency should be increased since no star tracking will be performed during a maneuver. The increased frequency should be the same for any maneuver. The attitude accuracy requirements also will remain the same. Star tracking will be resumed to obtain an attitude update following the maneuver. In addition, the gyro data processing and propagation frequency will be reset to the original value.

Number EX006

Originator G. Klitsch

Answered by F. Snow

Date 7/6/81

Date 7/10/81

Q: If star tracking is delayed because of a maneuver or the occultation of both star cameras, should the time limit for the next update be extended by an equivalent amount of time?

A: The time limit for an update should not be increased even if star tracking is delayed because of a maneuver or the occultation of both star cameras. A critical error message is downlinked if this time limit is exceeded. Ground support personnel should take into account any maneuver or the occultation of both star cameras in interpreting the error message.

Number EX007

Originator G. Klitsch

Answered by S. Sanders

Date 7/8/81

Date 7/17/81

Q: How accurate must the scheduling be? Must the scheduling be accurate to the centisecond or millisecond level? According to recent discussions, the proposed target computer will be getting timer interrupts approximately every 10 milliseconds, a condition making millisecond accuracy impossible. The choice of accuracy would also affect how the system time would be obtained on the VAX-11 computer. The \$GETTIM system directive returns the current system time in a 64-bit quadword. This system time is in 100-nanosecond units measured from the system base time. While this would achieve millisecond accuracy, manipulating a 64-bit quadword could be difficult. The subroutines IDATE and SECNDS return the date and seconds of the day. The seconds of the day are returned in a single-precision real variable accurate to .01 second. This would achieve centisecond accuracy.

The \$SETIMR system directive, used to set the system timer, requires input in the form of a 64-bit quadword. However, for variable time slicing, delta times can be specified instead of absolute times. This allows the quadword to be broken into two longwords with the first longword equal to -1 and the delta time specified in the second longword as a negative number of 100-nanosecond units. Conversion to these units would be simple and would allow for a delta time of over 7 minutes, much larger than any expected time slice.

Recommendation: Accuracy to centiseconds is recommended.

NOTE: The scheduling accuracy is at most 10 milliseconds. Procedures for increasing accuracy are TBD by GSFC (8/24/81).

A: The scheduling can be accurate to the centisecond level only. Although the \$SETIMR requires time input in 100-nanosecond units, it can measure time only in 10-millisecond units. Thus, accuracy to the millisecond level is impossible.

Number EX008

Originator G. Klitsch

Answered by F. Snow

Date 7/16/81

Date 7/17/81

Q: AADS will generate requests for occultation and ephemeris data, which are then produced by the OSS. Since maneuvers are planned by ground support personnel, the only maneuver schedules available to the OSS are those that the OSS receives from the ground. Should AADS generate requests for maneuver schedules, or should the lack of current maneuver schedules be construed to indicate that no future maneuvers have been planned?

A: AADS will not maintain a maneuver schedule unless the need for one is demonstrated at a later date. AADS will be able to detect a maneuver through a gyro gain change or a gyro rate change. When AADS detects a maneuver, the results of any current star camera processing will be purged, and the frequency of gyro and/or propagation processing will be increased. Following a maneuver, the frequency of processing will return to normal, and star camera processing will be restarted.

Number EX009

Originator G. Klitsch

Answered by F. Snow

Date 7/16/81

Date 7/18/81

Q: What is the expected time interval between ephemeris elements, and how much ephemeris data (how many elements) will be provided at a time?

A: A request for ephemeris data from AADS will include the timespan and the interval between ephemeris elements. This time interval between elements could be, for example, on the order of 1, 2, or 5 minutes. There will be an upper limit on the total number of ephemeris elements, probably 20. An ephemeris element will consist of the velocity vector and associated time, since this is all that is required by AADS. AADS will use a small interpolator to interpolate between elements as needed.

NOTE: AADS will request ephemeris at 1 minute intervals, and will use a six-point interpolator. Nominally, AADS will request 20 points at 1 minute intervals. The maximum number of points per request is 29 (4/2/82).

Number EX010

Originator G. Klitsch

Answered by F. Snow

Date 7/16/81

Date 7/17/81

Q: Will the occultation data provided to AADS take into account any planned maneuvers and the predicted attitude following a maneuver? If not, will updated occultation data be provided automatically following a maneuver, or should AADS generate a request? If AADS must generate a request, should it output the propagated attitude following the maneuver before generating the request?

A: A request for occultation data from AADS will include a requested timespan and the current propagated attitude. Following a maneuver, a request for occultation data will be made since any previous occultation data may be invalid.

NOTE: AADS will compute occultation information (8/24/81).

Number EX011

Originator G. Klitsch

Answered by S. Sanders

Date 7/16/81

Date 7/31/81

Q: On the Intel microprocessor, will the memory used by a process or by an executable image include the sizes of the global COMMON areas it accesses and/or the general subroutines (Math Pac) that it uses? This question is related to how executable images are created on the Intel and should be referred to Steve Sanders of Systex.

If the sizes of global COMMON areas and/or the general subroutines used by a process are included in the amount of memory used by a process, global COMMON areas and/or general subroutines could become a significant part of the 64 kilobyte limit per process. Furthermore, would the sizes of global COMMON areas and/or general subroutines be counted more than once in the 256-kilobyte overall limit?

A: Steve Sanders (from Systex) feels that the actual addressing limit for the Intel 8086 may be 128 kilobytes; the mathematical subroutines package can be shared by all the processes; and the global COMMON areas and the mathematical subroutine package will be counted in the address space allocated for each process. He also feels that there may be an address granularity of the order of 4K bytes. He will review the FORTRAN compiler when it becomes available to verify the aforementioned answers.

NOTE: Steve Sanders feels that 64 kilobytes each for instruction, data, and global COMMON is allowed (8/24/81).

Number EX012

Originator K. Saralkar

Answered by S. Sanders

Date 8/17/81

Date 8/21/81

Q: The high density RAM chips used for the Intel computer may exhibit random bit failures.

1. What is the rate of memory failure?
2. Is it possible to use error checking and correcting memories?
3. Can we use electrically erasable programmable memories (EPROMs) for certain critical portions of the program to guard against such memory failures?
4. Is there a monitor program that can perform check-sums to monitor the integrity of the memory?

A:

1. A 1-bit-per-month failure rate has been quoted for some 16-kilobyte RAM chips.
2. Error checking and/or correcting memories are too expensive and will not be used for the AADS target computer.
3. The EPROMS are currently available as 2 kilobytes per chip. This low density presents a packaging problem for the current hardware design.
4. No such monitor program is planned at present. The RAM failure rates are small enough to exclude the use of the monitor programs.

Number EX013

Originator K. Saralkar

Answered by K. Tasaki

Date 8/18/81

Date 8/21/81

Q:

1. Is there any requirement to uplink AADS or the star catalog during the mission? If so, define the mechanism for the uplink.

2. What is the limit for the dimmest star in the star catalog? Is this number variable during the mission?

A:

1. The star catalog will be loaded along with AADS before execution of the prototype program and will not be changed during execution. The mission requirement is TBD (but the star catalog will not be changed provided that the entire star catalog of the 8500 stars can be stored onboard).

2. 6.4 is the nominal limit for the dimmest stars. The actual limit is TBD by GSFC depending on the star catalog size considerations. The magnitude limits are not variable.

NOTE: The AADS star catalog contains 8666 stars in the magnitude range 2.0 to 6.5. The Landsat simulator uses stars with magnitudes down to 5.25 (4/2/82).

Number EX014

Originator K. Saralkar

Answered by K. Tasaki

Date 8/18/81

Date 8/21/81

Q: Can we use a software clock (with a scaled time) to speed up or slow down the testing?

AADS and the AADS simulator will use a synchronized software clock for this purpose. The following advantages can be seen.

1. Use of the clock will speed up real-time testing. For example, a 24-hour simulation can be performed in 1 or 2 hours.
2. High priority jobs or block times that could affect other users will not be required.
3. Conversely, high priority jobs by other users will not affect AADS system testing.
4. Costly manual supervision during testing will be reduced.
5. The program could be slowed down or stopped during a debug run to allow the use of the Dynamic Debugging Technique (DDT). A real-time clock will result in a loss of synchronization during stops for diagnostics.

A: AADS will get high priority on the VAX computer for testing purposes. GSFC does not want to use the scaled time since the AADS simulator will also have to be changed. However, this requirement is TBD by GSFC (8/24/81).

NOTE: Speeding up testing by compressing out dead times has more advantages (8/24/81).

Number EX015

Originator Y. Frenkel/K. Saralkar Answered by

Date 9/01/81

Date

Q: AADS code and data will be stored in RAM in the current flight system configuration. In this case, AADS should be supported during flight by RAM, checksum and benchmark test routines. These routines will be invoked by commands from the ground, or automatically for a predetermined condition such as a restart of AADS.

1. RAM test routine will check all RAM memory sequentially. Each location will be saved in a temporary area. The RAM test routine will (a) Set all bits (1), (b) reset all bits (0), (c) shift left 1 bit, and (d) shift right 1 bit. Finally, the original contents will be restored. The address of each location that produces an error will be transmitted to ground. (Note: find size/time for the test routine.)

2. Checksum routine will add all 1 bits in the entire/ROM memory. The count will be compared to the previously known count. An error message will be transmitted to the ground if the counts do not match.

3. A complete batch of data (gyro and star tracker data/ephemeris, etc.) will be stored in memory along with the previously calculated or known answers for attitude and identified stars.

a. Batch Benchmark Test: The benchmark data will be used in a batch run, i.e., all processes will run on completion of previously scheduled processes, and not wait for timer interrupts. A 7-minute major cycle will be completed in 2 minutes in this manner. The benchmark program will downlink the final results and compare the results with the known answers. An error will be generated if the answers do not match.

b. Real-Time Benchmark Test: Again, use the benchmark data, but the executive will schedule processes as is normally done.

Thus, AADS will check all computations and scheduling except the ephemeris request.

Number EX016

Originator K. Saralker

Answered by S. Sanders

Date 8/28/81

Date 8/28/81

- Q: 1. What is the software development procedure on the Intel?
2. How does the Intel LINK program work?
3. Can the AADS processes share the mathematical sub-routine library?

A: 1. The FORTRAN source is stored on disk on the VAX. A utility program (TBD) will transfer the source to the Intel development system. The source may be edited, compiled, and linked. Each process image is separately linked and produces a binary file. The binary file can be transmitted back to VAX for storage or can be stored on a floppy disk on the Intel.

- NOTE:
- a. Intel has fairly good editing facilities, but compilation is very slow.
 - b. The object image must be stored on the VAX for eventual transmission to AADS Intel computer.
 - c. The edited version of the source must be transmitted to VAX for storage (9/15/81).

Number EX017

Originator K. Saralker

Answered by K. Tasaki

Date 9/1/81

Date 10/1/81

Q: Is there any requirement to uplink/downlink AADS code for diagnostic testing purposes during flight? If so, how will the AADS code be transmitted?

A: There is no provision for uplinking/downlinking AADS code in the current design. However, the new Intel operating system by Systex should have the capability to store/dump AADS memory when commanded by the ground. The AADS code will have to be transmitted through OBC. GSFC will determine the total time needed to complete the uplink/downlink, which will be performed in the OBC spare time.

Number EX018

Originator C. Shenitz

Answered by K. Tasaki

Date 9/11/81

Date 11/15/81

Q: A software clock design is given below. Should AADS use this clock to save wall clock time during long simulation runs?

AADS can control the actual testing time by using scaled time or by compressing out dead time when the AADS is not performing any computations. The scaled time software clock uses a scale factor to speed up or slow down AADS time from real time. This procedure appears relatively complex, and will not be considered.

The second type of software clock eliminates the time during which AADS is idle; therefore, it can only speed up (not slow down) AADS time. However, this clock can be interrupted for online diagnostic testing without any loss of synchronization between the AADS subprocesses. Suppose AADS performs a state update once every 30 seconds, and requires only a fraction of a second for star data processing functions. Thus, AADS typically performs only 3 functions during the remaining time. It reads gyro data every 64 milliseconds, reads star tracker data every 100 milliseconds, and performs a state propagation every 512 milliseconds. The current timing estimates show that AADS can easily perform these three functions in 100 milliseconds or less leaving an idle time of nearly 400 milliseconds from the 512-millisecond propagation period. The software clock will eliminate this idle time by scheduling the three functions in succession. After each function is completed, the software clock "sets" the AADS clock ahead to the start time of the next scheduled function. Synchronization with the AADS simulator is maintained through a mailbox. If AADS is interrupted for online diagnostic testing, then the software

diagnostic testing, then the software clock is inactive and AADS time does not advance. Therefore, the synchronization between AADS and the AADS simulator is maintained.

Based on the above discussion, AADS time can be speeded up by a factor of 4 to 5 during simulation runs on the Intel 8086. The factor should be even larger for the VAX computer, because the VAX is appreciably faster compared to the Intel.

A: K. Tasaki has indicated that AADS will not use the software clock.

Number IO001

Originator C. Shenitz

Answered by K. Tasaki

Date 6/23/81

Date 6/26/81

Q: The execution log is required to contain a running record of errors encountered. Should we expand it to include

- a. Reception (acceptance) of various types of data (high-level ACK-NAK)?
- b. End of major processes (e.g., star tracking or update) for the past several minutes?

A: This question was submitted just before the release of a revision to the Requirements Definition for the AADS Simulator. This revision, dated June 25, 1981, redefines the activity log in such a way that items in category b in this question are now included in the log. Mr. Tasaki will consider part a of the question in making further revisions to the definition of the activity log.

- NOTE:
1. The ACK-NAK procedures are TBD by GSFC for the Intel version (8/24/81).
 2. See Section 4 for the contents of the activity log (4/2/82).

Number IO002

Originator C. Shenitz

Answered by K. Tasaki

Date 6/23/81

Date 6/26/81

Q: The ability to dump local (COMMON block) variables is not required. However, it will certainly be important for initial testing and system testing. We can

- a. output local variables directly from their particular process(s) during simulation (easy--"diagnostic output").

or

- b. arrange to pass local variables to the executive and then to the output queue by a "Send & Receive" mechanism (harder--but can be retained in later operational versions).

Which method is preferred?

A: Dumping sections of memory will be a capability available to the ground and provided by the operating system. (This raises question IO005.)

The capability of reporting (by the application program) local COMMON block contents will be further considered pending the presentation by Mr. C. Shenitz of the desirability of recovering such data.

NOTE: AADS does not have the capability of dumping sections of memory (4/2/82).

Number IO003

Originator C. Shenitz

Date 6/23/81

Answered by S. Sanders

Date 6/26/81

Q:

1. What is the expected stability of the AADS clock? If it is a software clock, is there overhead in reinitializing the clock after each timing interval (for example, re-issuing a Mark Time)?

2. What is the expected granularity of the clock when accessing it? For example, the SMM OBC issued interrupts every 15 milliseconds for executive rescheduling; however, the granularity of time stored in a particular memory location (the counter) was much coarser.

Possible "Clocks":

1. Programmable Interval Timer (PIT) (e.g., Intel 8253 chip) - suitable intervals available for Executive rescheduling interrupts and for fine granularity upon Read access.

2. Reissuing time interval commands (e.g., Mark Time). Overhead involved (tens of microseconds) each time an interrupt occurs. Periodic reset will be necessary to eliminate accumulated clock bias.

A: The clock for AADS will have a specific form when the AADS microprocessor will be attached to the VAX-11/780 computer. The AADS simulator software or, in particular, the OSS on the VAX will send a pulse (interrupt signal) to the AADS machine once every TBD (on the order of 10 milliseconds). The microprocessor operating system will respond by incrementing its running total of these clock ticks. This operation will be transparent to all application software.

The following capabilities regarding time should be provided to application programs by the operating system:

1. To set and reset the calendar date in year, month, day, and milliseconds of a day.
2. To provide the updated date upon demand by application programs.
3. To begin timing variable-length time intervals upon request and to end such events with an interrupt (e.g., the Mark Time macro on the DEC RSX-11M system).

- NOTE:
1. See Questions EX001, EX014, and EX018 for additional information (4/2/82)
 2. The SET CLOCK command has not been implemented in AADS (4/2/82).

Number IO004

Originator C. Shenitz

Answered by K. Tasaki

Date 6/25/81

Date 7/3/81

Q: In the following pages, the proposed format for messages for uplink and downlink is described. The blocking of records within a transmission is described first. This format is that used in AODS. Next, a record header format based on the one used in AODS is described. It is recommended that this proposed header format be used for AADS.

AADS TRANSMISSION FORMAT

One Transmission

Block 1 {
Record 1
Record n_1

Block 2 {
Record 1
Record n_2

Block N {
Record 1
Record n_N

Sentinel record = all 1s

Header for each record includes the following

Record number within the block

Record number within the transmission

Block number within the transmission

Uplink/Downlink Header Format for AADS

<u>Variable Name</u>	<u>Type</u>	<u>Dimension</u>	<u>Description</u>
SYNCl	Byte	1	Sync number (later, spacecraft ID)
INTYPE	Byte	1	Type of input: = 1, data = 2, command
INDATA	Byte	1	Type of data or command
NBLOCK	Byte	1	Running number of this record in block
IAREA	Byte	1	Data area number for in- sertion or report
KEY	Byte	1	Key for later reference (for handling a maneuver)
NTRAN	I*2	1	Running number of this record in transmission
IBLOCK	I*2	1	Running number of this block in transmission
NSIZE	I*2	1	Number of bytes used in record after header
TTRAN	R*8	1	Time of transmission (GMT)

1. KEY must be set to a negative number if it is not being used.
2. For inserting data into (uplink) or reporting data from (downlink) a specified data area, the following additional format will be adhered to:
 - a. The first 12 bytes will contain up to 48 contiguous indicators of 2 bits each
 - b. The indicators will describe the data starting at byte 33 of the record in order

- c. = 0, I*4 (data is present)
 = 1, R*4 (data is present)
 = 2, R*8 (data is present)
- d. Record contents will be loaded/unloaded by
 EQUIVALENCES

A: This format appears to be satisfactory.

- NOTE:
- 1. The 2-bit code for identifying variable type will not be used. The bytes representing parameter values will be stored in the tables by using a byte transfer routine (8/24/81).
 - 2. See Section 4 of this document for AADS message formats and contents (4/2/82).

Number IO005

Originator C. Shenitz

Answered by S. Sanders

Date 6/29/81

Date 7/3/81

Q: In view of the existence of operating system features discussed in the answers to questions IO002 and IO003, namely, the capability to dump memory areas and the accumulation of clock ticks driven by an external source, the following question arises:

Should there be a separate input port to the microprocessor that will be used for all commands and pulses to the operating system?

Recommendation: The separation of input ports should be done for the present system to facilitate the development of the system on the VAX-11. However, when upgrading to a microprocessor-based system, separate input ports may not be available.

A: There will be a separate input port to the microprocessor for operating system communications with the external world. This will provide for a separation of operating system and application communications.

NOTE: 1. Intel hardware configuration is TBD by GSFC (8/24/81).
2. AADS does not have the capability of dumping sections of memory (4/2/82).

Number IO006

Originator C. Shenitz

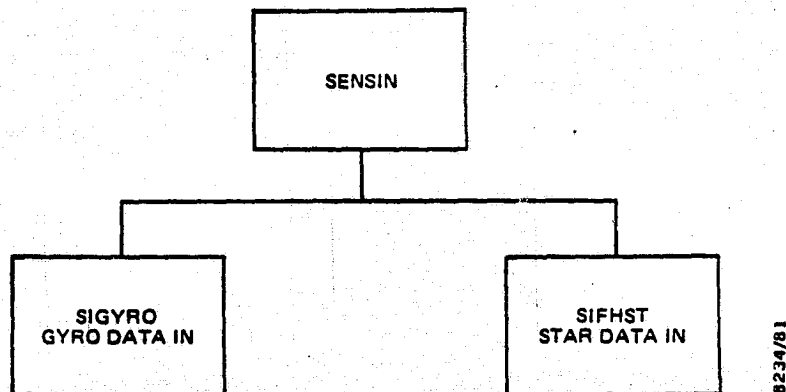
Answered by F. McGarry and
K. Tasaki

Date 7/13/81

Date 7/17/81

Q: At the Task 924 meeting at GSFC on July 10, 1981, Mr. F. McGarry raised the following point: several parts of the existing AADS design should be carefully reevaluated to justify the present relationships between components and to modify the design where necessary. This evaluation should be made with strict attention to AADS objectives and constraints.

In partial response to these remarks, several alternatives are presented for the basic design of the sensor data collection process. Design A is a part of the AADS design. It is recommended that the present Design A be kept for the advantages cited here. If the disadvantages make it difficult to implement AADS, then design B would be used with the necessary additions of control logic.



Design A

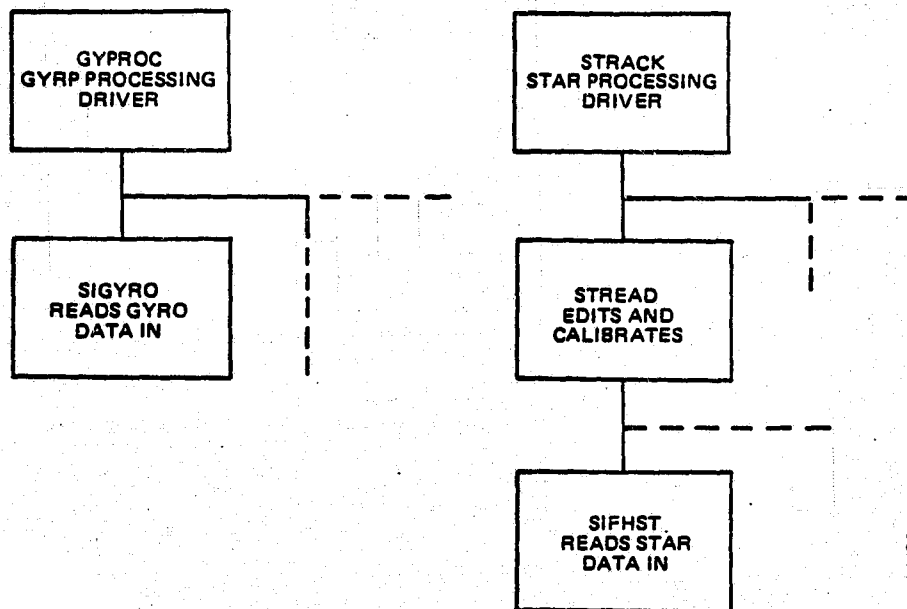
Advantages:

1. Allows for future (flight software) easy modification for interrupt-driven sensor data acquisition

2. Allows for logical independence and separate scheduling (at present) of sensor data retrieval and sensor data processing

Disadvantages:

1. Global COMMON (/SENSOR/ data base) needed for transferring sensor data to the processing processes
2. Overhead (space) needed for separate processes (FORTRAN library and system software routines repeated, until a sharable library is available)



Design B

Advantages:

1. Eliminates interprocess communication requirements (global COMMON)
2. Eliminates overhead (space) of a separate process (with no sharable library)

Disadvantages:

1. More logic and return codes needed for gyro and star data processing processes' interfaces with the executive; separately scheduled activities (acquisition and processing) are combined
2. Any future system requiring interrupt-driven data acquisition would require redesign

A: Design A is selected.

NOTE: See Section 1.5 on alternative AADS designs and justification for the current design (4/2/82).

Number IO007

Originator C. Shenitz

Answered by

Date 7/23/81

Date

Q: This multipart question concerns data communications mainly at the hardware level for the AADS microprocessor and the Onboard Support System (OSS).

1. Will there be "handshaking" between the two machine interfaces for I/O such that data transmitted from the OSS will not be lost if AADS delays the issuing of a read (QIO) request?

2. If the ACK and NAK characters are used for synchronizing transmissions, will the OSS initiate a timeout sequence after sending a (physical) record for waiting for an ACK or NAK? Will an error condition result if there is a timeout, or will the same record be automatically retransmitted until an ACK is received?

Assumptions being made are as follows:

1. Sufficient "handshaking" between the machines will exist for guarding against the loss of transmitted data

2. If a timeout period exists, it will result in automatic retransmission.

NOTE: 1. The exact handshaking (ACK/NAK) requirements are TBD by GSFC (8/24/81).
2. There are no handshaking procedures at the present time. GSS, OSS, and AADS will report errors in transmission, but will not ask for retransmission (4/2/82).

Number IO008

Originator C. Shenitz

Answered by K. Tasaki and
F. Snow

Date 7/23/81

Date 7/24/81

Q: Should the activity of updating any area of data pertaining to attitude computations be deferred until immediately after completing a major cycle (state update)? Should updating take place without regard to present processing, assuming that the resulting discontinuity in the attitude (or star tracking) history can be tolerated? Should updating these parameters be done at times when each would be appropriate (after a state propagation, after processing gyro data, after star processing, or after a state update)?

Recommendation: The last alternative is recommended.

A: AADS will queue uplinked commands that specify a change in data locations in designated areas. All these changes will be made after the completion of a major cycle (completion of a state update). If star tracking is not scheduled (either not initiated or suspended), the data location updates may be done after the completion of gyro data processing or state propagation. A maximum number (TBD) of such commands will be queued during any major cycle; if it is desired to uplink a greater number of data area updates, AADS processing must be halted. Otherwise, the records uplinked after the maximum number has been reached will be rejected.

NOTE: The number of parameters that can be updated during processing depends on the time taken to transfer a byte (8/24/81).

Number IO009

Originator C. Shenitz

Answered by K. Tasaki

Date 7/31/81

Date 8/7/81

Q: Should the data capture (DCAP) subsystem be placed in the same process as the INPUT subsystem, or should they be separate? The advantages of placing the subsystems in the same process image are

1. Reduced system code associated with a separate process
2. Elimination of global area(s) for queueing by DCAP of data area changes to be done by INPUT
3. Increased efficiency in reducing a filled queue (e.g., during system startup when data commands from the ground may be backed up in the OSS before their consecutive transmission to AADS); control can flow from DCAP to INPUT directly

The advantage of having separate processes for the above systems is that DCAP can be given highest priority, and INPUT can be assigned its proper lower priority.

Recommendation: Place the subsystems inside the same process image. If there are not sufficient task coordination and control facilities available in the Intel 8086 operating system, then separate the subsystems into separate processes.

A: DCAP and INPUT processes are merged.

Number I0010

Originator K. Saralker

Answered by S. Sanders

Date 8/28/81

Date 8/28/81

Q: Please describe the procedure for transmitting sensor data to AADS in the VAX-Intel system configuration.

A: See References 10 and 11 for general information available at this time.

Number AT001

Originator K. Liu

Answered by F. Snow

Date 7/31/81

Date 8/7/81

Q: Process personnel suggest that the occultation of the star tracker by the Sun, the Moon, or the Earth can be conveniently computed by AADS. This has the advantage of not having to request the occultation tables from the OBC. But the occultation computation needs the Moon ephemeris in addition to the Earth ephemeris from the OBC. (The Sun ephemeris will be computed internally by AADS.) Will this pose a problem for the OBC or the AADS simulator?

A: AADS will internally compute Sun, Moon, and Earth occultation. Sun and Moon ephemerides will be also computed by AADS.

Number AT002

Originator K. Liu

Answered by F. Snow

Date 7/31/81

Date 8/7/81

Q: Is correction of the star tracker reading due to the Earth's magnetic field a requirement for AADS?

A: Magnetic field correction is not required because it is expected to be 1 arc-second.

Number AT003

Originator K. Saralkar

Answered by F. Snow

Date 8/17/81

Date 8/21/81

Q: It is required to store all stars with magnitudes of between 3.0 and 6.4. There are approximately 8500 stars in this range in the SKYMAP catalog. The autonomous star identification document by P. Gambardella shows that there will be on average 28.6 stars in the FHST FOV. The following questions arise:

1. Is it required to get the star catalog from SKYMAP?
2. Can one set the threshold of the star camera to exclude stars beyond the magnitude of 6.2?
3. What will be the effect of using stars in the magnitude range of 3.0 to 6.2? There will be 1000 fewer stars in the star catalog. How will this affect star availability in certain less populated regions of the sky? This range reduction will mean a saving of approximately 16 kilobytes.

A:

1. The SKYMAP star catalog will be used to generate the onboard star catalog.
2. The star camera can be set at various thresholds commanded from the ground.
3. TBD by GSFC.

NOTE: The AADS star catalog contains 8666 stars in the magnitude range 2.0 to 6.5, and requires approximately 55 kilobytes (4/2/82).

Number AT004

Originator K. Saralkar

Answered by

Date 8/17/81

Date

Q: A preliminary version of the star catalog shows the stars arranged by right-ascension values. Each entry in the catalog consists of the right ascension, declination, magnitude, and the SKYMAP star number. Therefore, the catalog for all stars with magnitudes of between 3.0 and 6.4 will require approximately 130 kilobytes of memory for the 8500 stars.

1. Can we store the magnitude of a star as a 1-byte scaled integer? The star camera resolves stars with a magnitude difference of 0.25 units. Therefore, a one-byte field will be sufficient to represent the star magnitudes with required accuracy. This will save 3 bytes per star entry or approximately 24 kilobytes for the entire catalog.

2. AADS is required to output the SKYMAP catalog numbers for the identified stars. CSC will write a program (CATLOG) to create the onboard star catalog from the SKYMAP catalog for a given epoch. The onboard star catalog will be formatted as an array with each entry describing a star. The program CATLOG will also produce a table (TABLE) giving the array index in the onboard catalog versus the SKYMAP number for all stars. AADS will output the internal star reference number (array index) to the ground, and GSS can then find the actual SKYMAP number by means of a simple table lookup from TABLE. This procedure will eliminate 4 bytes per star that would be otherwise required to store the SKYMAP number in the onboard catalog. A memory saving of 4 bytes per star or approximately 34 kilobytes will be possible. Is this procedure acceptable?

NOTE: The AADS star catalog contains 8666 stars in the magnitude range 2.0 to 6.5, and requires approximately 55 kilobytes (4/2/82).

Number AT005

Originator K. Saralkar

Answered by

Date 8/17/81

Date

Q. An alternative star catalog structure is described here. An onboard star catalog will be divided into 360 subcatalogs. A given subcatalog will contain all stars with their right ascensions in a 1-degree range. For example, the 35th subcatalog will contain stars for which $34 \text{ degrees} \leq \text{right ascension of the star} < 35 \text{ degrees}$. The right ascension is divided into the integral and the fractional part. For a given star, the integral part of the right ascension will be the subcatalog index, and the fractional part will be stored as a 2-byte integer value.

For example, the entry 2389 in the 73rd subcatalog will represent a star with a right ascension of 72.2389 degrees (the 73rd catalog contains stars with a right ascension of 72 degrees or more). The declination will still be given by a 4-byte field. This procedure will save 2 bytes per star entry or nearly 17 kilobytes for the entire star catalog containing 8500 stars. However, it will take more time to decode the star catalog. In addition, the star catalog generation program will be more complicated because of the additional encoding involved. A 7-byte entry per star (2 bytes for right ascension, 4 bytes for declination, and 1 byte for magnitude) will allow the entire catalog (magnitude range of between 3.0 to 6.4) to be stored in approximately 60 kilobytes.

NOTE. The AADS star catalog uses the above design. It contains 8666 stars in the magnitude range 2.0 to 6.5 and requires approximately 55 kilobytes (4/2/82).

Number AT006

Originator K. Saralkar

Answered by F. Snow

Date 8/17/81

Date 8/21/81

Q: AADS will compute the occultation of the star cameras by the Sun, the Earth, and the Moon. Spacecraft-to-Earth ephemeris will be supplied by the OBC (OSS). Sun ephemeris data is computed by using the utility routine SUN1X, and the Moon ephemeris is computed by using the utility routine SMPOS. SUN1X is available in the library ATTIT.ATTMAIN.UTIL.PACK.FORT; SMPOS is also available in the same library.

Various constants needed to compute the occultation and the three tolerances (for Sun, Earth, and Moon, respectively) will be added to the list of data base variables. This procedure will eliminate the communication between the OBC and the AADS computer required for the occultation request. The occultation computation will be moved from the OBC to AADS. This is a simple calculation, and it is not expected to affect the AADS schedules. This is scheme A.

In an alternate scheme (scheme B), the OBC will compute the occultation information based on the Sun, Earth, and Moon ephemerides. A 1-byte flag indicating the occultation of both star cameras by the Sun, Earth, or Moon will be added to each ephemeris entry in the ephemeris sent by the OBC to AADS. This scheme has several advantages.

1. The OBC has to generate the spacecraft ephemeris regardless of which program computes the occultation indicator.
2. A separate occultation request is unnecessary since the occultation flag is sent along with each ephemeris element.
3. AADS scheduling is much easier when the occultation computation is eliminated.

However, the tolerances for occultation will be stored in the OBC's memory and will have to be changed (if needed) using a separate OBC uplink.

Which scheme should be used?

A: Use scheme A for the VAX version. As far as possible, AADS should not depend on other programs or computers.

Number AT007

Originator K. Saralkar

Answered by

Date 8/17/81

Date

Q: The current version of SMPOS will give Moon ephemeris with an accuracy of 0.25 degree (arc-length) until 1981. After this time, the accuracy will deteriorate. The maximum error for SUN1X is 0.012 degree arc-length for all times from 1971 to 1981. The epoch is 1900; therefore, the accuracy will remain close to 0.012 degree arc-length for times beyond 1981.

1. What is the accuracy of SMPOS beyond 1981?
2. Can we use SMPOS for Moon occultation?
3. Can we use SUN1X for Sun occultation?
4. Can we use a Sun ephemeris obtained from SUN1X for the aberration correction?

NOTE: Routines SMPOS and SUN1X can be updated by updating epoch and other constants used in the routines. J. Rowe (CSC) has done some analysis required to update these routines for the required time periods (8/24/81).

Number AT008

Originator K. Saralkar

Answered by

Date 8/18/81

Date

Q: AADS uses a pairwise star matching technique for identifying stars. If the initial attitude is uncertain by a large amount (2 degrees), each observed star may have several candidate stars in the star catalog. All candidates for all observed stars are carried forward for identification; then pairwise matching between candidate stars is performed to resolve ambiguities.

The following questions arise:

1. Is direct matching necessary in addition to the pairwise technique?
2. What is the upper limit in attitude uncertainty at which the pairwise matching will fail?
3. What should be done for stars that cannot be identified using pairwise matching? Is triplet matching required for this or other cases?

- NOTE:
1. The SMM ground program uses triplet matching (8/24/81).
 2. AADS uses a modified pairwise matching scheme when the attitude uncertainty exceeds user specified tolerance. AADS automatically reverts to direct matching when the attitude uncertainty satisfies the user specified tolerance. AADS does not use a triplet matching scheme (4/2/82).

Number AT009

Originator K. Saralkar

Answered by

Date 8/18/81

Date

Q: Some of the stars in the star catalog may have near neighbors that will make identification difficult (ambiguous). The observation corresponding to such a pair may not be used for a star update. These pair stars can be identified by flagging the entries corresponding to the two stars, for example, by using a negative magnitude value. Is this a required procedure?

NOTE: AADS uses a negative magnitude value to indicate near neighbors (4/2/82).

Number AT010

Originator K. Saralkar

Answered by F. Snow

Date 8/18/81

Date 8/24/81

Q: Gyro-propagated attitude is output by AADS for data annotation purposes at a nominal period of 512 milliseconds. It is possible that because of some delay the attitude update process will be running when gyro processing and propagation is scheduled. The alternatives are to complete the attitude update and delay the gyro propagation, or to suspend the star update, complete gyro propagation, and resume attitude update.

1. Which alternative should be used?
2. Gyro propagation may be delayed (probably a few milliseconds) when a high priority process (command input) causes the gyro propagation to be suspended. Thus, the gyro propagation computation and data annotation will be delayed. Is this acceptable?

A:

1. Suspend attitude update and complete gyro propagation since data annotation is important.
2. The delay is acceptable since the attached time for the computed attitude is not changed, even though the actual computation is delayed.

Number AT011

Originator K. Saralkar

Answered by E. Lefferts

Date 8/18/81

Date 9/18/81

Q• The nominal period of gyro propagation is 512 milliseconds. E. Lefferts had indicated that the 30-arc-second (3σ) attitude accuracy required may be maintained at a gyro propagation period of 1024 milliseconds. What is the expected attitude accuracy when the present Kalman filter is used at a period of 1024 milliseconds?

A• The accuracy is TBD by GSFC personnel.

Number AT012

Originator C. Shenitz

Answered by

Date 8/17/81

Date

Q: In the existing AADS design document (March 1981), the so-called stabilized (Joseph) form of covariance update is described for the Kalman filter algorithm to be implemented. This is done to satisfy the design criteria of stability with regard to roundoff errors (see Sections 4.9.1, 4.9.2, and Section 5).

However, the attached excerpts from the JPL technical memorandum by C. L. Thornton and G. J. Bierman make the following points:

1. When double-precision arithmetic is used, numerical errors due to roundoff are of no major significance (in an interplanetary trajectory filtering problem)

2. The stabilized (Joseph) form--in a computationally efficient implementation--is still subject to numerical roundoff instability when single-precision variables are used.

3. The conventional Joseph form specified in the design is computationally very expensive and not even used in the cited study.

4. The Bierman-Thornton U-D (UDU^T) algorithm is stable and computationally economical.

5. Implement the conventional covariance update algorithm instead of the Joseph form as shown

$$P_j(+) = (I - K_j H_j) P_j$$

which is straightforward and easy to code and test. If numerical instability (loss of positive definiteness of the covariance) appears to be present in simulation testing, then the UDU^T factorization can be implemented in the final version.

CSC recommends that this algorithm be studied further to determine whether it will satisfy AADS requirements.

NOTE: AADS uses the routine RECUR from the file ATTIT.ATTMAIN.UTIL.PACK.FORT on the IBM 360/95 computer. This routine is used to process all available stars to compute a single correction to the current estimate of the state (4/2/82).

Number AT013

Originator E. Lefferts

Answered by K. Saralkar

Date 11/1/81

Date 11/10/81

Q: Can we separate state and covariance propagation? In the current AADS design, state propagation and covariance matrix propagation is performed after n gyro data processing intervals. Can the AADS design be changed so that the covariance matrix propagation is performed every m state propagations where m is typically 100?

A: The covariance matrix can be propagated every m state propagations after several changes in the gyro data processing and propagation code. In addition, the state covariance matrix should be updated just before star identification regardless of the current value of m, the covariance matrix propagation factor. This out of cycle propagation is required to provide the best estimate of attitude uncertainty for selecting the star identification method.

The following variables will be added to the COMMON areas.

STATIC COMMON

MCOVAR	Covariance matrix propagation interval factor. Covariance matrix propagation is performed after MCOVAR state propagation intervals. Default value 100.
JCOVAR	Interval factor for synchronizing state and covariance matrix propagation during the time just before star identification. For N*JCOVAR gyro data processing intervals <u>before</u> star identification, the covariance matrix will be propagated whenever state propagation is scheduled. The default value should be 2 or 3 for JCOVAR to ensure a current covariance matrix, which is needed to calculate the attitude uncertainty.

GLOBAL COMMON

Flags and other counters needed so that the executive can do the extra scheduling.

PROPAG LOCAL COMMON

$\Sigma T1$	Sum of incremental times	{ variable used for state propagation
$\Sigma A1$	Sum of incremental angles other variables	
:		
$\Sigma T2$	Sum of incremental times	{ variables needed for separate covariance propagation
$\Sigma A2$	Sum of incremental angles other variables	
:		

Actual Code Changes

1. Change one or two executive routines to perform the additional covariance matrix propagation scheduling.
2. Add logic to several GYPROC and PROPAG routines to store additional variables for covariance matrix propagation. PROPAG will keep an internal counter and perform covariance matrix propagation after MCOVAR state propagation intervals. GYPROC will also reset all counters and other variables such as $\Sigma T2$, $\Sigma A2$, etc.

APPENDIX C - WEEKLY DESIGN REVIEW DOCUMENTATION

This section contains letters that document the weekly design reviews.

The information contained in these letters has been incorporated in the body of the document; however, the letters are reproduced here to document the evolution of AADS design. In some cases, the letters contain incomplete or inaccurate information because sufficient information was not available at the time the letters were written.

Sections 1 through 5 of this document should be consulted to resolve ambiguities or discrepancies.

<u>Review Date</u>	<u>Letter Date</u>	<u>Page Number</u>
June 1	June 5	C-3
June 9	June 22	C-5
June 26	July 14	C-7
July 10	July 17	C-9
July 17	July 21	C-12
July 31	Aug. 20	C-14
Aug. 7	Aug. 20	C-14
Aug. 14	Aug. 20	C-18
Aug. 21	Aug. 25	C-21
Aug. 28	Sept. 1	C-23
Sept. 1	Sept. 8	C-25
Sept. 1	Sept. 9	C-27
Sept. 4	Sept. 9	C-30
Sept. 18	Oct. 1	C-32
Sept. 25	Oct. 1	C-34
Oct. 2	Oct. 5	C-36
Oct. 7	Oct. 13	C-38
Oct. 14	Oct. 19	C-40

<u>Review Date</u>	<u>Letter Date</u>	<u>Page Number</u>
Oct. 21	Undated	C-42
Nov. 3	Nov. 11	C-45
Nov. 3	Nov. 12	C-48

COMPUTER SCIENCES CORPORATION

SYSTEM SCIENCES DIVISION

(301) 589-1545

8728 COLESVILLE ROAD • SILVER SPRING, MARYLAND 20910

June 5, 1981

National Aeronautics and Space Administration
Goddard Space Flight Center
Greenbelt, Maryland 20771

Attention: Mr. K. Tasaki
Code 581.2
Bldg. 23, Room E-225

and

Mr. F. Snow
Code 582.3
Bldg. 23, Room E-239

Subject: Contract NAS 5-24300
Task Assignment 92400
Autonomous Attitude Determination System

Dear Messrs. Tasaki and Snow:

This memorandum summarizes CSC's understanding of the decisions made at a meeting held on June 1, 1981, at GSFC to review and clarify the software requirements for the microprocessor-based autonomous attitude determination system (AADS). Present were F. McGarry and K. Tasaki of GSFC; and S. Cheuvront, V. Church, G. Page, and K. Saralkar of CSC.

F. McGarry reviewed the concept of autonomous attitude determination and the way it fits in with the 'packetization' (autonomous processing for attitude, orbit, time, and science data). K. Tasaki reviewed the AADS system design created during the previous task (Task 92300). The following decisions were made during discussions:

- Attitude computation requirements and analysis are mainly complete except that the attitude accuracy requirements have not been completely specified.
- The design for the attitude computation function is mainly complete and only needs a light review.
- The design for the input and output subsystems is not complete because the system requirements have not been completely specified.

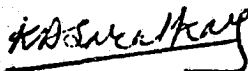
Mr. K. Tasaki
Mr. F. Snow

-2-

June 5, 1981

- A specification summary document is needed to consolidate all the facts needed to complete the AADS design.
- The current task assignment will be amended to require completion of detailed design for a skeleton system (shell) which will include the executive, input, and output subsystems. The requirement to code and test the shell will be deleted.
- Regular weekly meetings will be held at GSFC to monitor task progress.

Yours truly,



Dr. K. Saralkar
Task Leader
Attitude Systems Development

KS:gsp

copies: GSFC

F. McGarry
E. Lefferts

CSC

M. Plett
S. Cheuvront
G. Page
K. Liu
C. Shenitz
G. Klitsch

COMPUTER SCIENCES CORPORATION

SYSTEM SCIENCES DIVISION

(301) 589-1545

8728 COLESVILLE ROAD • SILVER SPRING, MARYLAND 20910

June 22, 1981

National Aeronautics and Space Administration
Goddard Space Flight Center
Greenbelt, Maryland 20771

Attention: Mr. K. Tasaki
Code 581.2
Bldg. 23, Room E-225

and

Mr. F. Snow
Code 582.3
Bldg. 23, Room E-239

Subject: Contract NAS-524300
Task Assignment 92400
Autonomous Attitude Determination System

Dear Messrs. Tasaki and Snow:

This memorandum summarizes CSC's understanding of the topics discussed at a meeting held on June 9, 1981, at GSFC to review the Autonomous Attitude Determination System (AADS) simulator and the attitude propagation and update equations. Present were E. Lefferts, F. Snow, K. Tasaki of GSFC; and K. Liu, G. Page, and K. Saralkar of CSC. Below is a summary of the information exchanged at the meeting.

1. The nominal period of gyro processing and attitude propagation is 0.512 seconds. It may be possible to compute attitude with the required accuracy using a larger period for gyro processing. Mr. Lefferts will determine the upper limit on the period.
2. The AADS simulator is made up of 3 functions:
 - Ground Support Subsystem (GSS)
 - Onboard Support Subsystem (OSS, also called OSFS in other documents)
 - Sensor Data Subsystem (SDS)
3. AADS will perform some validation of the commands and data input by the user.
4. OSS will generate ephemeris and occultation information as required. A subset or segment of this will be stored in the AADS memory as a table. AADS will read in a new segment when it reaches the last time in the current segment. The exact form of the occultation table is to be determined.

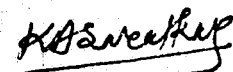
K. Tasaki
F. Snow

-2-

June 22, 1981

5. AADS will prepare any requested output as a header record followed by n data records. OSS will transmit (downlink) this block of records to the ground without further processing.
6. GSS, OSS, SDS, and AADS will need a consistent source of time which may be a software clock in each subsystem, plus a hardware clock. The form of this software clock and time synchronization mechanism is not completely specified.
7. SDS will store the gyro data in a buffer. The data will be updated every 64 milliseconds and will always be available to the AADS. The star tracker data, except for the temperature, will also be stored in a buffer and will be updated at smaller intervals.
8. The following questions must be answered before gyro and star tracker data can be time tagged properly.
 - What type of clock will be used by the system?
 - Since gyro data times will be in error by 64 milliseconds (maximum), how will this affect attitude propagation?
 - Is there a (constant or otherwise) time delay between the end of occultation and the start of star tracker output?
 - How will the AADS label the time of star temperature data in relation to the rest of star data? (The star temperature may be available asynchronously.)

Yours truly,



Dr. K. Saralkar
Task Leader
Attitude Systems Development

KS:gsp
copies:

GSFC

F. McGarry
E. Lefferts

CSC

M. Plett
S. Chevront
G. Page
K. Liu
C. Shenitz
G. Klitsch

COMPUTER SCIENCES CORPORATION

SYSTEM SCIENCES DIVISION

(301) 589-1545

8728 COLESVILLE ROAD • SILVER SPRING, MARYLAND 20910

July 14, 1981

National Aeronautics and Space Administration
Goddard Space Flight Center
Greenbelt, Maryland 20771

Attention: Mr. K. Tasaki
Code 581.2
Bldg. 23, Room E-225
and
Mr. F. Snow
Code 582.3
Bldg. 23, Room E-239

Subject: Contract NAS-524300
Task Assignment 92400
Autonomous Attitude Determination System (AADS)

Dear Messrs. Tasaki and Snow:

This memorandum summarizes CSC's understanding of the topics discussed at a meeting held on June 26, 1981, at GSFC to review task status. Present were F. Snow and K. Tasaki of GSFC; S. Sanders of Systex; and G. Klitsch, G. Page, K. Saralkar, and C. Shenitz of CSC. Following is a summary of information given by Systex and GSFC personnel in response to CSC questions:

- The Intel/8086 FORTRAN compiler will be available in the August to October, 1981 period.
- There is no suitable cross-compiler for the Intel/8086 on the VAX-11/780 computer. The current cross-compiler does not support double precision arithmetic and does not interface with the Intel/8087 numeric data processor chip.
- The Intel/8086 operating system will support time-of-day and timing interrupt functions which will be similar to the functions available on the VAX-11/780 computer.
- Gyro and star data (including star temperature data) will be available in digital form. AADS will access these data in a synchronous manner from a buffer area.

Mr. K. Tasaki
Mr. F. Snow

-2-

July 14, 1981

- A preliminary version of the AADS simulator will be available by September 30, 1981, and will be used for AADS testing purposes.

Yours truly,

K. Saralkar

Dr. K. Saralkar
Task Leader
Attitude Systems Development

KS:gsp
copies:

GSFC

F. McGarry
E. Lefferts

CSC

S. Cheuvront
G. Page
K. Liu
C. Shenitz
G. Klitsch

COMPUTER SCIENCES CORPORATION

SYSTEM SCIENCES DIVISION

(301) 589-1545

8728 COLESVILLE ROAD • SILVER SPRING, MARYLAND 20910

July 17, 1981

National Aeronautics and Space Administration
Goddard Space Flight Center
Greenbelt, Maryland 20771

Attention: Mr. K. Tasaki
Code 581.2
Bldg. 23, Room E-225
and
Mr. F. Snow
Code 582.3
Bldg. 23, Room E-239

Subject: Contract NAS-524300
Task Assignment 92400
Autonomous Attitude Determination System

Dear Messrs. Tasaki and Snow:

This memorandum summarizes CSC's understanding of the topics discussed at a meeting held on July 10, 1981, at GSFC to discuss schedules and task status. Present were F. McGarry and F. Snow of GSFC; and G. Klitsch, G. Page, C. Shenitz, and K. Saralkar of CSC.

CSC recommended that the AADS be implemented in three builds. The first build will include detailed design for the executive, input, and output tasks. This skeleton system which can interface with the existing simulator will be implemented on the VAX-11/780 computer. The Build 1 schedule is as follows:

Recommended Build 1 Schedule

Design	8/14/81
Review	9/1/81
Implementation	9/30/81

Additional capabilities such as input command and data validation, detailed output, and extensive error handling will be implemented in Build 2. The detailed design of the attitude computation tasks will also be completed in Build 2 which will end on October 30, 1981. The attitude system will be implemented in the Build 3 which will end on November 30, 1981.

July 17, 1981

During the ensuing discussion the following points were made:

- The handwritten specifications summary will be available for the design reviews.
- CSC will periodically submit the system design to the ATR for review and comments. In addition, informal design reviews will be held at GSFC before the Critical Design Review (CDR).
- CSC will present at the CDR the functional specifications for AADS and the ability of the design to meet those specifications. The emphasis will be on the main functional flow through the system rather than the detailed computations.
- Viewgraphs will not be required for the CDR.

F. McGarry raised several questions about the current design with respect to memory and time constraints, implementation techniques, and design flexibility to meet the changing requirements. CSC personnel justified the current design as follows:

- The breakdown of AADS into an executive, input, computational, and output tasks is based on functionality. The further breakdown of the computational task into five tasks (gyro processing, gyro propagation, star data, star identification, attitude update) is to generalize the system. Because these five functions may be performed at different and variable frequencies, a design decision was made to keep the scheduling logic at the executive level.
- The subtasks were designed to meet the 64K byte memory constraint and to meet the design goals of functional integrity (one function - one task).
- The earlier design (completed under Task 92300) showed that the AADS will meet timing constraints. It showed that nearly 250 milliseconds of the allotted 512 milliseconds will be available each cycle through AADS. In addition, E. Lefferts has indicated that the AADS may be able to achieve the 30 arc-second attitude accuracy requirement at a gyro-processing period of 1024 milliseconds. Further, actual CPU times will depend on the FORTRAN compiler used for the Intel/8086 microprocessor. Thus memory constraints and design considerations were given more weight at this stage of the design.
- The current design is flexible due to the logical and functional breakdown and should allow incorporation of most new or changed requirements without a major redesign.
- Memory and time optimization, if required, should be attempted after the system is developed, particularly for a prototype system.

July 17, 1981

- CSC will reevaluate the design of the computational subtasks during Build 2. The design will be modified if required in light of additional knowledge and experience gained.

Yours truly,



Dr. K. Saralkar
Task Leader
Attitude System Development

KS:gsp

copies: GSFC

F. McGarry
E. Lefferts

CSC

S. Cheuvront
G. Page
K. Liu
C. Shenitz
G. Klitsch

COMPUTER SCIENCES CORPORATION

SYSTEM SCIENCES DIVISION

(301) 589-1545

8728 COLESVILLE ROAD • SILVER SPRING, MARYLAND 20910

July 21, 1981

National Aeronautics and Space Administration
Goddard Space Flight Center
Greenbelt, Maryland 20771

Attention: Mr. K. Tasaki
Code 581.2
Bldg. 23, Room E-225
and

Mr. F. Snow
Code 582.3
Bldg. 23, Room E-239

Subject: Contract NAS-524300
Task Assignment 92400
Autonomous Attitude Determination System (AADS)

Dear Messrs. Tasaki and Snow:

This memorandum summarizes CSC's understanding of the topics discussed at a meeting held on July 17, 1981, at GSFC to review task status. Present were F. McGarry, F. Snow, and K. Tasaki of GSFC; S. Sanders of Systex; and G. Klitsch, G. Page, K. Saralkar, and C. Shenitz of CSC.

CSC personnel reviewed the original AADS design and presented an alternate design. The alternate design combines several tasks into single tasks in an attempt to reduce the overhead associated with the large number of tasks present in the original design. During the discussion that followed, S. Sanders indicated that CSC's questions about upper and lower limits on task size, whether the mathematical subroutine package is shared by all tasks or duplicated, and whether or not global COMMON areas are included in the address space of each task can be resolved only after the Intel/8086 firmware becomes available. After comparing the two designs, the attendees agreed to use the original design to develop the prototype system on the VAX-11/780. The design will be reevaluated later in light of the experience gained with the prototype system and information on the Intel/8086 firmware.

Mr. K. Tasaki
Mr. F. Snow

-2-

July 21, 1981

CSC and GSFC personnel made the following design decisions in response to CSC questions EX007 through EX010.

- The maneuver schedule will not be uplinked. AADS will sense a maneuver by checking for a higher gyro rate.
- AADS will suspend star data processing during a maneuver. Star data collected before the beginning of the maneuver will be discarded.
- AADS will request a new occultation table from the on-board computer after a maneuver. The request will contain the current attitude and the time span for the occultation table. Star data processing will be resumed after checking the new occultation table.
- AADS scheduling will be accurate to 10 milliseconds because of system clock limitations.

Yours truly,



Dr. K. Saralkar
Task Leader
Attitude Systems Development

KS:gsp

copies: GSFC

F. McGarry
E. Lefferts

CSC

S. Cheuvront
G. Page
K. Liu
C. Shenitz
G. Klitsch

ORIGINAL PAGE IN
OF POOR QUALITY

COMPUTER SCIENCES CORPORATION

SYSTEM SCIENCES DIVISION

(301) 589-1545

3728 COLESVILLE ROAD • SILVER SPRING, MARYLAND 20910

August 20, 1981

National Aeronautics and Space Administration
Goddard Space Flight Center
Greenbelt, Maryland 20771

Attention: Mr. K. Tasaki
Code 581.2
Bldg. 23, Room E-225
and

Mr. F. Snow
Code 582.3
Bldg. 23, Room E-239

Subject: Contract NAS-524300
Task Assignment 92400
Autonomous Attitude Determination System (AADS)

Dear Messrs. Tasaki and Snow:

This memorandum summarizes CSC's understanding of the topics discussed at a meeting held on July 31, 1981, at GSFC to review task status. Present were E. Lefferts, F. Snow, and K. Tasaki of GSFC; and G. Klitsch, K. Liu, K. Saralkar, and C. Shenitz of CSC.

G. Klitsch reviewed the top-level design for the executive task. Ground commands to AADS and executive action for these commands was discussed. The STOP command will cause the AADS to complete the update process, downlink the last output, and stop gyro and star processing. The AADS will wait for the next command. The HALT command suspends star processing; gyro processing and propagation activities continue. The operating system will maintain all time-related functions. The time of day will be available to the AADS from the operating system. The operating system will also synchronize its clock after a "synchronize clock" command from the ground to the operating system. However, certain schedule tables which the executive maintains should be updated as a result of the time synchronization. In addition, this means that the SET CLOCK command specified in the requirements is not meaningful. Systex personnel will study the problem of time synchronization further.

E. Lefferts suggested that a modified pairwise technique should be employed so that the AADS will be able to compute attitude when the initial attitude estimate is in error by a large amount (2-3 degrees). All possible candidates for all observed stars will be stored in memory. A pairwise matching scheme using each candidate star will provide positive star identification. Task personnel

Mr. K. Tasaki
Mr. F. Snow

-2-

August 20, 1981

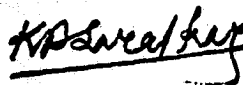
feel that the current top-level design can be modified easily to permit this enhancement; however, the detailed design of the new pairwise matching algorithm will require additional design effort during the follow-on task. GSFC and CSC personnel will study the star identification methods to determine whether the attitude computation requirements are satisfied in 1) normal case nominal attitude, 2) attitude drifting away from the nominal attitude; 3) attitude acquisition (approaching nominal attitude).

E. Lefferts also suggested that the AADS can be completely autonomous by the addition of a Sun sensor/magnetometer (Sun/Mag) or Infrared sensor/Sun sensor (IR/Sun) attitude computation system to AADS. CSC personnel feel that this will require a major redesign effort, particularly, to reevaluate time and memory resources. Therefore, such a capability should be added as a major enhancement to the AADS during a future task.

Occultation of star cameras by Sun, Earth, and Moon was discussed. In the current design, the OBC computes and transmits an occultation schedule to AADS upon request. The ATR requested that the task use an alternate means (such as the bright object sensor (BOS) associated with the star camera) to obtain this information to minimize AADS-to-OBC communication. Task personnel will study this problem.

In response to a CSC question, F. Snow replied that the minimum gyro propagation period is 512 milliseconds. However, gyro data will be sampled (approximately) every 64 milliseconds or more. The SENSIN task should check for gyro data saturation at that frequency.

Yours truly,



Dr. K. Saralkar
Task Leader
Attitude Systems Development

KS:gsp
copies:

GSFC

F. McGarry
E. Lefferts

CSC

S. Cheuvront
G. Page
K. Liu
C. Shenitz
G. Klitsch
Y. Frenkel

COMPUTER SCIENCES CORPORATION

SYSTEM SCIENCES DIVISION

(301) 589-1545

3728 COLESVILLE ROAD · SILVER SPRING, MARYLAND 20910

August 20, 1981

National Aeronautics and Space Administration
Goddard Space Flight Center
Greenbelt, Maryland 20771

Attention: Mr. K. Tasaki
Code 581.2
Bldg. 23, Room E-225
and

Mr. F. Snow
Code 582.3
Bldg. 23, Room E-239

Subject: Contract NAS-523400
Task Assignment 92400
Autonomous Attitude Determination System (AADS)

Dear Messrs. Tasaki and Snow:

This memorandum summarizes CSC's understanding of the topics discussed at a meeting held on August 7, 1981, at GSFC to review task status. Present were F. McGarry, F. Snow, and K. Tasaki of GSFC; K. Liu, G. Page, K. Saralkar, and C. Shentiz of CSC.

C. Shenitz reviewed the design of the input and output tasks. In response to an earlier request from the ATR, data flow diagrams showing external and internal task interfaces were prepared and reviewed at the meeting. In addition, the physical procedure for reading ground commands was discussed in detail. The following sequence describes the main features of the input task.

- Upon activation of AADS, the input task issues a real request to the OBC and waits for a command from the ground (via OBC), or an ephemeris transmission from the OBC. AADS processing goes on normally while the input task waits at a high priority.
- The operating system (for the attitude computer) receives a transmission from OBC and stores it in a buffer specified in the read request, and then 'wakes up' the waiting input system with a read completion interrupt. Any active task is suspended. The operating system also sends the transmission acknowledged signals (ACK/NAK) as specified by the communications protocol.

Mr. K. Tasaki
Mr. F. Snow

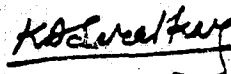
-2-

August 20, 1981

- The input task decodes the commands and sets flags for the executive. The ephemeris table is updated if requested ephemeris was received, and data for table update are stored for later processing. Some minimal input validation is performed. At this stage the input system issues another read request and suspends itself. AADS resumes normal processing and any waiting or suspended tasks are resumed according to their priorities.
- The executive task will schedule the input task at the end of a major cycle (attitude update completed) to validate ephemeris and table updates, and to modify the tables. Tables are modified at the end of a major cycle to avoid possible disruption of attitude computation during a major cycle. If the table areas must be modified immediately, then the ground will send commands to STOP processing, update table areas, then START processing. It is expected that such action will be required infrequently.
- AADS generates error messages when received transmissions fail validation checks. Other than these error messages, regular acknowledgement of input is not planned at this time. This is done to minimize communication with the OBC.
- System personnel and AADS simulator development personnel will determine whether ACK-NAK procedures are required at the applications level between all communication interfaces, including the output from the AADS to the OBC.

CSC recommended that a software clock be used for both the AADS and the AADS simulator so that processing can be speeded up during testing. The software clock will contain a scale factor that will allow the clock to speed up in relation to the external world. This will also allow task personnel to use the online dynamic debugging technique (DDT) for testing. Use of DDT with a realtime clock is not possible because of the synchronization requirements. In summary, the software clock will enable CSC to improve testing efficiency without requesting many high priority jobs or programmer present blocktimes. The ATRs and GSFC personnel developing the AADS simulator are considering this suggestion.

Yours truly,



Dr. K. Saralkar
Task Leader
Attitude Systems Development

KS:gsp

copies: GSFC

F. McGarry
E. Lefferts

CSC

S. Cheuvront	C. Shenitz
G. Page	G. Klitsch
K. Liu	Y. Frenkel

COMPUTER SCIENCES CORPORATION

SYSTEM SCIENCES DIVISION

(301) 589-1545

8728 COLESVILLE ROAD • SILVER SPRING, MARYLAND 20910

August 20, 1981

National Aeronautics and Space Administration
Goddard Space Flight Center
Greenbelt, Maryland 20771

Attention: Mr. K. Tasaki
Code 581.2
Bldg. 23, Room E-225
and
Mr. F. Snow
Code 582.3
Bldg. 23, Room E-239

Subject: Contract NAS-524300
Task Assignment 92400
Autonomous Attitude Determination System (AADS)

Dear Messrs. Tasaki and Snow:

This memorandum summarizes CSC's understanding of the topics discussed at a meeting held on August 14, 1981, at GSFC to review task status. Present were L. Jun, R. Shwenk, F. Snow, and K. Tasaki of GSFC; S. Sanders of Systex; and G. Klitsch, K. Liu, K. Saralkar, Y. Frenkel, and C. Shenitz of CSC.

S. Sanders discussed data transmission and a system clock. Data can be transmitted between the VAX-11/780 and the Intel/8086 over two sets of 16-bit, full duplex paths. Ground commands, data, and ephemeris will be transmitted as 256-byte blocked records. Blocking will be done by the applications program according to a predefined format. FORTRAN callable utilities will be supplied by Systex for transmitting (writing) and receiving (reading) data over the communications paths.

These routines (to be supplied) will specify the buffer area for data, number of records to be transmitted/received, and a unit for data transmission/reception. The read routine will read incoming records in a loop with WAIT/NOWAIT and timeout options. The timeout option will allow the applications software to abort the read if all expected records are not received in a given time due to either loss of synchronization or transmission failure. The applications software should validate transmissions (checksums, sync bytes, etc). The actual transfer of a single record will take approximately 6.4 milliseconds at approximately 50 microseconds per word of two bytes.

The data transmission scheme outlined above will also be used for transmitting AADS simulator generated sensor data to the AADS. The current design assumes (as discussed in a previous design walkthrough) that the sensor data will be written to a buffer in AADS by a Remote Interface Unit (RIU) using a direct memory access (DMA) or memory sharing scheme. This would take a negligible amount of time from the AADS point of view. The current design will have to be modified because of the new requirements. The gyro data are available every 64 milliseconds, and the star tracker is available every 50 milliseconds. The data transfer time (6.4 milliseconds every 64 milliseconds) will be excessive if the standard 256-byte record is used for sensor data transmission. A smaller record (approximately 30 bytes) will take between 1 to 2 milliseconds. CSC will study the new data transmission procedures to determine the modifications to the current design.

The Intel/8086 operating system will maintain the time of day for the AADS scheduling and time tagging requirements. The time of day will be initially loaded when the Intel operating system is loaded by VAX. The clock will be maintained with a periodic (approximately 10 milliseconds) pulse sent over a separate line between the Intel and VAX computers. The RESET CLOCK command is thus unnecessary for this scheme and will not be required for the prototype software. CSC personnel recommended that a single, central time base (clock) with a 1 millisecond pulse should be used in the final version. The single time base is needed to maintain consistent times for all computers onboard the spacecraft, and the 1 millisecond resolution is required to satisfy the requirement of processing gyro data every 64 milliseconds. This requirement may be satisfied in the current version by using a separate 1 millisecond clock to maintain small time differences after the main 10 millisecond pulse arrives.

S. Sanders will investigate and respond to questions on:

- size and capabilities of the mathematical subroutine package
- arithmetic exceptions
- addressing exceptions

G. Klitsch reviewed the top-level design of the executive task. Data flow diagrams were presented to indicate major external and internal interfaces, and flowcharts were used to demonstrate executive scheduling of tasks during normal processing and maneuver

Mr. K. Tasaki
Mr. F. Snow

-3-

August 20, 1981

and occultation periods. Attendees discussed occultation computation and star catalog size and structure. These points will be documented later as answers to questions.

Yours truly,



Dr. K. Saralkar
Task Leader
Attitude Systems Development

KS:gsp
copies:

GSFC

F. McGarry
E. Lefferts

CSC

S. Cheuvront
G. Page
K. Liu
C. Shenitz
G. Klitsch
Y. Frenkel

COMPUTER SCIENCES CORPORATION

SYSTEM SCIENCES DIVISION

301 589-1545

3728 COLESVILLE ROAD • SILVER SPRING, MARYLAND 20910

August 25, 1981

National Aeronautics and Space Administration
Goddard Space Flight Center
Greenbelt, Maryland 20771

Attention: Mr. K. Tasaki
Code 581.2
Bldg. 23, Room E-225
and
Mr. F. Snow
Code 582.3
Bldg. 23, Room E-239

Subject: Contract NAS-524300
Task Assignment 92400
Autonomous Attitude Determination System (AADS)

Dear Messrs. Tasaki and Snow:

This memorandum summarizes CSC's understanding of the topics discussed at a meeting held on August 21, 1981, at GSFC to review task status. Present were L. Jun, F. Snow, M. Stark, R. Shwenk, and K. Tasaki of GSFC; S. Sanders of Systex; and Y. Frenkel, G. Klitsch, K. Liu, K. Saralkar, and C. Shenitz of CSC.

S. Sanders discussed data transmission and the Intel operating system. Gyro and sensor data will be stored in the AADS memory using direct memory access (DMA), because of the excessive input/output (I/O) time needed to send the data as blocked records. The DMA transfer requires negligible amount of time, and simulates the Remote Interface Unit (RIU) that will be used onboard the spacecraft. This information supersedes the description of data transmission given in the previous letter of August 20, 1981, and will be documented in greater detail in a question to the ATR.

S. Sanders gave the following information in response to CSC questions:

- Arithmetic exception (divide by zero) will produce an interrupt. The operating system will then pass control to an error handling task in AADS.

Mr. K. Tasaki
Mr. F. Snow

-2-

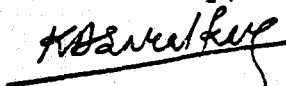
August 25, 1981

- Addressing protection is available if an invalid address beyond available memory space is generated, but addressing protection is not available for individual tasks; a task can still write in the memory space of another task. Again, in the event of an addressing error, the operating system will pass control to an error handling task.
- The current VAX clock produces a timer interrupt once every 10 milliseconds. This does not provide sufficient accuracy for gyro data and star camera data processing requirements. GSFC is considering buying a 1 millisecond clock.
- The Intel will have 224 kilobytes of random access memory (RAM), and 16 kilobytes of read only memory (ROM). Thus, AADS does not have the full 256 kilobyte memory as was specified previously for the Intel.

C. Shenitz reviewed the design for the input and output tasks and described formats for ephemeris request and error messages.

L. Jun, M. Stark, and R. Shwenk briefly reviewed the status of various subsystems in the AADS simulator. Onboard support system (OSS) will send requested ephemeris to AADS via the VAX mailbox facility. The format of the ephemeris table was defined by the attendees.

Yours truly,



Dr. K. Saralkar
Task Leader
Attitude Systems Development

KS:gsp

copies: GSFC

F. McGarry
E. Lefferts

CSC

S. Cheuvront
G. Page
K. Liu
C. Shenitz
G. Klitsch
Y. Frenkel

COMPUTER SCIENCES CORPORATION

SYSTEM SCIENCES DIVISION

(301) 589-1545

8728 COLESVILLE ROAD • SILVER SPRING, MARYLAND 20910

September 1, 1981

National Aeronautics and Space Administration
Goddard Space Flight Center
Greenbelt, Maryland 20771

Attention: Mr. K. Tasaki
Code 581.2
Bldg. 23, Room E-225
and

Mr. F. Snow
Code 582.3
Bldg. 23, Room E-239

Subject: Contract NAS-524300
Task Assignment 92400
Autonomous Attitude Determination System (AADS)

Dear Messrs. Tasaki and Snow:

This memorandum summarizes CSC's understanding of the topics discussed at a meeting held on August 28, 1981, at GSFC to review task status. Present were L. Jun, R. Shwenk, F. Snow, M. Stark, and K. Tasaki of GSFC; S. Sanders of Systex; Y. Frenkel, G. Klitsch, K. Liu, K. Saralkar, and C. Shenitz of CSC.

K. Tasaki discussed the procedure for starting AADS simulation. A simulation tape containing gyro and star tracker data will be prepared by using the Landsat-D Simulator. Predicted attitude information will be stored on the tape along with data. The tape header will contain values for simulation control variables (orbital elements, biases, misalignments, etc.). AADS will be loaded into the Intel memory, and the Intel clock will be initialized with the simulation time. The Sensor Data Simulator (SDS) will be invoked and start transmitting gyro data to AADS. At this point, there are several ways of synchronizing the activities of AADS and the AADS simulator tasks. The detailed startup procedure and synchronization will be documented in question number EX016.

ORIGINAL PAGE IS
OF POOR QUALITY

Mr. K. Tasaki
Mr. F. Snow

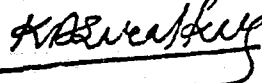
-2-

September 1, 1981

S. Sanders reviewed the services available through the Intel operating system and gave the following information in response to questions posed by the design team:

- A new operating system will be developed for the Intel because the Intel RMX operating system provides services which are somewhat different from those provided by the VAX/VMS operating system. Some portions of AADS will have to be redesigned to test AADS in the Intel-VAX system configuration if the old Intel operating system is used.
- S. Sanders will distribute a list of operating system services to the design team. These services should be available to AADS through the new operating system being developed by Systex.
- The mathematical subroutine library can be shared among all tasks if the library routines use a stack to store intermediate calculations. However, if the library routines use a fixed area in memory, then each task must have a separate copy of the mathematical subroutine library. Further details about the library and a link procedure that will be used to link the AADS tasks will be documented in question number EX017.
- Uplink and downlink messages can be longer than 256 bytes.

Yours truly,



Dr. K. Saralkar
Task Leader
Attitude Systems Development

KS:gsp

copies: GSFC

F. McGarry
E. Lefferts
A. Fuchs

CSC

S. Chevront
G. Page
K. Liu
C. Shenitz
G. Klitsch
Y. Frenkel

September 8, 1981

TO: Distribution

FROM: 582.1/Systems Development and Analysis Branch

SUBJECT: Results from the Autonomous Attitude Determination System (AADS)
Critical Design Review

A critical design review (CDR) of Autonomous Attitude Determination System (AADS), which included both the proposed onboard attitude system and the support simulator, was held on Tuesday, September 1, 1981. The presentation of the review material was shared by the following persons:

Introduction	K. Tasaki/GSFC/582.1
Design of Ground Support Subsystem	M. Stark/GSFC/582.1
Design of Onboard Support Subsystem	L. Jun/GSFC/582.1
Design of Sensor Data Subsystem	R. Schwenk/GSFC/582.1
Design of System Software	S. Sanders/SYSTEX, Inc.
AADS Overview	K. Saralkar/CSC/Task 924
Design of AADS Executive	G. Klitsch/CSC/Task 924
Design of I/O Subsystem	C. Shenitz/CSC/Task 924
Overview of Mathematical Modules	K. Liu/CSC/Task 924
Development Schedule	K. Tasaki/GSFC/582.1

This CDR was primarily to cover the AADS Executive and the AADS I/O subsystem for which the detailed design has been completed. The CDR for the attitude propagation/determination modules has been scheduled tentatively for November 3, 1981. The design of the three subsystems for the AADS simulator was also presented along with the description of the required system support software.

The following points and action items were noted during and after the CDR:

1. The Landsat data tape format design by R. Schwenk will be presented to the Landsat Simulator task personnel for their comments since they will be providing the data tape to the AADS simulator group. Arrangements will be made between the two groups for the generation of data tapes after the contents and the format are finalized.

Action Item: R. Schwenk and K. Tasaki will meet with the Landsat simulation group within a week to begin discussion.

2. The function of the remote interface unit (RIU) needs to be better understood since the raw gyro and star tracker data are made available to the rest of the S/C via this piece of hardware. In particular, the bit patterns and the field definitions for the two sensor data types, the operating characteristics, the delay between the time of observation and the arrival time of the data in RIU, and other relevant information must be known in order to accurately simulate this device.

Action Item: F. Snow will obtain the information within 2 weeks.

3. The simulation of uplinking and downlinking of AADS code was eliminated from the original set of requirements. To simulate the downlinking of AADS code and data areas (memory dump) can be done without much difficulty. However, the simulation of uplinking the code via the AADS simulator is another matter. It is very mission-dependent, i.e., we have to know the boot-strap procedure of the onboard processor into which AADS is to be loaded. The boot program, which must reside in ROM, must be a specialized program to recognize AADS message format and protocols. For this reason, it is proposed that only the memory dump capability be implemented.

Action Item: K. Tasaki will further investigate the feasibility with the Task 924 personnel and accordingly modify the requirements and the design.

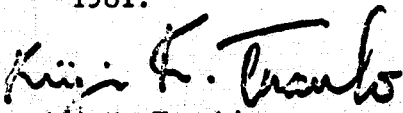
4. As stated on page 1-33 of the CDR review material, all intertask communication will be via global COMMON, and the SEND/RECEIVE type of system services will not be used. In addition, Task 924 personnel will examine the list of system services routines proposed by SYSTEX for INTEL operating system implementation.

Action Item: K. Saralkar will examine and reduce the number of the system services routine by September 30, 1981.

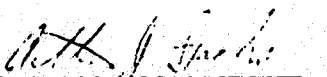
5. A comment was made concerning the possibility of having a software clock to synchronize the timing among all Processes and to control the simulation, i.e., eliminating the dead time, slowing down the simulation when desired, etc. During the PDR held on February 23, 1981, a point was made that the final AADS must be demonstrated in real time. With 50 and 64 millisecond cycle times for the star tracker data and the gyro data, respectively, the elimination of any dead time within these cycles will lock the VAX CPU disabling any other user on the VAX to execute his program. It is envisioned that stand-alone times on the VAX will not be allowed. For these reasons, the feature to eliminate dead times will not be incorporated into AADS. For a long simulation case, i.e., a few days, AADS and the simulator can be set up to run several hours at a time introducing the last computed attitude as input at the time of restart.

6. There are a number of ways to implement the Kalman filter, i.e., the Joseph form, the UDUT, the standard approach, etc., primarily to minimize the execution time while retaining sufficient accuracy and stability. A choice should be made soon so that there will be no impact on the software design.

Action Item: F. Snow and E. Lefferts will make the decision with C. Shenitz and K. Liu of Task 924 on the choice of filter implementation by September 30, 1981.


Keiji K. Tasaki

Concurrence:


Arthur J. Fuchs

ORIGINAL PAGE 19
OF POOR QUALITY

COMPUTATIONAL COMMUNITIES CORPORATION
SYSTEM SCIENCES DIVISION (301) 589-1545
8778 COLLEVILLE ROAD • SILVER SPRING, MARYLAND 20910

September 9, 1981

National Aeronautics and Space Administration
Goddard Space Flight Center
Greenbelt, Maryland 20771

Attention: Mr. K. Tasaki
Code 581.2
Bldg. 23, Room E-225
and

Mr. F. Snow
Code 582.3
Bldg. 23, Room E-239

Subject: Contact NAS-524300
Task Assignment 92400
Autonomous Attitude Determination System (AADS)
Critical Design Review, September 1, 1981

Dear Messrs. Tasaki and Snow:

This memorandum summarizes CSC's understanding of the topics discussed at the Critical Design Review (CDR) held on September 1, 1981, at GSFC. Present were A. Fuchs, L. Jun, E. Lefferts, R. Shwenk, F. Snow, M. Stark, K. Tasaki, and R. Werking of GSFC; S. Sanders of Systex; and T. Brown, V. Church, Y. Frenkel, G. Klitsch, K. Liu, G. Page, K. Saralkar, and C. Shenitz of CSC.

K. Tasaki gave an overview of AADS and the AADS simulator, and presented a schedule for completing development and testing of AADS and related software. M. Stark, L. Jun, and R. Shwenk presented, respectively, the ground support simulator (GSS), the onboard support simulator (OSS), and the sensor data simulator (SDS). S. Sanders presented the Intel operating system (Intel/OS), and described the Intel development system and the Intel-VAX system configuration that will be used during AADS testing; K. Saralkar gave an overview of AADS; K. Liu reviewed the top-level design for the attitude computation function; G. Klitsch reviewed the executive task and described scheduling logic; and C. Shenitz reviewed the input, output, and sensor data collection tasks.

Mr. R. Tasaki
Mr. F. Snow

-2-

September 9, 1981

No major changes will be required to the current AADS design as a result of the CDR; however, several questions were raised during the discussion, particularly about the interfaces between the various simulator tasks and AADS. A summary of these questions is given below. Some of these questions will be documented in questions to the ATRs.

- The Landsat-D simulator currently being developed by CSC (Task 91100) will be used to provide test data for AADS.
- GSFC personnel will investigate the hardware specification for the gyro and star cameras used for the Landsat-D simulator. GSFC personnel will also investigate the hardware specifications for the Remote Interface Unit (RIU) and obtain more information about sensor data collection and transmission onboard the spacecraft.
- GSFC personnel will investigate the capabilities and specifications of the Landsat-D simulator with respect to sensor data format, predicted attitude, observed star information, and header information containing the simulation options.
- AADS will require several days (greater than 3) of continuous simulation data. GSFC personnel will determine whether the Landsat-D simulator can support data spans that long.
- The Landsat-D simulator does not simulate scanning of the field-of-view for new stars. A bit in the star camera data indicates star presence. The design team will obtain the complete format of star camera data to determine whether the current design properly supports processing the data even when the star cameras are turned off.
- GSS will provide for checking the accuracy of AADS computation and star identification. A plot package will be needed in GSS to generate line printer plots of the computed versus predicted (simulator) attitude.
- GSFC personnel will determine if the Joseph form of the Kalman filter, which is specified for state update in AADS, is suitable with respect to execution time and roundoff error elimination.
- The design team will investigate the possibility of using a software clock for AADS and the AADS simulator. The software clock will speed up AADS testing, and will be particularly useful for the long (3-to-4-day) test runs planned for AADS.
- AADS will be stored in random access memory (RAM). The Intel/OS will support uplink/downlink of AADS in the event of RAM failure or for debugging purposes. The design team will identify all such capabilities required to maintain AADS onboard the spacecraft.

Mr. K. Tasaki
Mr. F. Snow

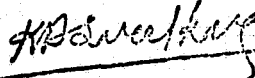
-3-

September 9, 1981

- The design team will determine the communications protocol between AADS and the OBC, since the OBC is expected to transmit data annotation results to the ground.
- The Intel/OS will simulate the RIU by accepting the sensor data transmitted by SDS. Star data from alternate cameras will be available at 50-millisecond periods. These data will be stored in two separate locations in AADS memory by the Intel/OS. Gyro data will be stored in a separate location every 64 milliseconds. The sensor data collection task in AADS will read gyro data once every 64 milliseconds and star data from both cameras once every 100 milliseconds.
- It is expected that the RIU will also store sensor data in AADS memory at known intervals. However, the sensors and data format are mission dependent.
- AADS will perform autonomously as long as the attitude error is small (less than 2 degrees) and spacecraft ephemeris data are available from the OBC. AADS is currently designed to perform gyro propagation and data annotation at a minimum period of 512 milliseconds.

Minor questions on occultation computation, maneuver handling, star catalog size and structure, pairwise and direct star matching techniques, and the capabilities of the Intel/OS were discussed. These will be documented as questions where appropriate.

Yours truly,



Dr. K. Saralkar
Task Leader
Attitude Systems Development

KS:gsp

copies: GSFC

F. McGarry
E. Lefferts
A. Fuchs

CSC

S. Cheuvront
G. Page
K. Liu
C. Shenitz
G. Klitsch
Y. Frenkel

ORIGINAL PAGE 19
OF POOR QUALITY

COMPUTER SYSTEMS CORPORATION
SYSTEM SCIENCES DIVISION (301) 589-1545
8728 COLESVILLE ROAD • SILVER SPRING, MARYLAND 20910

September 9, 1981

National Aeronautics and Space Administration
Goddard Space Flight Center
Greenbelt, Maryland 20771

Attention: Mr. K. Tasaki
Code 581.2
Bldg. 23, Room E-225
and
Mr. F. Snow
Code 582.3
Bldg. 23, Room E-239

Subject: Contract NAS-524300
Task Assignment 92400
Autonomous Attitude Determination System (AADS)
Follow-on Task Assignment

Dear Messrs. Tasaki and Snow:

This memorandum summarizes CSC's understanding of the decisions made at a meeting held on September 4, 1981, at GSFC to review and clarify the requirements for the microprocessor based autonomous attitude determination system (AADS). Present were F. Snow and K. Tasaki of GSFC; and G. Page and K. Saralkar of CSC. The following decisions were made during the meeting:

- Baseline diagrams, I/O definitions and the process descriptions will be delivered to the ATRs 1 week before the Critical Design Review for the attitude computation function. Subroutine prologs will be available on the VAX computer at this time, and can be obtained from the librarian upon request.
- AADS testing will be completed on the VAX computer by the end of the follow-on task period. However, the amount of testing in the Intel-VAX system configuration will depend on the availability of the Intel operating system and other hardware required for Intel-VAX testing.
- The new Intel operating system being developed for AADS by Systex will be compatible with the VAX/VMS operating system used during AADS development on the VAX. In addition, the Intel FORTRAN/86 compiler is expected to be compatible with VAX FORTRAN (Version 2.0). Under these two assumptions, the conversion of AADS to the Intel should be straightforward.

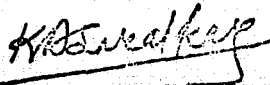
Mr. K. Tasaki
Mr. F. Snow

-2-

September 9, 1981

- CSC personnel recommend that the new Intel operating system should be thoroughly tested by Systex personnel with the AADS skeleton as early as possible to uncover any interface problems with the AADS executive. AADS testing in the Intel-VAX system configuration could be severely impacted if any interface problems remain at this point.
- The final system description document will include sections on 1) system overview, 2) requirements and specifications summary, 3) descriptions for all control variables in global COMMON areas, 4) system execution and computer resources, 5) baseline diagrams and processing overviews for all tasks, 6) basic algorithms, and 7) subroutine prologs. This document will evolve from the design document that was prepared under Task 92400. Section 2 will be completed by incorporating information from all task letters and answers to the questions. Section 2 will have a subsection on changed and new requirements. Sections 3, 5, 6, and 7 are mainly complete at this stage and will be modified as required during the design of the attitude computation function. Section 4 will describe file structure and commands necessary for system generation and execution.
- CSC personnel will prepare a system task plan for the testing of AADS. The test plan will have a detailed description of test data, procedures to run the tests, and the expected results. Test evaluations will compare the actual results with the expected results. The final evaluation report document will be prepared by adding the test reports to the system test plan. The AADS system test plan will be similar to the acceptance test plans prepared for the Dynamics-B Explorer Attitude Determination System, but the precise format of this document will be defined later.
- GSFC personnel will prepare a user's guide for AADS. This guide will be written for both the system operator and the system analyst. CSC will provide material on the use of the filter that is used to update the state vector.

Yours truly,



Dr. K. Saralkar
Task Leader
Attitude Systems Development

KS:gsp
copies:

GSFC

F. McGarry
E. Lefferts
A. Fuchs

CSC

S. Cheuvront
G. Page
K. Liu

C. Shenitz
G. Klitsch
Y. Frenkel

COMPUTER SCIENCES CORPORATION

SYSTEM SCIENCES DIVISION

(301) 589-1545

8728 COLESVILLE ROAD • SILVER SPRING, MARYLAND 20910

October 1, 1981

National Aeronautics and Space Administration
Goddard Space Flight Center
Greenbelt, Maryland 20771

Attention: Mr. K. Tasaki
Code 581.2
Bldg. 23, Room E-225
and
Mr. F. Snow
Code 582.3
Bldg. 23, Room E-239

Subject: Contract NAS-524300
Task Assignment 92400
Autonomous Attitude Determination System (AADS)

Dear Messrs. Tasaki and Snow:

This memorandum summarizes CSC's understanding of the topics discussed at a meeting held on September 18, 1981, at GSFC to review task status. Present were K. Tasaki of GSFC; and G. Klitsch, K. Saralkar, and C. Shenitz of CSC.

The attendees discussed system test procedures including the startup scenario and specific build 1 capabilities to be tested during the initial period. The following decisions were made during the meeting:

- Sensor Data Simulator (SDS) will be activated a short time before AADS is started. This will simulate the actual onboard conditions.
- Task scheduling and the interfaces between AADS and the AADS simulator will be tested in build 1 testing.
- Landsat-D simulator data will be available on or before November 18, 1981. Therefore, SDS will send dummy data to AADS to verify the interface between SDS and AADS.
- R. Shwenk will study the software clock design proposed by C. Shenitz to determine the impact on the SDS.

Mr. K. Tasaki
Mr. F. Snow

-2-

October 1, 1981

- The design team will examine the need for a restart capability.

Yours truly,

K. Saralkar

Dr. K. Saralkar
Task Leader
Attitude Systems Development

KS:gsp

copies: GSFC

F. McGarry
E. Lefferts
A. Fuchs

CSC

S. Chevront
G. Page
K. Liu
C. Shenitz
G. Klitsch
Y. Frenkel

COMPUTER SCIENCES CORPORATION

SYSTEM SCIENCES DIVISION

(301) 589-1545

8728 COLESVILLE ROAD • SILVER SPRING, MARYLAND 20910

October 1, 1981

National Aeronautics and Space Administration
Goddard Space Flight Center
Greenbelt, Maryland 20771

Attention: Mr. K. Tasaki
Code 581.2
Bldg. 23, Room E-225
and

Mr. F. Snow
Code 582.3
Bldg. 23, Room E-239

Subject: Contract NAS-524300
Task Assignment 92400
Autonomous Attitude Determination System (AADS)

Dear Messrs. Tasaki and Snow:

This memorandum summarizes CSC's understanding of the topics discussed at a meeting held on September 25, 1981, at GSFC to review task status. Present were L. Jun, F. McGarry, R. Shwenk, F. Snow, M. Stark, and K. Tasaki of GSFC; S. Sanders of Systex; and Y. Frenkel, G. Klitsch, K. Liu, K. Saralkar, and C. Shenitz of CSC.

S. Sanders reviewed the status of the Intel system software. He reported that the mathematical subroutine library requires approximately 2.5 kilobytes of memory, and it can be shared by all tasks in AADS. The attendees then reviewed the FORTRAN source to verify interfaces between AADS and the AADS simulator.

The following points were noted during the discussion that followed:

- The Sensor Data Simulator (SDS) will write gyro and star tracker data to a mailbox. SDS will then write the contents of the mailbox to a global COMMON accessible to the sensor data read task in AADS. A system routine will write the contents of the mailbox to a specific Intel memory location in the Intel version of the software.
- The star data times will be in error by 50 milliseconds (maximum), because the data are read every 50 milliseconds. A signal is available onboard which indicates how much time has elapsed since the data were accessed by the Remote Interface Unit (RIU). F. Snow will obtain more details about this signal.

Mr. K. Tasaki
Mr. F. Snow

-2-

October 1, 1981

- The ephemeris request from AADS to the Onboard Support System (OSS) will contain the start time for the ephemeris in seconds since September 1, 1957, 0 hours UT. The ephemeris interval will be given in seconds. OSS will return the spacecraft-to-Earth vector in kilometers, and the spacecraft velocity relative to the Earth in kilometers/second.
- The design team recommends that AADS should be tested using a 4-day data segment to verify the autonomy goal. However, the actual requirement may have to be set lower because long spans of data may not be available. F. Snow will determine the autonomy requirement.

Yours truly,



Dr. K. Saralkar
Task Leader
Attitude Systems Development

KS:gsp

copies: GSFC

F. McGarry
E. Lefferts
A. Fuchs

CSC

S. Cheuvront
G. Page
K. Liu
C. Shenitz
G. Klitsch
Y. Frenkel

COMPUTER SCIENCES CORPORATION

SYSTEM SCIENCES DIVISION

(301) 582-545

3728 COLESVILLE ROAD • SILVER SPRING, MARYLAND 20910

October 5, 1981

National Aeronautics and Space Administration
Goddard Space Flight Center
Greenbelt, Maryland 20771

Attention: Mr. K. Tasaki
Code 581.2
Bldg. 23, Room E-225
and
Mr. F. Snow
Code 582.3
Bldg. 23, Room E-239

Subject: Contract NAS-524300
Task Assignment 94200
Autonomous Attitude Determination System (AADS)

Dear Messrs. Tasaki and Snow:

This memorandum summarizes CSC's understanding of the topics discussed at a meeting held on October 2, 1981, at GSFC to review task status. Present were L. Jun, R. Shwenk, F. Snow, M. Stark, and K. Tasaki of GSFC; S. Sanders of Systex, and Y. Frenkel, G. Klitsch, K. Liu, and K. Saralkar of CSC. The following points were noted during the meeting:

- The design team will prepare a configuration control plan that will protect the source and task images, control the implementation of new capabilities, eliminate errors, and provide a mechanism for reporting the status of system testing to the ATRs and to GSFC and CSC management.
- The design team will prepare Digital Command Language (DCL) procedures to initiate AADS and the AADS Simulator and to create task images.
- K. Tasaki will determine a procedure for downlinking/uplinking an AADS image to the Intel memory during a mission.
- The design team will study the procedure to uplink processing options to AADS.
- CSC personnel delivered a list of the required Intel operating system services to the ATR. Some previously specified system services were eliminated and several other system services will be replaced with stubs to reduce coding changes in the Intel version of AADS.


Mr. K. Tasaki
Mr. F. Snow

-2-

October 5, 1981

- There are some differences between VAX-11 FORTRAN and Intel FORTRAN. K. Tasaki will provide the available Intel FORTRAN documentation to CSC personnel. Task members will use the Intel FORTRAN conventions as far as possible.

Yours truly,



Dr. K. Saralkar
Task Leader
Attitude Systems Development

KS:gsp
copies:

GSFC

F. McGarry
E. Lefferts
A. Fuchs

CSC

S. Cheuvront
G. Page
K. Liu
C. Shenitz
G. Klitsch
Y. Frenkel

COMPUTER SCIENCES CORPORATION

SYSTEM SCIENCES DIVISION

(301) 589-1545

3728 COLESVILLE ROAD • SILVER SPRING, MARYLAND 20910

October 13, 1981

National Aeronautics and Space Administration
Goddard Space Flight Center
Greenbelt, Maryland 20771

Attention: Mr. K. Tasaki
Code 581.2
Bldg. 23, Room E-225
and

Mr. F. Snow
Code 582.3
Bldg. 23, Room E-239

Subject: Contract NAS 5-24300
Task Assignment 94200
Autonomous Attitude Determination System (AADS)

Dear Messrs. Tasaki and Snow:

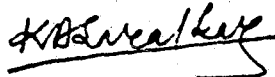
This memorandum summarizes CSC's understanding of the topics discussed at a meeting held on October 7, 1981, at GSFC to review task status. Present were L. Jun, F. McGarry, M. Stark, and K. Tasaki of GSFC; S. Sanders of Systex; and G. Klitsch, K. Saralkar, and C. Shenitz of CSC. The following points were noted during the meeting:

- In the Intel version of AADS, sensor data are made available to AADS using Direct Memory Access (DMA). Gyro and star tracker data are written to a specified area in AADS memory and then read by the sensor data read task. It is possible that some data may be lost if the DMA write and subsequent read functions are not synchronized. The problem could be eliminated by disabling the DMA write while the data written earlier is being read by AADS. The attendees agreed that more information is needed about the sensor data collection function of the Remote Interface Unit (RIU) onboard the spacecraft before any particular hardware or software model of the RIU is selected.
- The ATRs will review and then provide comments on the outlines provided by CSC for the AADS system description manual, the system test plan and evaluation report, and the configuration control procedures.

October 13, 1981

- The ATRs will provide input toward selecting the test data sets to be used during system testing. Ten separate tests and data sets will be defined for VAX system testing. The same tests and data sets will be used for testing the Intel version of AADS.
- CSC will prepare a short, informal test summary after system testing of each of the 4 builds on the VAX.
- COMMON/STATIC/ contains all variables that control AADS processing. F. Snow will review the descriptions of these variables (VAX file {AADS.GL}STATIC.FOR) and determine the correctness of the variable definitions.
- K. Saralkar will decompose COMMON/STATIC/ into 4 separate COMMON areas so that a single category of control variables (e.g. task schedules and priorities) can be uplinked to AADS independently of other control options.
- There are some differences between the VAX and Intel internal representations for numeric data. The Ground Support System (GSS) will convert data to the Intel format before uplinking, and convert the Intel output to the VAX format. The Sensor Data System (SDS) will convert sensor data to the Intel format before transmitting the data to AADS. This procedure will be used when AADS is implemented on the Intel microcomputer.

Yours truly,



Dr. K. Saralkar
Task Leader
Attitude Systems Development

KS:gsp

copies: GSFC
F. McGarry
E. Lefferts
A. Fuchs

CSC
S. Cheuvront
G. Page
K. Liu
C. Shenitz
G. Klitsch
Y. Frenkel

COMPUTER SCIENCES CORPORATION

SYSTEM SCIENCES DIVISION

(301) 589-1545

8728 COLESVILLE ROAD • SILVER SPRING, MARYLAND 20910

October 19, 1981

National Aeronautics and Space Administration
Goddard Space Flight Center
Greenbelt, Maryland 20771

Attention: Mr. K. Tasaki
Code 581.2
Bldg. 23, Room E-225
and

Mr. F. Snow
Code 582.3
Bldg. 23, Room E-239

Subject: Contract NAS 5-24300
Task Assignment 94200
Autonomous Attitude Determination System (AADS)

Dear Messrs. Tasaki and Snow:

This memorandum summarizes CSC's understanding of the topics discussed at two separate meetings held on October 14, 1981, at GSFC to review task status and star identification methods, respectively.

Present at the first meeting were L. Jun, R. Shwenk, M. Stark, and K. Tasaki of GSFC; S. Sanders of Systex; and Y. Frenkel, G. Klitsch, and K. Saralkar of CSC. The following points were noted during this meeting:

- S. Sanders discussed the software development procedure on the Intel development system. This procedure will be documented in a question to the ATRs.
- K. Saralkar noted that the Intel FORTRAN manual does not describe the DOWHILE construct. This construct is available in the VAX FORTRAN, and is used extensively in the AADS code. K. Tasaki will determine whether the DOWHILE construct is available in the Intel FORTRAN.
- K. Tasaki described a preliminary version of the Digital Command Language (DCL) procedure that will be used to initiate an AADS simulation. At least two terminals will be needed during the simulation. G. Klitsch will provide to K. Tasaki the following information about AADS: 1) process priorities, 2) parameters needed to run the processes, 3) files opened.

K. Tasaki
F. Snow

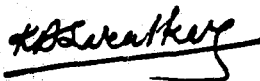
-2-

October 19, 1981

- M. Stark described the difference between VAX and Intel internal numeric data formats. The attendees agreed that the Intel version of AADS will not send a negative floating point zero to the Onboard Support System (OSS) on the VAX, because the VAX treats that particular bit pattern as an error. Ground Support System (GSS) and OSS will perform other required conversions before transmitting or receiving data from AADS.

Present during the second meeting were E. Lefferts, F. Snow, and K. Tasaki of GSFC; Y. Frenkel, G. Klitsch, K. Liu, and K. Saralkar of CSC. K. Liu described the star identification method used by AADS, and also described a triplet matching algorithm used by the Solar Maximum Mission Attitude Determination System (SMM/ADS). The attendees agreed that the modified pairwise matching scheme used in the current design is adequate. This modified pairwise matching scheme and other questions raised during the meeting will be documented in questions to the ATRs.

Yours truly,



Dr. K. Saralkar
Task Leader
Attitude Systems Development

KS:gsp
copies:

GSFC

F. McGarry
E. Lefferts

CSC

S. Cheuvront
G. Page
S. Waligora
K. Liu
C. Shenitz
G. Klitsch
Y. Frenkel

COMPUTER SCIENCES CORPORATION

SYSTEM SCIENCES DIVISION

(301) 589-1545

8728 COLESVILLE ROAD • SILVER SPRING, MARYLAND 20910

National Aeronautics and Space Administration
Goddard Space Flight Center
Greenbelt, Maryland 20771

Attention: Mr. K. Tasaki
Code 581.2
Bldg. 23, Room E-225

and

Mr. F. Snow
Code 582.3
Bldg. 23, Room E-239

Subject: Contract NAS 5-24300
Task Assignment 94200
Autonomous Attitude Determination
System (AADS)

Dear Messrs. Tasaki and Snow:

This memorandum summarizes CSC's understanding of the topics discussed at two separate meetings held on October 21, 1981, at GSFC to review attitude determination function design, and AADS system testing, respectively.

Present at the first meeting were E. Lefferts, R. Shwenk, F. Snow, and K. Tasaki of GSFC; Y. Frenkel, G. Klitsch, K. Liu, and K. Saralkar of CSC. The following points were noted during this meeting:

- In response to a question from CSC, the ATRs replied that the section 5 (AADS Algorithms) of the AADS system description document will contain the state propagation and update analysis done by E. Lefferts for AADS. The ATRs have not specified any other material for inclusion in section 5.
- E. Lefferts described the computation of attitude uncertainty. In the small angle approximation, the diagonal elements of the covariance matrix are the squares of rotational angles. Either the squareroot of the sum of diagonal elements, or the squareroot of the largest diagonal element can be used as a rough measure of the attitude uncertainty.

- K. Liu described the star identification procedure. AADS will compute the attitude uncertainty E , and the window size for accessing catalog stars, $N \cdot E$. N is nominally set to 4 or 5, and can be changed by a ground command. When the attitude is known accurately, the attitude uncertainty, and therefore, the window size is small. The window around an observed star will contain a single candidate star, and direct star identification is used. When attitude uncertainty is large, the window size is also large. It is possible that several candidates may be found for a single observation, and pairwise matching is used for star identification.
- Current AADS design uses doublet stars for star identification. F. Snow will determine if doublet stars should be discarded.
- AADS will calculate the star magnitude from the calibrated star intensity. This magnitude will be compared with the visible star magnitude stored in the AADS star catalog.
- AADS will rotate track point unit vectors to the GCI frame using gyro propagated attitudes. The track point unit vectors in the GCI frame will be then averaged to form a single observation unit vector per star.
- AADS will use the regular form of the Kalman filter for state propagation and update. The filter will be implemented using a standard recursive algorithm in which the identified stars are processed one at a time. The state covariance matrix will be renormalized after every update. E. Lefferts indicated that the diagonal elements of the covariance matrix will not become zero or negative during the computation, but the covariance matrix may lose its symmetric form. The logic for correcting this condition will be added to the design.
- Gyro saturation check will be removed from the sensor data collection (SENSIN) task, and will be done in the gyro data processing (GYPROC) task. Successive points will be used to check for saturation; end points will be used for rate computation. The system executive will schedule an immediate update when the gyro saturation condition is reported by the GYPROC task.
- Spacecraft maneuvers may start and end in the middle of a span of data being processed by the GYPROC and PROPAG tasks. In this case, the data will be processed separately to perform state propagation.

Mr. K. Tasaki
Mr. F. Snow
Page Three

Present during the second meeting were L. Jun, R. Shwenk, M. Stark, and K. Tasaki of GSFC; Y. Frenkel, G. Klitsch, and K. Saralkar of CSC. In response to a request by K. Tasaki, G. Klitsch delivered a list of priorities and system resources for all tasks in AADS. M. Stark and K. Saralkar reviewed the formats for the various reports transmitted by AADS. The attendees discussed AADS system testing completed during the week and planned future test sessions.

Yours truly,



Dr. K. Saralkar
Task Leader
Attitude System Development

KS/stb
copies:

GSFC

F. McGarry
E. Lefferts
A. Fuchs

CSC

S. Cheuvront
G. Page
S. Waligora
K. Liu
C. Shenitz
G. Klitsch
Y. Frenkel

COMPUTER SCIENCES CORPORATION

SYSTEM SCIENCES DIVISION

(301) 589-1545

8728 COLESVILLE ROAD • SILVER SPRING, MARYLAND 20910

November 11, 1981

National Aeronautics and Space Administration
Goddard Space Flight Center
Greenbelt, Maryland 20771

Attention: Mr. K. Tasaki
Code 581.2
Bldg. 23, Room E-225

and

Mr. F. Snow
Code 582.3
Bldg. 23, Room E-239

Subject: Contract NAS 5-24300
Task Assignment 94200
Autonomous Attitude Determination System (AADS)
Critical Design Review, November 3, 1981

Dear Messrs. Tasaki and Snow:

This memorandum summarizes CSC's understanding of the topics discussed at the Critical Design Review (CDR) held on November 3, 1981, at GSFC. Present were A. Fuchs, L. Jun, E. Lefferts, F. McGarry, R. Shwenk, F. Snow, and K. Tasaki of GSFC; Y. Frenkel, G. Klitsch, K. Liu, G. Page, K. Saralkar, C. Shenitz, and S. Waligora of CSC. The design team presented the detailed design for the attitude determination function in AADS, the top level design for which had been presented at an earlier CDR held during the previous task period (Task 92400).

K. Saralkar described the purpose of the CDR and detailed the agenda for the meeting. He then gave a brief overview of AADS, described the major changes in AADS requirements, and current task status. Y. Frenkel presented the design for the gyro data processing (GYPROC) and gyro propagation (PROPAG) tasks. K. Liu presented the design for the star data processing (STRACK), star identification (STARID), and state update (UPDATE) tasks. Y. Frenkel presented the design for a utility which creates an AADS star catalog from the main SKYMAP star catalog. Finally, K. Tasaki discussed the system testing conducted for a skeleton AADS system (which includes the executive and input/output functions), and K. Tasaki and K. Saralkar summarized the results of the CDR.

Mr. K. Tasaki
Mr. F. Snow
Page Two

Following points were noted during the meeting for appropriate future action by the design team:

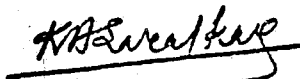
- E. Lefferts expressed a concern about the lengthy calculations being performed during state propagation and update. He suggested that the AADS performance can be improved by bypassing some relatively unimportant calculations, and using approximations and/or alternative formulations where possible. K. Saralkar replied that the predevelopment timing estimates show that AADS can perform all required calculations in approximately half the available time even for the worst case scenario. Therefore, all attitude determination algorithms were implemented in a simple and direct manner without any attempt at time optimization. Time and/or memory optimization will be performed later, if required in light of the knowledge gained during the actual testing of AADS.
- In the current design, gyro data is processed nominally every 512 milliseconds; however, state vector and covariance matrix propagation is performed after N gyro data processing intervals. Here the value of N is nominally equal to 1, but can be optionally set to a higher integral value. E. Lefferts suggested that in order to save computational time, covariance matrix propagation should be separated from state propagation, and should be performed at a lower frequency. That is, the covariance matrix should be propagated after the state vector is propagated M times, where M may be typically set equal to 100. The design team will review this requirement and will outline the necessary design change. The design change will be implemented during build 3 if it is approved by the ATRs.
- E. Lefferts raised additional questions about the implementation of the propagation algorithm and the use of the recursive filter for state update. Y. Frenkel and K. Liu will describe in detail the specific equations, numerical approximations, etc, used for attitude determination, and submit this handwritten material to the ATRs for approval.
- In response to a question from K. Tasaki, K. Liu described the procedure for computing average star vectors. In the current design, all star observations for a single observed star are transformed to a geocentric inertial (GCI) frame, and then averaged. A rotation matrix computed from the attitude quaternions is used for this transformation. In an alternate scheme, the previously averaged star vector is transformed to the current tracker frame using a small angle rotation matrix and averaged with the current star vector. The resulting average vector is finally transformed to

Mr. K. Tasaki
Mr. F. Snow
Page Three

the GCI frame. These two procedures are mathematically equivalent. The second procedure may be slightly faster as compared to the first procedure, but is more complex.

- AADS computes a window size from the current attitude uncertainty (and appropriate multiplication factor) to select candidate stars in the vicinity of the observed star. Thus, the window is coupled to the attitude uncertainty, obviating any need for automatically expanding the window when there are no candidate stars in the window. This condition (no candidate stars) implies that the initial attitude estimate may be in error by a large amount, and it must be corrected by a ground command.

Yours truly,



Dr. K. Saralkar
Task Leader
Attitude Systems Development

KS/stb
Copies:

GSFC

F. McGarry
E. Lefferts
A. Fuchs

CSC

G. Page
S. Waligora
K. Liu
C. Shenitz
G. Klitsch
Y. Frenkel

November 12, 1981

TO: Distribution

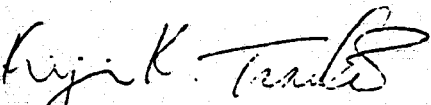
FROM: 582.1/Systems Development and Analysis Branch

SUBJECT: Results of the Critical Design Review (CDR) of Mathematical Modules for the Autonomous Attitude Determination System (AADS)

A critical design review (CDR) of the mathematical modules for the Autonomous Attitude Determination System (AADS) was held on Tuesday, November 3, 1981. The primary purpose of this CDR was to review the detailed software design of the following modules: the gyro data processing, quaternion and state covariance propagation, star tracker data processing, star identification, and state update. The presentation of the review material was shared by the following persons:

Overview	K. Saralkar/CSC/Task 942
Gyro data and propagation	Y. Frenkel/CSC/Task 942
Star identification and update	K. Liu/CSC/Task 942
Star catalog management	Y. Frenkel/CSC/Task 942
Design consideration	K. Saralkar/CSC
Schedule	K. Tasaki/GSFC/582.1

Only one major design problem was uncovered during the CDR. E. Lefferts noted that there should be a separate frequency for the covariance propagation which is different from the state propagation frequency. Since the covariance matrix need not be propagated as frequently as the attitude state, by providing a separate frequency as another input parameter, the computational time for this function can be significantly reduced. As an action item, Task 942 will assess the impact of this change to the current subprocess scheduling logic, as well as to any existing design and/or software by November 16, 1981. All decisions concerning the implementation of this change will be made after the review of the impact assessment.


Kenji K. Tasaki

Distribution:	Mr. Groves/580	Mr. Snow/582.3	Dr. Page/CSC
	Mr. Werking/581.2	Ms. Moe/502	Dr. Saralkar/CSC
	Mr. Fuchs/582	Mr. Nelson/502	Mr. Shenitz/CSC
	Mr. McGarry/582.1	Mr. Smith/560	Mr. Cheuvront/CSC
	Ms. Jun/582.1	Mr. Sanders/SYSTEX, Inc.	Ms. Waligora/CSC
	Mr. Schwenk/582.1	Mr. Klitsch/CSC	Dr. Frenkel/CSC
	Mr. Stark/582.1	Mr. Liu/CSC	
	Mr. Lefferts/582.3		

REFERENCES

1. Computer Sciences Corporation, CSC/SD-81/6096, Autonomous Attitude Determination System (AADS) System Description (Preliminary), K. Saralkar et al., November 1981
2. National Aeronautics and Space Administration, Goddard Space Flight Center, "Requirements Definition for AADS Simulator," (Memorandum), K. Tasaki et al., June 1981
3. --, Goddard Space Flight Center, AADS Simulator System Description (in preparation)
4. --, Goddard Space Flight Center, AADS User's Guide (in preparation)
5. Computer Sciences Corporation, CSC/TM-80/6237, Autonomous Onboard Attitude Determination System Specifications and Requirements, M. D. Shuster et al., December 1980
6. --, CSC/TM-80/6303, Algorithms for Autonomous Star Identification, P. Gambardella, October 1980
7. J. R. Wertz, Editor, Spacecraft Attitude Determination and Control, Dordrecht, Holland: D. Reidel Publishing Company, Astrophysics and Space Science Library, 1978
8. (a) E. J. Lefferts et al., "Kalman Filtering for Spacecraft Attitude Estimation" (paper presented at the AIAA 20th Aerospace Sciences Meeting, Orlando, Florida, January 1982)
(b) E. J. Lefferts, Kalman Filtering for Spacecraft Attitude Estimations Documentation, January 1982
9. Intel Corporation, FORTTRAN 86/88 User's Guide (draft), May 1981
10. Systex Inc., Autonomous Attitude Determination System, Intercomputer Communication Requirements and Functional Design, S. Sanders, October 1981
11. Computer Sciences Corporation, CSC/TM-81/6085, Microprocessor-Based Autonomous Attitude Determination System Design, P. Gambardella et al., March 1981

12. (a) Intel Corporation, Intel 8086 Documentation
(b) Computer Sciences Corporation, "Intel: SBC 86/12 Single-Board (8086/8087) Microcomputer System" (technical memorandum), G. V. Rao, March 1981
13. National Aeronautics and Space Administration, Goddard Space Flight Center, S-700-55, Multimission Modular Spacecraft (MMS) Onboard Computer (OBC) Flight Executive Definition, A. Merwarth, March 1976
14. --, Mission and Data Operations IBM 360 User's Guide, Revision 1, June 1978
15. Digital Equipment Corporation, VAX-11 Documentation, Maynard, Massachusetts
16. Computer Sciences Corporation, CSC/SD-76/6041UD2, SKYMAP System Description: Star Catalog Data Base Generation and Utilization, Revision 2, D. Gottlieb, June 1979
17. Systex, Inc., "AADS 8086 O.S.--System Services," (memorandum), S. Sanders, December 1981
18. Intel Corporation, Intel Microprocessor Development System (MDS) Documentation
19. Computer Sciences Corporation, CSC/SD-82/6019, Landsat-D (LSD) Truth Model Attitude Simulator Algorithm Description and Operating Guide, J. G. Gegner (in preparation)
20. --, CSC/SD-78/6082, Solar Maximum Mission (SMM) Attitude System Functional Specifications and Requirements, R. Byrne et al, September 1982
21. --, CSC/SD-79/6081, Solar Maximum Mission Onboard Attitude Determination and Control Software Description, P. Camillo et al., June 1979

STL BIBLIOGRAPHY

Systems Technology Laboratory, STL-78-0001, System Description of the IMP-16C Microprocessor Orbit Determination Program, C. Shenitz, C. Rabbin, and G. Snyder, October 1978

--, STL-78-002, Landsat/NAVPAC System Description and User's Guide, S. R. Waligora, December 1978

--, STL-79-001, Intel 8080 Orbit Propagation Program System Description and User's Guide, C. Rabbin, April 1979

--, STL-79-002, Evaluation of the IMP-16 Microprocessor Orbit Determination System Filter, C. Shenitz, September 1979

--, STL-79-003, Multimission Modular Spacecraft Ground Support Software System (MMS/GSSS) MODCOMP Device and MAX IV Dependency Study, T. Weldon and M. McClellan, December 1979

--, STL-80-001, Orbit Determination Software Development for Microprocessor-Based Systems: Evaluation and Recommendations, C. M. Shenitz, July 1980

N/S --, STL-80-002, The Two-Way TDRSS Observation Model for the LSI-11/23 Microcomputer, C. E. Goorevich, July 1980

--, STL-80-003, Automated Orbit Determination System (AODS) Requirements Definition and Analysis, S. R. Waligora, C. E. Goorevich, J. Teles, and R. S. Pajerski, September 1980

--, STL-80-004, Algorithms for Autonomous Star Identification, P. Gambardella, October 1980

--, STL-80-005, Autonomous Onboard Attitude Determination System Specifications and Requirements, M. D. Shuster, S. N. Ray, and L. Gunshol, December 1980

--, STL-81-001, Systems Technology Laboratory (STL) Library Methods and Procedures, W. J. Decker and P. D. Merwarth, September 1981

--, STL-81-002, Mathematical Specifications of the Onboard Navigation Package (ONPAC) Simulator (Revision 1), J. B. Dunham, February 1981

--, STL-81-003, Systems Technology Laboratory (STL) Compendium of Utilities, W. J. Decker, E. J. Smith, W. A. Taylor, P. D. Merwarth and M. E. Stark, July 1981

--, STL-81-004, Automated Orbit Determination System (AODS) Environment Simulator for Prototype Testing (ADEPT) System Description, S. R. Waligor, J. E. Fry, Jr., B. J. Prusiewicz, and G. N. Klitsch, August 1981

NIS --, STL-81-005, Preliminary Automated Orbit Determination System (AODS)/AODS Environment Simulator for Prototype Testing (ADEPT) User's Guide, S. R. Waligora, Y. Ong, J. E. Fry, Jr. and B. J. Prusiewicz, September 1981

--, STL-82-001, GPSPAC/Landsat-D Interface (GLI) System Description, J. B. Dunham, H. M. Sielski, and W. T. Wallace, April 1982

--, STL-82-002, GPSPAC/Landsat-D Interface (GLI) System User's Guide, H. M. Sielski, J. B. Dunham, and P. D. Merwarth, March 1982

NIS --, STL-82-003, Autonomous Attitude Determination System (AADS), Volume 1: System Description, K. A. Saralker, Y. G. Frenkel, G. N. Klitsch, K. Liu and E. Lefferts, April 1982