MCR-82-569
Contract No. NAS8-34679

Final
Report

June 1982

Development of an Autonomous Video Rendezvous and Docking System

(NASA-CR-162076) DEVELOPMENT OF AN AUTONOMOUS VIDEO RENDEZVOUS AND DOCKING SYSTEM Final Report (Martin Marietta Aerospace, Denver, Colo.) 284 p HC A13/MF A01 CSCL 12B G3/66

N82-33135

Unclas
28930

MCR-82-569
Contract No. NAS8-34679

Final
Report June 1982

DEVELOPMENT OF AN AUTONOMOUS VIDEO RENDEZVOUS AND DOCKING SYSTEM

John C. Tietz and
Joy H. Kelly

MARTIN MARIETTA AEROSPACE
DENVER AEROSPACE
P.O. Box 179
Denver, Colorado 80201

FOREWORD

This report presents the results of a nine-month study by Martin Marietta for the National Aeronautics and Space Administration's George C. Marshall Spaceflight Center. This study, including conceptual design and simulation of three video guidance systems for spacecraft, was performed under contract NAS8-34679, Development of an Autonomous Video Rendezvous and Docking System. Significant benefits were obtained from previous related work under Martin Marietta IR&D task D-11R.

CONTENTS

		Page
I.	SUMMARY	I-1
II.	INTRODUCTION TO THE PROBLEM	II-1
III.	CONCLUSIONS AND RECOMMENDATIONS	III-1
A.	A Three-Light Docking Aid Is Recommended	III-1
B.	Many Factors Determine Performance	III-2
C.	The Hard Part Is the Last Eight Meters	III-3
D.	Tumbling Targets Make Control Difficult	III-3
E.	A Physical Simulation Should Include a Control System, Camera, and Target	III-4
IV.	THE THREE SYSTEMS THAT WERE SIMULATED	IV-1
A.	Image of Three Lights Gives Attitude and Position in First System	IV-1
B.	Projected Rainbows and Stereo Rangefinding Form Second System	IV-12
C.	Image of Ring of Lights Leaves Roll Undefined in Third System	IV-21
V.	SIMULATION RESULTS AND DISCUSSION	V-1
A.	The Three-Light System Works Best	V-1
B.	The Three-Light System Works with Tumbling Targets	V-5
C.	There Is Only One Option	V-9
D.	The Hardware Technology Is Available	V-11
VI.	TWO OTHER CANDIDATE SYSTEMS	VI-1
A.	Correlation System Was Too Expensive	VI-1
B.	Curve Fitting Scheme Duplicated MSFC Effort	VI-1
VII.	OTHER TECHNIQUES THAT WERE INVESTIGATED	VII-1
A.	Corner/Edge Recognition Is Too Expensive	VII-1
B.	Bar Pattern Requires High Resolution and Good Lighting	VII-2
C.	Rangefinding by Optical Focusing Is Hard to Use	VII-2
VIII.	A GENERIC VIDEO GUIDANCE SYSTEM	VIII-1
A.	The Kalman Filter Improves Accuracy and Derives Rates	VIII-2
B.	The Mathematical Dynamics Model Allows Dead Reckoning	VIII-4
C.	The Inertial Measurement Unit Provides Attitude and Attitude Rate Information	VIII-6
D.	Goal-Setting Logic Provides Intelligence	VIII-7
E.	The Control Law Determines Ideal Thrust Vectors	VIII-9
F.	Thrusters Approximate Required Forces and Torques	VIII-11
G.	Chase Vehicle Dynamics Respond to Thruster Commands	VIII-11
H.	Target Attitude is Modeled As Deterministic	VIII-15
I.	Different Spacecraft Can Be Modeled	VIII-16
IX.	COMPUTER MODELS OF MEASUREMENTS AND NOISE	IX-1
A.	Light Positions Modeled are with Perspective Projection	IX-1
B.	Random Noise Corrupts Image Coordinates	IX-2

APPENDIX

A. COMPUTER PROGRAM TO SIMULATE THREE-LIGHT SYSTEM A-1

B. COMPUTER PROGRAM TO SIMULATE SYSTEM THAT USES "RAINBOW" BEACON AND RANGEFINDER B-1

C. COMPUTER PROGRAM TO SIMULATE SYSTEM THAT USES RING OF LIGHTS . C-1

D. ATTITUDE PARAMETERIZATION D-1

E. ADAPTING THE KALMAN FILTER FOR OTHER SIMULATIONS E-1

Figure

II-1 Chase Vehicle Modeled for Computer Simulations II-2

II-2 Target Spacecraft Modeled for Computer Simulations II-3

III-1 Flashing-Light Docking Aid III-1

III-2 Modification to Ring-of-Lights Docking Aid III-2

IV-1 Flashing-Light Docking Aid IV-2

IV-2 Analog Centroid Computer IV-4

IV-3 Interpretation of r and c IV-8

IV-4 Alternate Attitude Interpretations IV-9

IV-5 Angle Measurement System IV-12

IV-6 Stereo Rangefinder on the Chase Vehicle IV-13

IV-7 Stereo Rangefinder Block Diagram IV-14

IV-8 The Stereo Rangefinding System and Nomenclature IV-15

IV-9 Percent Error vs Range for Stereo Rangefinding System IV-17

IV-10 Ring-of-Lights Docking Aid IV-22

IV-11 Possible Hardware to Compute Centroid and Ellipse Parameters . IV-23

IV-12 Equations Solved by Circuit of Figure IV-11 . . . . IV-23

IV-13 Two Ellipse Interpretations IV-25

V-1 Typical Trajectories with Three-Light System V-2

V-2 Typical Trajectories with Rainbow System V-3

V-3 Typical Trajectories with Ellipse System V-4

V-4 Typical Trajectories with Three-Light System with Various Target Tumble Rates about the Docking Axis V-6

V-5 Trajectories with Three-Light System with Various Tumble Rates about the Pitch Axis V-7

V-6 Trajectories with Three-Light System with Various Tumble Rates about the Yaw Axis V-8

VIII-1 Generic Video Rendezvous Guidance Systems VIII-2

VIII-2 Physical Significance of Equation (VIII-10) VIII-8

VIII-3 Esstential Logic of Function ACCEL VIII-10

IX-1 Physical Interpretation of Equation (IX-1) IX-1

Table

II-1 Chase Vehicle Characteristics II-2

IV-1 Derivations of Formulas (IV-18) and (IV-19) IV-16

V-1 Tradeoff Matrix for Three Rendezvous and Docking Systems . . . V-10

VII-1 Techniques Investigated VII-1

VIII-1 Relationship between Block Diagram Blocks and Subroutines in the Simulation Programs VIII-1

VIII-2 Definitions of Elements of True-State Array STATE VIII-12

VIII-3 Major Equations for True-State Computation VIII-13

VIII-4 Definition of Variables in Table VIII-3 . . . . VIII-14

I. Summary

ERRATA

Please note the following errors in Development of An Autonomous Video Rendezvous and Docking System:

Page	Line	Original Text	Correction
iii	2nd from bottom	Light Positions Modeled are...	Light Positions are Modeled...
I-1	6th line of 2nd paragraph	...20,000 degrees per second...	...20,000 degrees per hour...
IV-11	3rd line from bottom	...1/600th of the...	...1/400th of the...
IV-11	2nd line from bottom	...380 television lines...	...128 television lines...
IV-12	1st and 2nd lines	...better resolution, and resolution as...	...better resolution. [delete remainder of sentence]
IV-16	1st line of figure	...diagram in Figure IV -9.	...diagram in Figure IV -8.
IV-21	3rd line of 1st paragraph	...system. The same resolution was modeled, and the same assumptions were made.	...system, and the same assumptions were made, but 380-line resolution was modeled.
V-5	5th line of last paragraph	...degrees per hour above...	...degrees per hour about...
V-10	3-Light system line of Table, Resolution column	380 Television Lines	128 Television Lines
VIII-7	After definition of A_c	[Text missing before paragraph that begins, "In these simulations...]	$\underline{\omega}_b$, represented in the program by the 3-element array BODVEL, is the chase vehicle angular velocity, expressed in the body coordinate system.
VIII-15	Row defining $\underline{q}$ and $\dot{\underline{q}}$, "Meaning" column	Allitude Quaternion	Attitude Quaternion

I. SUMMARY

A simple docking aid consisting of three flashing lights has proved viable for use in an autonomous video rendezvous and docking system for spacecraft. A television image of this target can be analyzed to determine the relative positions and attitudes of the two spacecraft. The analysis time is only 100 milliseconds because a simple dedicated electronic circuit assists in the analysis.

Control systems using this and two other types of docking aids were evaluated through computer simulation in this study and other approaches were considered. However, this three-light system performed much better than the others. Its accuracy is affected little by tumbling of the target spacecraft, and in the simulations it was able to cope with attitude rates up to 20,000 degrees per hour) about the docking axis. Its performance with rotation about other axes is determined primarily by the state-estimation and goal-setting portions of the control system, not by measurement accuracy.

A physical simulation of the three-light control system would be useful to verify the validity of the assumptions and mathematical models used in this study. The simulation should employ scaled target models, a television camera, and hardware video signal processing electronics. The simulation should also model the spacecraft control system and the limitations of a practical spacecraft. This level of detail is recommended to force consideration of compatibility problems and may reveal weaknesses in the approach that might not otherwise be detected. This report includes a discussion of a suitable control system, and Appendix A discusses a computer program that can serve as the basis for the physical simulation.

II. Introduction to the Problem

II. INTRODUCTION TO THE PROBLEM

The need for an Automated Rendezvous and Docking System has been shown in a number of mission models, including those in the Mars sample return and large space system studies. Several factors make automation attractive: practical limitations on human interaction, fuel-use and trajectory optimization requirements, safety, communication limitations, and the need for real-time operation.

The purpose of this study was to identify video techniques that might be suitable for such an automated system, define the equations and algorithms these techniques would use, and evaluate video guidance control systems based on these techniques through computer simulation.

To ensure that practical problems were considered, the simulation was to model not only the sensor but also methods for dealing with a number of practical problems, e.g., maintaining control when the target spacecraft leaves the field of view of the guidance sensor. The simulation also was to model the characteristics and limitations of practical spacecraft, because this may reveal subtle incompatibilities. A mission model was then defined to serve as a basis for the simulation.

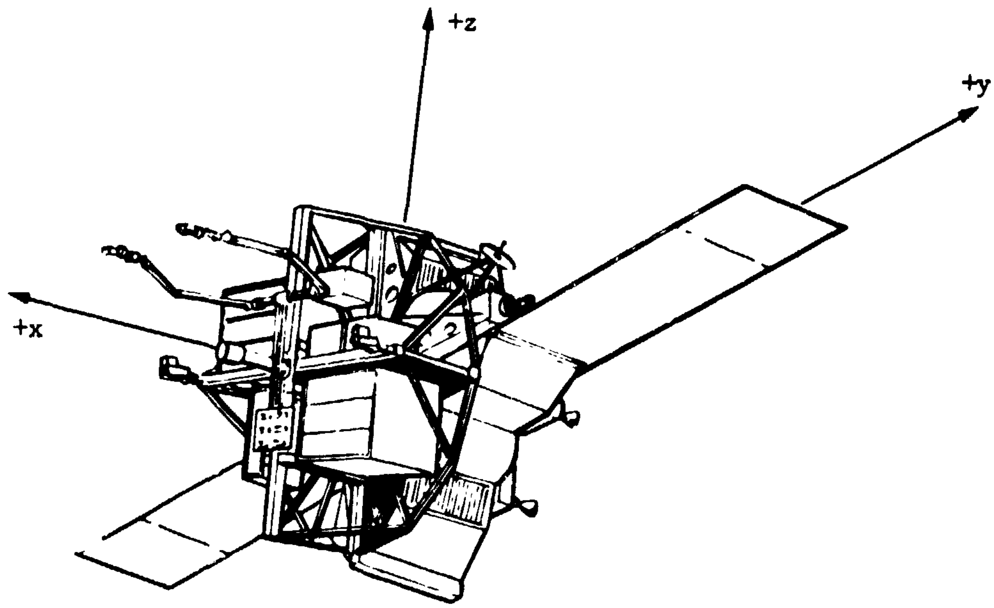
In this model the chase vehicle is a general-purpose spacecraft for repair, refurbishment, and retrieval of other spacecraft. After it is deployed from the Space Shuttle, it must rendezvous and dock with a passive target spacecraft that is in a circular orbit about the earth at an altitude of 300 km. The chase vehicle is illustrated in Figure II-1 and has the characteristics summarized in Table II-1.

The target spacecraft (Fig. II-2), is similar to the Long Duration Exposure Facility (LDEF). It has an appropriate docking aid and docking fixture, but it is passive during the docking operation. It can neither perform cooperative maneuvers nor maintain a stable attitude. Its only means of cooperation is turning docking-aid lamps on and off in response to radio commands.

Each computer simulation was to begin when the coarse rendezvous system, which has gotten the chase vehicle to within 1000 feet of the target, hands control to the video system. The coarse rendezvous is assumed accurate enough to guarantee that the target is within the field of view of the video guidance sensor. (Initial target acquisition is considered in the study but is not modeled in the computer simulations.) It is also assumed that, although fuel represents a significant fraction of the vehicle's mass, the mass during the terminal rendezvous can be predicted with reasonable accuracy. Because fuel use can be measured, this assumption is not unreasonable.

The scope of the study did not include system optimization, trajectory planning, or detailed hardware design.

ORIGINAL PAGE IS
OF POOR QUALITY



The Body Coordinate System Used in the Simulations Are Shown

Figure II-1 Chase Vehicle Modeled for Computer Simulations

Table II-1 Chase Vehicle Characteristics

Vehicle Size: Length: 5m; Width: 13m; Height: 4.5m		
Vehicle Mass: 3700 kg (Full Fuel Tank); 1800 kg (Empty Fuel Tank)		
Fuel: Monopropellant Hydrazine with GN_2 Blowdown		
Approximate Translational Acceleration Authority (Each Axis): 0.1 m/s^2*		
Approximate Angular Acceleration Authority (Each Axis): 0.037 rad/s^2*		
Total Impulse: 4.9×10^6 N·s*		
Moments of Inertia ($kg \cdot m^2$):		
	Full Fuel Tank	Empty Fuel Tank
Roll	4240	1910
Pitch	5110	2300
Yaw	5030	2260

* These Characteristics Vary with Fuel Load

ORIGINAL PAGE IS
OF POOR QUALITY

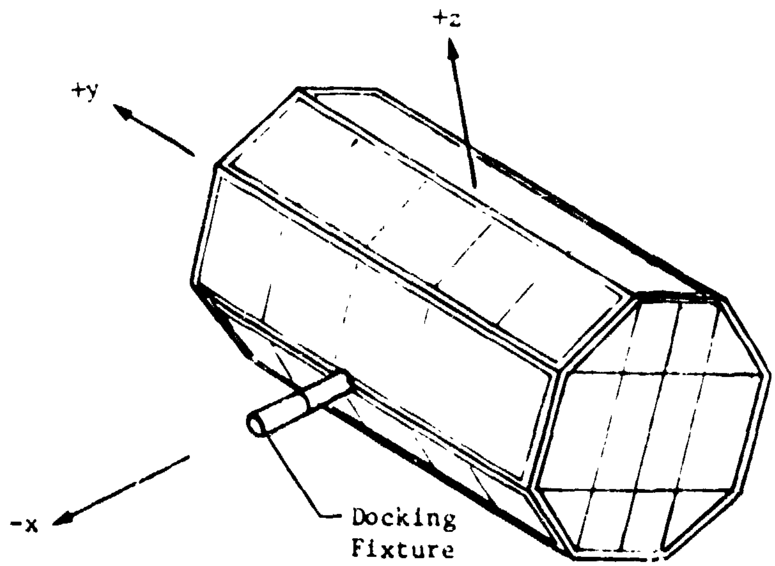


Figure II-2 Target Spacecraft Modeled for Computer Simulations
The Body Coordinate System Used in the Simulations Is Shown. Note the Negative X Axis Is Shown: Chase Vehicle and Target X Axes Are Parallel When the Two Spacecraft Are Docked.

III. Conclusions and Recommendations

III. Conclusions and Recommendations

III. CONCLUSIONS AND RECOMMENDATIONS

A. A THREE-LIGHT DOCKING AID IS RECOMMENDED

Of the three video guidance systems simulated in this study, the best system used the three-light docking aid shown in Figure III-1. Not only was it the simplest of the three systems, it was also the only system that worked reliably in the simulations.

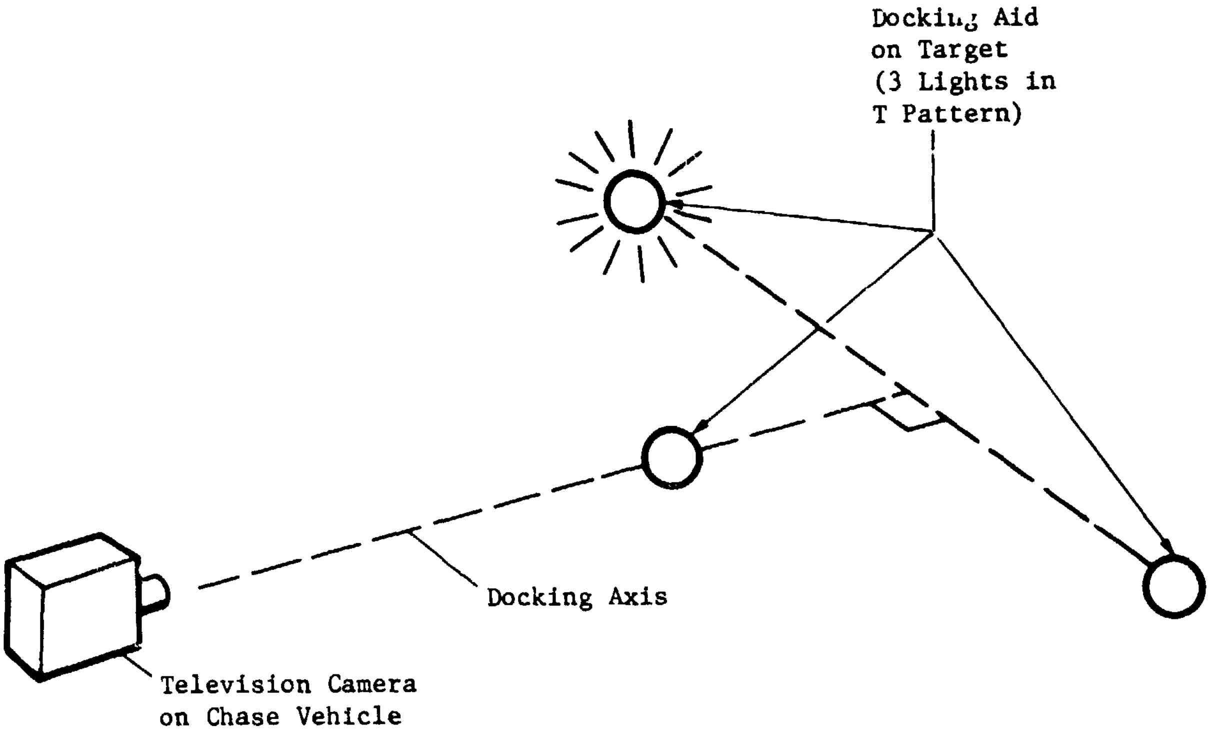


Figure III-1 Flashing-Light Docking Aid

It was not accuracy that made this system best; it consistently outperformed the others even though the resolution and target size that were simulated gave it the poorest accuracy at long range. Its advantage was that it could measure both relative position and relative attitude from a single observation. The other systems had to resolve ambiguities in attitude by combining two successive measurements, an approach the simulation results proved to be dangerous.

The problem with the other two systems could be solved by modifying the docking aids slightly. For example, one of these systems used a ring of lights on the target spacecraft for a docking aid. By observing the target, the guidance system can determine that the camera lies somewhere on a circle about the docking axis, but it cannot determine the camera's position on the circle. The addition of one light (Fig. III-2) would greatly improve the performance of the guidance system by uniquely defining the direction to the docking axis. Unfortunately, this approach results in a system that is more complicated than the three-light system because the modification requires adding a way of distinguishing between the ring of lights and the added light. Further, target roll about the docking axis still cannot be observed with this system.

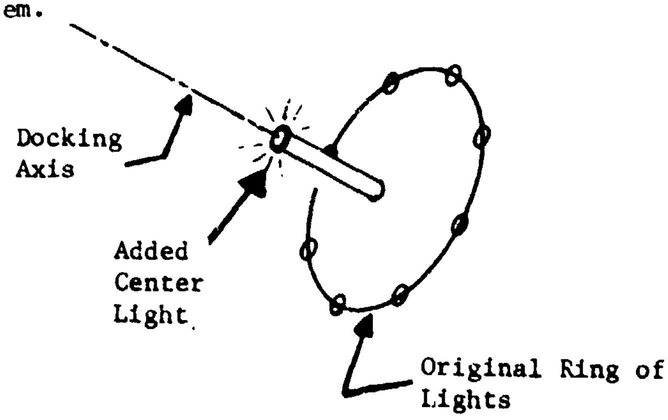


Figure III-2 Modification to Ring-of-Lights Docking Aid Target Roll about the Docking Axis Is Still Not Observable, but the Direction to the Docking Axis Is Uniquely Defined.

The other system simulated used a docking aid that projects a rainbow of light in each of two directions. Color sensing and stereo rangefinding are used in this system to uniquely define the chase vehicle's position. Relative attitude, however, is ambiguous because chase vehicle rotation about the line of sight is not observable. The problem could be solved by adding some means for sensing this rotation, but this system is already the most complex of the three.

B. MANY FACTORS DETERMINE PERFORMANCE

The video guidance system must do much more than determine the current position and attitude. It must also turn chase vehicle thrusters on and off to steer a safe course from the current position to the target spacecraft. It must ensure that the chase vehicle arrives at the target with the proper attitude and that the two spacecraft contact at low velocity. It must also align the chase vehicle's docking fixture properly with the docking fixture on the target spacecraft. In the process of doing these things, it must not allow the target's image to leave the field of view of its television camera for more than a moment.

A control system may perform these functions poorly despite accurate knowledge of relative position and attitude. In this study we tried to uncover subtle incompatibilities between video guidance techniques and the remainder of the control system by modeling a complete system. If we had not, problems with the "rainbow" and "ring-of-light" approaches might have gone undiscovered. The problem in evaluating the systems this way is that performance depends on the entire system and not just the video guidance technique. If the system as a whole does not make effective use of the video information, there is a danger that a potentially useful technique will be rejected.

For example, we found that a small change in the goal-setting logic of the control system made a great difference in the ability of the three-light system to cope with tumbling targets. Suitable changes in the other simulations might have made the "rainbow" and "ring-of-lights" approaches look more attractive.

While we do not believe that such changes would have altered the conclusions of the study (the three-light system is still simplest and least sensitive to the rest of the system), we believe that "fine-tuning" of the control system for best performance is worth the effort. We also want to caution the reader that the performance reported here is not necessarily the best the system might be able to provide.

C. THE HARD PART IS THE LAST EIGHT METERS

A system that measures only the distance and direction to the target is adequate to approach within eight meters of the target. At this point attitude information becomes vital because offsets among the docking aid, target docking fixture, and target center of mass become major contributors to alignment errors. The offsets among the camera, chase vehicle center of mass, and chase vehicle docking fixture make attitude information doubly important because chase vehicle attitude and position must be controlled. To further complicate the problem, the target may be coning and nutating, making it difficult to anticipate attitude changes.

All three systems performed well at great distances from the target, and all three had problems at close range. The ring-of-lights system was totally unable to cope with close range because the relative velocity between the chase vehicle and target is small at close range. This system depends on differences between successive observations to measure attitude. With the velocity reduced, it was using observations in which the differences were almost entirely due to random effects.

The "rainbow" system received its best attitude information at close range because small movements produced greater changes in orientation of the line of sight. But during the last few critical seconds of the operation, it loses range information because the rangefinder cameras cannot see the docking aid.

Although the three-light system maintained good accuracy at close range, it still had problems. Small translational movements between light flashes made it more difficult to keep the entire target within the camera's field of view. Target rotation made the problems more severe. Also, because the equations this system uses for image interpretation are based on orthographic projection, their accuracy became poorer as range decreased: this projection approximates the image's appearance well only when the range is much greater than the distance between the lights on the docking aid.

Some of the close-range problems might be handled by using two different docking aids. A large docking aid could be used at long range, and a smaller one with the same geometry could be used for close-in operations. Even if this is done, close range will remain the most difficult problem.

D. TUMBLING TARGETS MAKE CONTROL DIFFICULT

Effective operations with tumbling and coning targets will require sophisticated state estimators and goal-setting logic. The normal tumble rate of the target in the mission model for this study is one revolution per orbit, or 240 degrees per hour. Although the control system used in the simulations was able to cope with roll rates of over 20,000 degrees per hour about the docking axis, the system had trouble in keeping the docking aid within the field of view and in docking with the proper alignment with rates as small as 1000 degrees per hour in pitch or yaw. At rates of about 4000 degrees per hour in pitch or yaw, docking was extremely difficult. The problem is not measurement accuracy but control. The structure of the control system presented in Chapter VIII is adequate for dealing with rotating targets, but the simplified version used in the simulations is inadequate for high tumble rates.

Some fundamental constraints must be kept in mind in refining the control system. First, the chase vehicle's thrust authority is limited. If the control system tries to minimize stresses on the docking fixtures by staying on the docking axis at all times, the thrusters may not be able to overcome centrifugal force. If the control system tries to minimize problems with centrifugal force, it can plan a trajectory that takes the chase vehicle directly to the point where the target's docking fixture is predicted to be on arrival. However, this approach may greatly stress the docking fixtures of both spacecraft. Further, a slight error in estimating tumble parameters may cause the docking fixtures to miss each other. Solving problems like these was beyond the scope of the study reported here. We recommend a separate study to investigate possible solutions.

E. A PHYSICAL SIMULATION SHOULD INCLUDE A CONTROL SYSTEM, CAMERA, AND TARGET

By modeling an entire six-degree-of-freedom control system, this study uncovered problems that might have gone unnoticed if only measurements had been modeled. We highly recommend the same approach in a physical simulation. This will not be difficult to implement because the computer programs in Appendices A through C already include "dummy" subroutines for control of a physical simulator.

The simulator should allow positioning of a television camera anywhere in a large room under computer control, and have a gimbal set to allow pointing of the camera. It is not necessary to rotate the target model because only relative position and attitude are detectable in the television image. Tumbling of any complexity can be simulated, and it is not necessary to make mechanical changes to switch between simple tumbling about the docking axis and a complex coning and nutating motion. However, the simulated rates must be small enough so only one side of the target need be visible.

Simulator speed is entirely a matter of convenience. Although computer control will be required to allow simulations to be run in a reasonable amount of time, little is gained by running the simulation in exactly the amount of time required for an actual operation in space. Although full-speed operation would show the effects of image smear and after images, implementing the system with a solid-state camera and flash lamps would make these factors insignificant.

To simulate the full dynamic range of 300 meters to contact, it will probably be necessary to build two or three different target models. One of these models would be one or two percent of full scale and would simulate the appearance of the target at long range. Another would be much larger, perhaps full scale, and would show the detail visible at close range. Depending on the dimensions of the room and the accuracy of the simulator, it may be desirable to use a third model for intermediate ranges.

The results of this study show that a camera with resolution comparable to standard television will be suitable. A solid-state camera is attractive for space because it does not require high voltage and a fragile high-vacuum camera tube. We therefore recommend the use of a solid-state camera in the physical simulation. This will help to uncover subtle problems related to camera problems such as blooming, image granularity and resolution, lens adjustments, dead picture elements, and variations among picture elements in dark signal and sensitivity.

The computer used in the simulations should be equipped with interfaces to drive the simulator and to receive information from the television camera. If flashing lights are used, it will also require an interface to synchronize light flashes with the computer's calculations. Any video signal processing that would be done in hardware in a real spacecraft should be done in similar hardware in the simulation to evaluate hardware design problems and shortcomings. Aside from these requirements, the computer does not have to be particularly large or fast. The program could, in fact, fit into many personal computers.

IV. The Three Systems That Were Simulated

IV. The Three Systems That Were Simulated

IV. THE THREE SYSTEMS THAT WERE SIMULATED

Three separate video guidance systems were simulated in this study. This chapter describes the technical details, computer models and key mathematical equations used in each system to derive relative position and attitude.

One of the most formidable challenges in autonomous video guidance is recognizing and analyzing the target. The image analysis must be done quickly, because a long computation delay degrades the performance of the control system. The analysis must also be accurate, because position data will be used to determine velocity and perhaps other parameters. For example, if the target is tumbling rapidly, the control system may have to estimate parameters for predicting what the target's attitude, attitude rate, and rotation axis will be when the two spacecraft touch.

Whether this is a job for a mainframe computer or an inconspicuous electronics package depends largely on the design of a docking aid on the target spacecraft. In all of the simulated systems, the docking aids were designed to be analyzed with a minimum of onboard intelligence. Each system uses a hardware analyzer that reduces the video data to a small set of parameters. This approach gives these systems high speed without placing a heavy burden on the flight computer.

A. IMAGE OF THREE LIGHTS GIVES ATTITUDE AND POSITION IN FIRST SYSTEM

1. Description of Technique

The first system, developed in 1980 at Martin Marietta under IR&D task D-11R, is the simplest of the three systems. It uses a docking aid that consists of the three lights (Fig. IV-1). By using a simple dedicated electronic circuit and a small routine in the flight computer, this system can analyze the docking aid in approximately 100 milliseconds. The speed and simplicity are achieved by giving the analyzer a very simple job: finding the coordinates of a spot of light in a television image.

The simplification results from the fact that only one lamp is on at a time. Bright flash lamps are used so that the lamps are by far the brightest objects in the television image. The dedicated electronic analyzer can therefore analyze the image by simply finding the coordinates of the center of brightness of the image as a whole.

There are several ways to make individual lamps stand out without flashing the lamps in sequence. Colored lights and polarization, for example, can be used to give each lamp a unique "signature."

ORIGINAL PAGE IS
OF POOR QUALITY

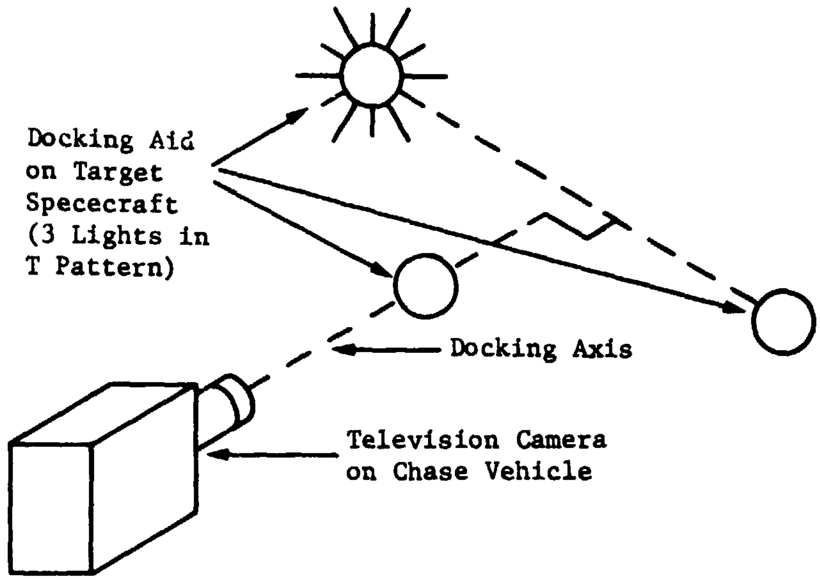


Figure IV-1 Flashing-Light Docking Aid
Three Sequentially Flashed Lamps Provide Full Relative Attitude and Position Information with a Minimum of Computation. In the Basic Configuration, Lamp Flashes are Directly Controlled by the Chase Vehicle via a Radio Command Link, But Variations Not Requiring a Command Link Are Discussed in the Text.

However, sequentially-flashed lamps have three significant advantages:

1) The system can very easily determine which lamp is which.

2) The flashes can be synchronized to a shutter on the camera so that light from other sources can get to the camera only during the flash. This reduces the effective brightness of background "clutter" by a factor of 30 but does not change the effective brightness of the lamp. Further reduction in background lighting can be realized by comparing an image showing the lamp flashing with an image showing the lamp off.

3) The flash duration is short, about a millisecond. This is short enough to fit into the television camera's retrace interval so that there is no question about whether the flash occurred before or after the corresponding picture elements were scanned. Flashing during the retrace also allows the analyzer to process a different lamp on each video frame, greatly speeding the analysis. If an integrating sensor is used (a charge coupled device or vidicon), the electron image left by the flash will remain in the sensor until the image is scanned.

When the three lamps are flashed in sequence, high attitude rates may cause image distortion. This distortion can be partially corrected by flashing one of the lamps twice, noting the change in the location of the center of brightness, and adjusting the other two light locations accordingly.

2. Camera and Video Processing

The center of brightness or "centroid" of an image is defined by the coordinates:

$$X_c = k \int_{frame} V(t)X(t)dt / \int_{frame} V(t)dt \qquad (IV-1)$$

and

$$Y_c = k \int_{frame} V(t)Y(t)dt / \int_{frame} V(t)dt \qquad (IV-2)$$

where:

k is a constant that depends on scan rate and amplitude;

V is the video signal with blanking and synchronizing pulses removed;

X and Y are the horizontal and vertical deflection, varying from a negative value at the left side or bottom of the image to a positive value of equal magnitude at the right side or top of the image;

t is time.

These coordinates can be computed with analog electronic components such as multipliers and integrators, or they can be approximated with digital components such as counters, adders, and accumulators. Multiplication of the video signal by the deflection signal is reduced to a gating operation if the video signal is converted to a two-level digital signal with a comparator, but careful selection of the comparator threshold is required to ensure that the lights and background are reliably separated by this technique.

The choice of technology for implementing equations (IV-1) and (IV-2) will depend on a number of factors including environmental factors, reliability requirements, and camera type. In general, it will be convenient to uses an analog approach with vidicons and other cameras that use an analog deflection signal. A digital approach may be more convenient for systems using self-scanned solid-state arrays. Figure IV-2 illustrates the analog approach.

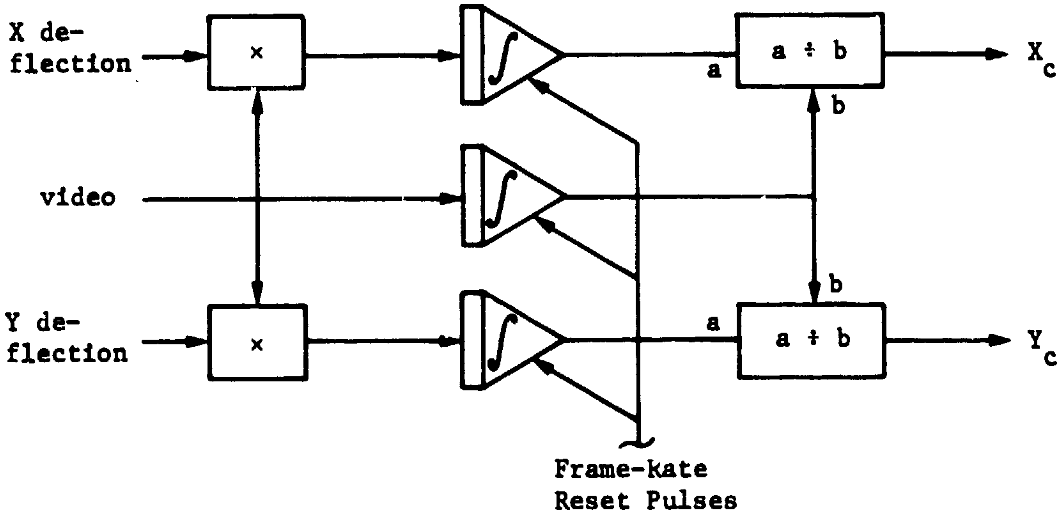


Figure IV-2 Analog Centroid Computer
The Centroid Coordinates X_c and Y_c Are Valid at the End of a Frame.
The Boxes Labeled "a÷b" Are Analog Dividers.

Besides computing the image centroid, the video preprocessing circuit is required to synchronize the scanning of the camera with the flashing of the lights. This can be done quite easily if the circuit directly commands the lights by a low-power radio link. This approach has a number of advantages:

1) It is a simple matter to determine which of the three lamps is flashing if each is commanded individually. (The lamps must be uniquely identified for proper image interpretation);

2) Operation with multiple targets in the same area is simplified, as each can be assigned a unique code or radio channel;

3) The lights will flash only when they are needed;

4) Acquisition of the target is simplified;

5) Very little power is required, since the transmitter's operating range is only 300 meters.

An alternate approach to synchronization, phase locking to independently flashing lights, may reduce the cost of the total system when one (or a few) chase vehicles are used for many missions. In such a system, the target cost and complexity are reduced at the expense of a more complicated chase vehicle system.

When target simplicity is most important, the flash-lamp approach will be less attractive. The same pattern of three reference points on the target can be used in a much simpler docking aid, but with retroreflectors and colored filters replacing the flash lamps. A color television camera is then used to distinguish among the three points, and a flashing light on the chase vehicle supplies illumination. If lighting is favorable, cost could be further reduced by using painted spots for the reference points. In either case, the reference points are distinguishable by the ratios among the red, blue, and green camera outputs. The color-camera approach allows the video preprocessor to find all three centroids from a single video frame, so image distortion from vehicle motion will not need correction.

After finding the coordinates of the three lamps in the image, the system determines the position of the chase vehicle in the instantaneous target reference frame. The calculations the flight computer uses to do this are found in subroutine POSIT of the program listing in Appendix A. The calculations are derived from the positions of the lights in an orthographic projection of the light pattern. This projection is a good approximation to the appearance of the image when the distance from the camera to the lights is many times greater than the distances between lights, and although perspective effects become significant at close range, the equations still properly indicate the direction to the docking axis of the target. The perspective effects do not cause serious problems if the camera and docking aid are set back from the docking fixtures so that the camera is a few meters from the lamps when the two spacecraft are docked.

Orthographic projection was used in deriving the equations to minimize the amount of calculation. A more accurate solution could be developed to consider perspective effects, or a correction algorithm could be used to refine the position estimate provided by the simpler equations presented here. Unless the camera must be mounted at the very front of the chase vehicle, such improvements produce little measurable benefit.

3. Equations Used

There are two aspects to attitude control with this system: the camera must remain pointed at the docking aid, and the chase vehicle must determine the target spacecraft's attitude and match it to dock properly.

Camera pointing is the easiest. Yaw and pitch errors are simple trigonometric functions of the coordinates of the center light's image on the television screen:

$$\theta_{pitch} = \tan^{-1}(v_2/f) \qquad \text{(IV-3)}$$

$$\theta_{yaw} = \tan^{-1}(u_2/f) \qquad \text{(IV-4)}$$

where (u_i, v_i) are the coordinates of the ith lamp's image on the camera-focal plane and f is the lens focal length. The roll error can be approximated by the angle between horizontal and the line connecting the images of the two side lamps:

$$\theta_{roll} = \text{atan2}(v_3 - v_1, u_3 - u_1) \qquad \text{(IV-5)}$$

where atan2 is the FORTRAN two-argument arctangent function. This formula is accurate only when the camera is in line with the x axis of the docking aid, but since roll errors are usually unimportant until the instant of docking, and the guidance system forces the chase vehicle toward this axis, the formula works very well in practice. Subroutine RPY of the simulation program uses these formulas to correct attitude after each light-flashing sequence, as long as the lights are in the camera's field of view. Alternate formulas that use the "state estimate" from an onboard mathematical dynamics model are used when the light cannot be seen. These formulas are found in subroutine ESTRPY, a complete discussion of which will be found in Chapter VIII.

The more complicated problem of determining target spacecraft attitude is solved by comparing the camera's position in two different coordinate systems. The first system is a reference frame centered at the center docking-aid light and parallel to the target-body frame (Fig. II-2). This may be called the "docking-aid" frame. The second coordinate system is the so-called "primary" reference frame used for navigation. The latter frame is a nonrotating frame that is centered at the center of mass of the target spacecraft and is parallel to the chase vehicle body axes at the instant the video guidance system takes control. (See Chapter VIII for a more complete discussion of this coordinate system.) In the simulation program, the calculations for deriving target attitude are done in subroutine ATITUD, which calls the lower-level subroutines FINDCV, QUATRN, and DIRMAT.

Subroutine FINDCV computes the camera's position in a coordinate system parallel to the primary coordinate system but centered at the center light. The equation it uses is:

$$\underline{c} = A_c^T \begin{bmatrix} -f \\ u_2 \\ -v_2 \end{bmatrix} \rho / \sqrt{f^2 + u_2^2 + v_2^2} \qquad \text{(IV-6)}$$

in which:

$\underline{c}$, represented in the program as CVPOS, is the required camera position vector.

ρ, represented in the program as RHO, is the measured range from the camera to the docking aid. This value is provided by subroutine POSIT, which was previously discussed.

u_2, v_2, represented in the program as UC and VC, are the coordinates of the center light's image at the focal plane of the camera as in equations (IV-3) through (IV-5).

A^T, represented in the programs as ACVT, is the transpose of the chase vehicle's attitude direction cosine matrix, which is obtained from the inertial measurement unit.

f, represented in the program as FOCLEN, is the lens focal length.

ORIGINAL PAGE IS
OF POOR QUALITY

To compute the target's attitude, the guidance system uses $\underline{c}$ and another vector, $\underline{r}$, which defines the camera's position in the "docking aid" reference frame. In the program, $\underline{r}$ is represented by the array RELPOS and is computed from the image appearance in subroutine POSIT. Subroutine QUATRN computes the target attitude quaternion with the following formulas:

$$\underline{w} = \underline{r} \times \underline{c} \quad \text{(cross product)} \tag{IV-7}$$

$$\phi = \tan^{-1}\left(\frac{|\underline{w}|}{\underline{r}\cdot\underline{c}}\right) \tag{IV-8}$$

$$\underline{q}' = \begin{vmatrix} \underline{w}\,\sin(\phi/2)/|\underline{w}| \\ \cos(\phi/2) \end{vmatrix} \tag{IV-9}$$

$$\underline{S} = A_c \begin{bmatrix} -2(q_1' q_2' - q_3' q_4') \\ q_1'^2 - q_2'^2 + q_3'^2 - q_4'^2 \\ -2(q_2' q_3' + q_1' q_4') \end{bmatrix} \tag{IV-10}$$

$$\theta = \tan^{-1}(S_3/S_2) - \tan^{-1}\left(\frac{v_3 - v_1}{u_1 - u_3}\right) \tag{IV-11}$$

$$\underline{q}'' = \begin{vmatrix} \underline{r}\,\sin(\theta/2)/|\underline{r}| \\ \cos(\theta/2) \end{vmatrix} \tag{IV-12}$$

$$\underline{q} = \underline{q}'\,\underline{q}'' \quad \text{(quaternion product--see Appendix D)} \tag{IV-13}$$

in which:

$\underline{q}$, represented in the program as QT, is the required quaternion,

$\underline{r}$ and $\underline{c}$ are the camera position vectors defined previously,

u_i, v_i are the coordinates of the image of lamp i at the camera focal plane, as before,

A_c, represented in the program as ACV, is the chase vehicle attitude direction cosine matrix from the inertial measurement unit,

The remainder of the variables are intermediate results.

The physical interpretation of this procedure is as follows. The vectors $\underline{r}$ and $\underline{c}$ both represent the line of sight, or the camera's position with respect to the docking aid, but they are expressed in two different coordinate systems that have a common origin. The onboard computer knows the orientation of one of those coordinate systems, because that

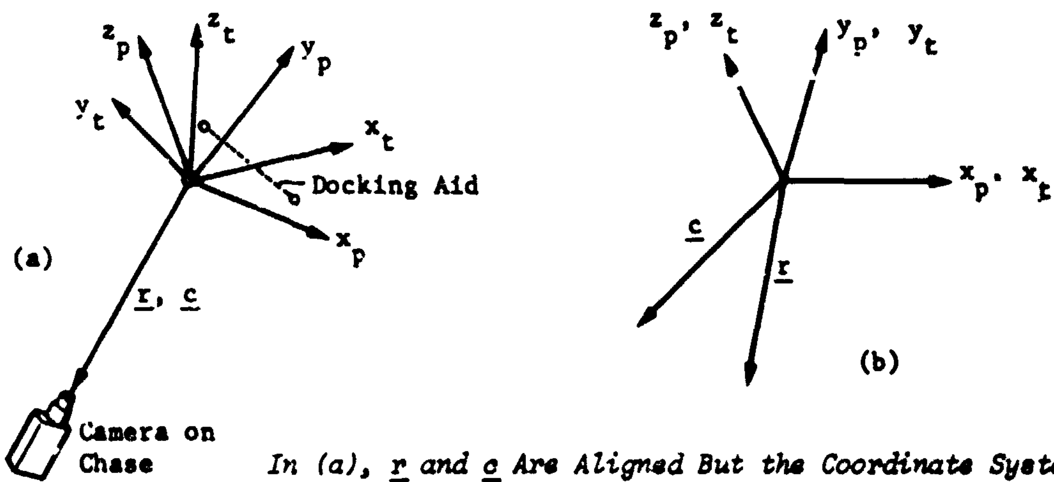
ORIGINAL PAGE IS
OF POOR QUALITY

system is the reference frame for its inertial measurement unit. The other coordinate system, which is parallel to the target's body axes, is unknown.

The computational strategy is to initially assume that the two coordinate systems are identical. If they are, r and c should have identical numerical values, and the observations should agree with predictions based on projective geometry. The assumption is probably wrong, so they probably won't.

The computer's task is to find a rotation that will give predicted observations that agree with what the camera sees.

Consider what happens if the docking aid coordinate system rotates and r retains its orientation with respect to that system (r rotates with the coordinate system). If the docking-aid coordinate system is rotated to align with the primary frame, r and c will point in different directions. This is where the computer starts: assuming that the coordinates systems are aligned and noting that r and c have different numerical values (Fig. IV-3).



In (a), r and c Are Aligned But the Coordinate Systems Are Not. This Represent Physical Reality. In (b), the Coordinate Systems Have Been Aligned. This Misaligns r and c.

Figure IV-3 Interpretation of r and c

Immediately the computer senses that something is wrong. The vectors r and c don't line up, so the first step is to find a rotation of the docking-aid coordinate system that will line them up.

The smallest rotation that will align r and c is a rotation about the axis defined by the cross product r x c (equation [IV-7]). The magnitude of the cross product is $|r|\,|c| \sin\phi$, where ϕ is the required rotation angle. Similarly, the dot product $r \cdot c$ is $|r|\,|c| \cos\phi$. The ratio of these quantities is

$$\frac{|\underline{r}||\underline{c}| \sin \phi}{|\underline{r}||\underline{c}| \cos \phi} = \tan \phi, \qquad \text{(IV-14)}$$

which allows the computer to determine ϕ (equation [IV-8]).

An axis of rotation and a rotation angle define a quaternion (Appendix D), and this quaternion, $\underline{q}'$, is computed with equation (IV-9).

This quaternion only partially specifies target attitude; any rotation of the docking-aid frame about $\underline{r}$ leaves $\underline{r}$ and $\underline{c}$ aligned, so there are an infinite number of target attitudes to select from, as Figure IV-4 illustrates. Some other method must now be found to find the proper rotation about the line of sight. The approach here is to use the appearance of lamps 1 and 3.

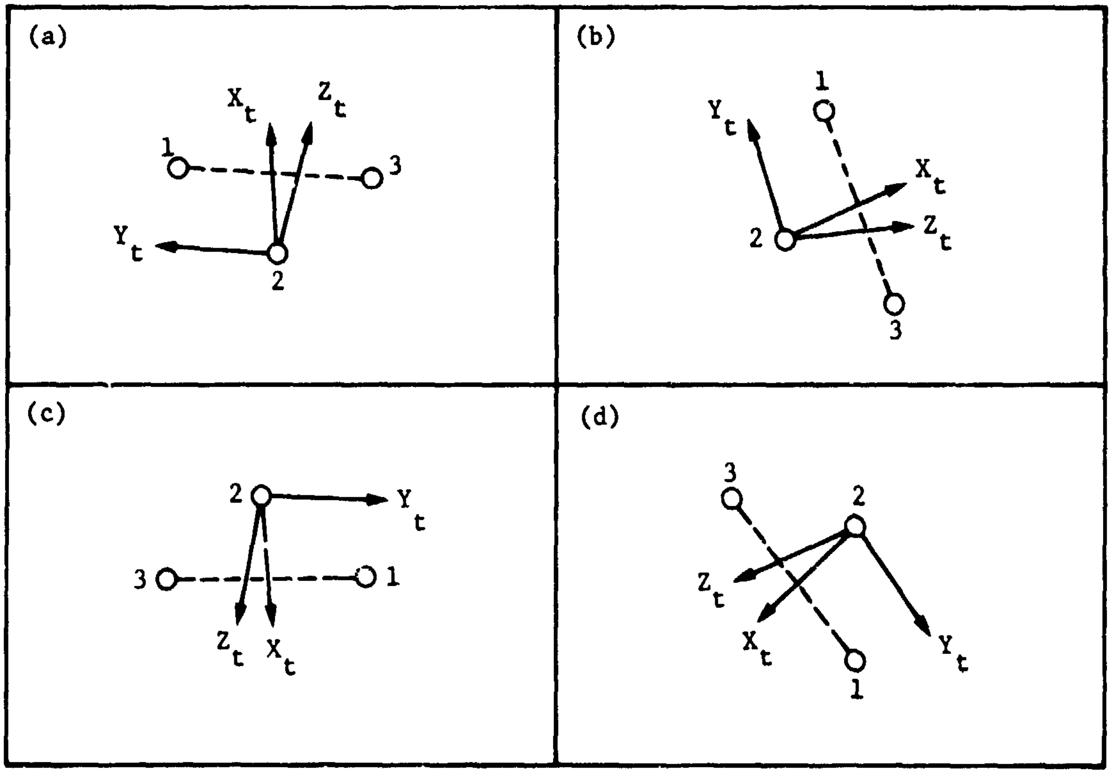


Figure IV-4 Alternate Attitude Interpretations
An Infinite Number of Target Attitudes Align $\underline{r}$ *and* $\underline{c}$*.*
Four Possibilities Are Shown in (a) through (d).

A vector from lamp 3 to lamp 1 lies parallel to the docking aid y axis and points in the -y direction. This vector's orientation in the television image can be measured:

$$\theta_1 = \tan^{-1}\left(\frac{v_3 - v_1}{-(u_3 - u_1)}\right) \qquad \text{(IV-15)}$$

(The minus sign in the argument's denominator shows that the positive horizontal axis of the image is aligned with the negative y axis of the chase vehicle.)

The vector's orientation can also be predicted. Suppose that $\underline{q}'$ represents the target attitude. If A_t is the direction cosine matrix that corresponds to $\underline{q}'$,

$$\underline{s} = A_c A_t^T \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix} \qquad \text{(IV-16)}$$

is a unit vector in the direction of the target's -y axis but expressed in chase vehicle coordinates. The expression

$$A_t^T \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix}$$

can be evaluated with equation (D-5) from Appendix D. The result is equation (IV-10).

The projection of $\underline{s}$ onto the y-z plane of the chase vehicle's coordinate system (which is parallel to the camera focal plane) gives the predicted orientation of the vector in the television image:

$$\theta_2 = \tan^{-1} \frac{s_3}{s_2} \qquad \text{(IV-17)}$$

Perspective effects may produce errors of approximately one or two degrees at close range, because this procedure uses orthographic projection.

The difference between θ_2 and θ_1 is the required amount of rotation about the line of sight, and the axis of rotation is defined by a unit vector in the direction of $\underline{r}$. The axis and rotation define a second quaternion $\underline{q}''$ (equation [IV-12]).

Finally, the computer calculates the net rotation to get the target attitude. The associated quaternion is found from the quaternion product of $\underline{q}'$ and $\underline{q}''$ in equation (IV-13).

4. Mission Constraints and Compatibility

Except for the camera position requirement--the camera must be set back far enough from the front of the chase vehicle to avoid severe image distortion--this approach to video guidance imposes few mission constraints. For example, it can operate in total darkness or full sunlight as long as the sun does not enter the field of view. Unlike the other approaches, this approach allows the docking aid and camera to be offset from the docking fixture, simplifying the design of the system and reducing the impact on other systems. Further, it computes position and attitude fast enough to allow its use with targets that tumble several degrees per second, although other factors may make docking with such a target impractical. (Other parts of the control system may not be able to keep up with such rates, the thrusters may have too little control authority, or the docking mechanism may not be strong enough to withstand the stresses. Moreover, the Kalman filter in the simulated control system was not designed to estimate tumble parameters, and the goal-setting logic used in the system does not optimize trajectories.) With improvements in these portions of the control system, the three-light approach might do much better with tumbling targets than the simulation results suggest.

Although the system requires a computer, it is likely that an autonomous spacecraft will require one with any rendezvous system. Very little additional hardware is needed beyond what one might expect to be required for other functions. The additional hardware consists of a television camera, a handful of electronics weighing perhaps two kilograms including packaging, and a low-power command transmitter with a range of 300 meters. Power requirements will depend on the camera type and other implementation details, but it is reasonable to assume the added hardware can be designed to consume less than ten watts. Because the system is used for only two to four minutes, the energy requirement is negligible when compared to the requirements of other onboard systems.

If the flashing-light approach is used, compatibility with target spacecraft systems must be considered carefully. The spacecraft must provide power to the lights and command receiver, and although the energy requirement is small, it may result in a requirement for solar panels and a power system on a spacecraft that otherwise has no need for them. Other potential compatibility problems with the target spacecraft include interfering with science instruments by blocking their fields of view, overpowering instruments with bright lights, or generating electromagnetic interference. The impacts on propulsion, telemetry, coarse rendezvous, launch vehicle, and ground support systems cannot be meaningfully evaluated without specific details of the mission. However, the small size, ruggedness, low power requirement, and speed of the technique will minimize problems.

5. Measurement Model for Simulation

The simulation program used the perspective projection techniques described in Chapter IX to provide "measurements" from this system. The program simulated motions of the chase vehicle and target between the three light flashes of a single measurement, because this motion produces image distortion that may affect accuracy. The effective resolution of the camera is determined by the random shifting of computed image coordinates in subroutine FLASH. Normally distributed random shifts were added in horizontal and vertical directions. The mean shift was zero, which models negligible fixed biases, and the standard deviation was 1/600th of the width of the field of view, which models an effective resolution of 380 television lines. This is comparable with standard commercial television. Better accuracy was obtained with

ORIGINAL PAGE IS
OF POOR QUALITY

better resolution, and resolution as coarse as 128 lines allowed reliable docking with good control.

B. PROJECTED RAINBOWS AND STEREO RANGEFINDING FORM SECOND SYSTEM

1. Description of Technique

In the second system, a beacon on the target spacecraft projects "rainbows" of light. The apparent color of the beacon, viewed from the chase vehicle, depends on the angle between the docking axis and the line of sight. If the light from the beacon appears blue, the chase vehicle is to the left of the docking axis; if the light appears yellow, the chase vehicle is on the docking axis; if the light appears red, the chase vehicle is to the right of the axis. The beacon also projects a second rainbow and rapidly alternates between the two. The second rainbow shows whether the chase vehicle is above or below the docking axis. In the simulations, it was assumed that the beacon is directly commanded by the chase vehicle by radio, but other approaches are possible. For example, the chase vehicle could send a radio signal to indicate which rainbow is active, or the chase vehicle's guidance system could synchronize to a one-two-off flashing sequence.

To detect the colors, the system uses a prism spectrometer as illustrated in Figure IV-5. Light from the beacon is refracted in the prism, and the angle of refraction is a function of the wavelength or color of the light. A linear Charge Coupled Device (CCD) array behind the grating detects this deflected light, and the position of the brightest spot along the array indicates the color. The system locates the bright spot with a peak detector that monitors the video signal from the CCD array. The peak detector records the number of clock pulses required to scan out the brightest picture element in the line, and the count is used to compute the wavelength or distance from the docking axis.

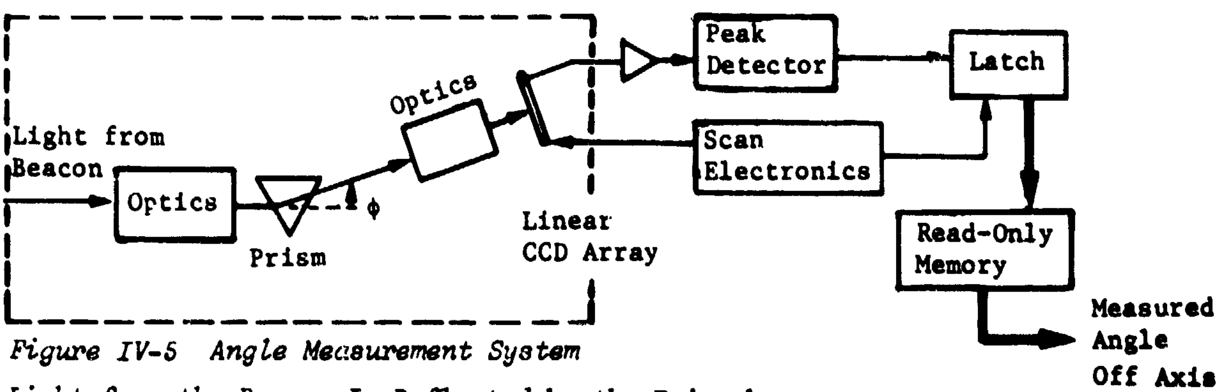


Figure IV-5 Angle Measurement System

Light from the Beacon Is Deflected by the Prism by an Angle ϕ, Which Depends on Wavelength. The Electronics Convert This Deflection to a Measurement of the Chase Vehicle's Angular Separation from the Docking Axis.

ORIGINAL PAGE IS
OF POOR QUALITY

The two television cameras used in the stereo rangefinder are mounted on opposite sides of the chase vehicle (Fig. IV-6) to maximize the interocular distance. Both cameras find the center of brightness, the location of the beacon in the image, by using the centroid detector technique used in the three-light system. The range to the target is computed by comparing the center-of-brightness calculations from the two cameras.

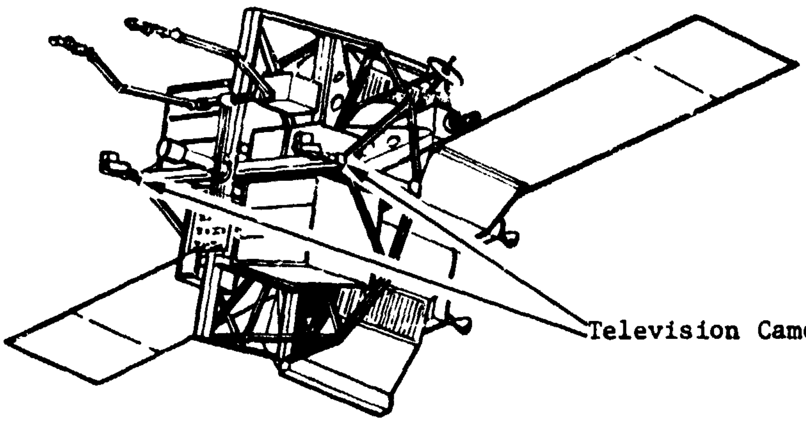


Figure IV-6 Stereo Rangefinder on the Chase Vehicle

The Cameras Are Mounted on Opposite Sides of the Chase Vehicle for Greatest Stereo Effect. The Cameras Are Not Gimballed But Mounted at a Fixed Angle to the Vehicle Frame.

The center-of-brightness information is also used to point the color detector, which has a much smaller field of view than the cameras.

Because accuracy at 300 meters calls for precise alignment, the cameras are mounted rigidly to the frame of the chase vehicle; they are not gimballed.

Figure IV-7 is a block diagram of the rangefinder. Two centroid calculators, similar to the one in Figure IV-2, calculate the image-plane coordinates of the image of the beacon, and an onboard computer uses the coordinates to calculate the distance to the target and to keep the cameras pointed at the target. The system provides 30 range measurements per second.

ORIGINAL PAGE IS
OF POOR QUALITY

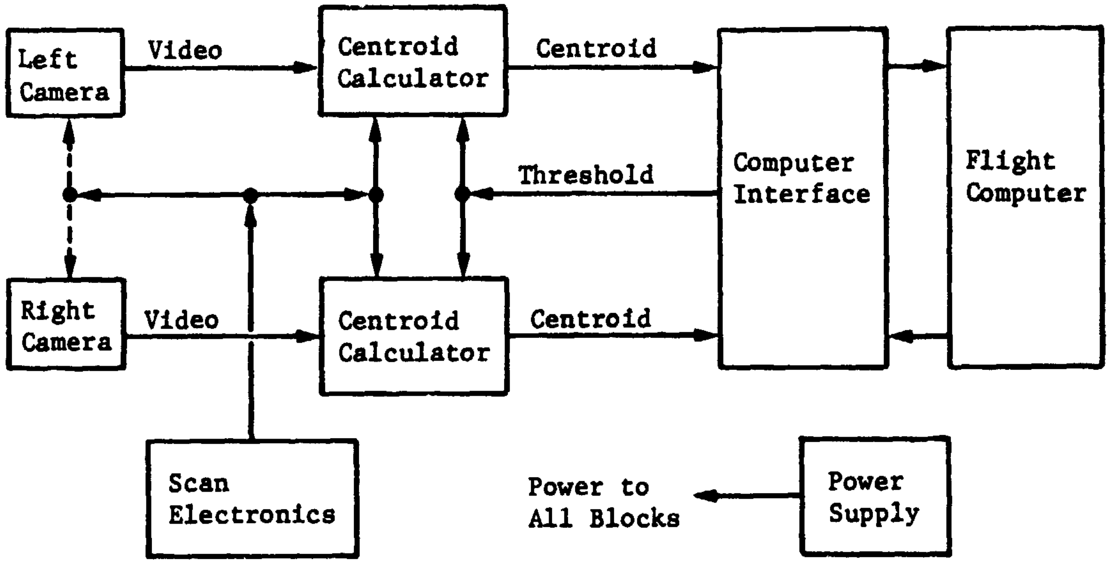


Figure IV-7 Stereo Rangefinder Block Diagram

The Video Signals from the Cameras Are Processed to Determine the Center of Brightness for the Images of a Beacon on the Target Spacecraft. The Range Is Updated by an Onboard Computer Using Equation (4-18). The Centroid Calculators Are the Same As in Figure IV-3.

2. Equations Used

The range formula* is:

$$\rho = \frac{fd(X_\ell + X_r)}{(X_\ell + X_r)^2 - 4(\Delta X)^2} \qquad \text{(IV-18)}$$

where

ρ is the estimated range,
d is the interocular distance (see Figure IV-8),
f is the camera lens focal length,
X_ℓ and X_r are the distances along the image plane from the center of the field of view to the image center of brightness for the left and right cameras, respectively, and

ΔX is the tolerance on X_ℓ and X_r, which includes:

- The effects of having discrete photosensitive sites in the image plane,
- Uncertainty in camera boresighting;
- Scan position uncertainty;

*Equations IV-18 and IV-19 are derived in Table IV-1.

- Tolerances in the electronics that calculate the center of brightness of the image,
- Image blur from motion and lens imperfections.

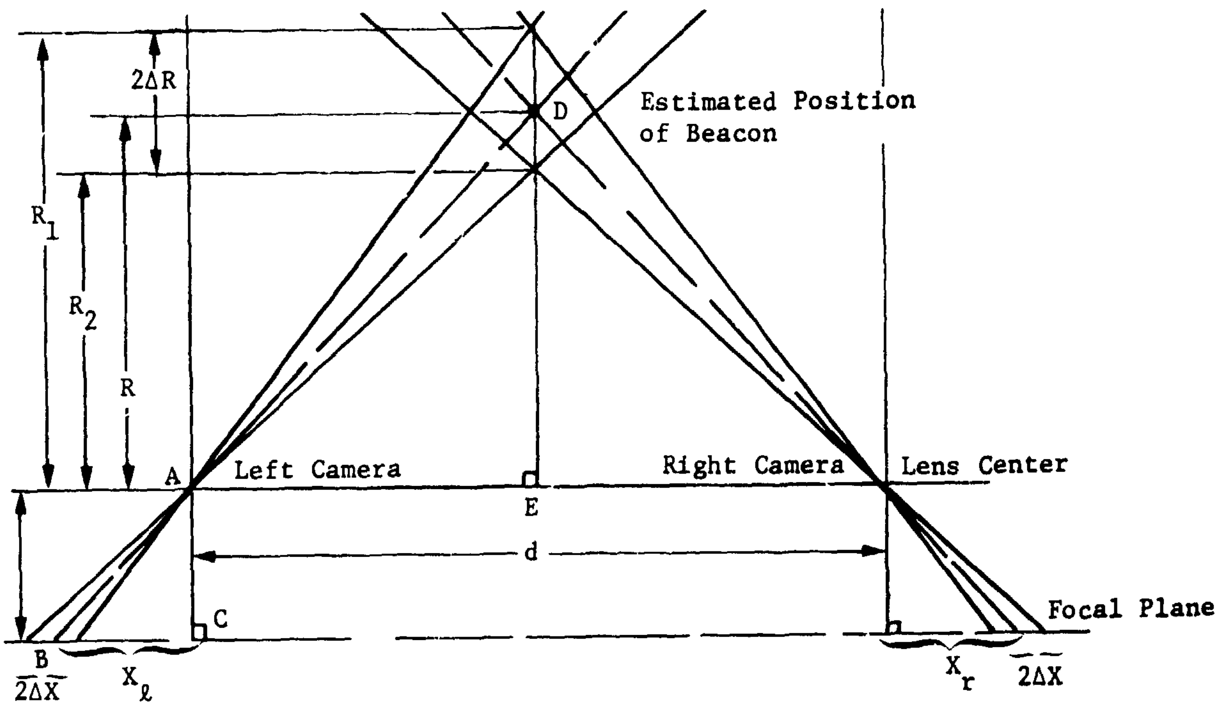


Figure IV-8 The Stereo Rangefinding System and Nomenclature

Because the Guidance System Will Keep the Chase Vehicle Cameras Pointed Toward the Target, the Range Will Equal the Distance along the Perpendicular from the Camera Image Plane to the Object. Uncertainty (ΔX) in the Image Position Causes Uncertainty in Range (ΔR).

In the simulation program (Appendix B) this formula is implemented in subroutine POSIT, where ρ is represented by the variable RHO. If resolution is the only significant limitation on accuracy, the relative measurement error for moderate values of ρ is

$$\frac{\Delta R}{\rho} = \frac{2\rho \Delta X}{fd} \quad \text{(IV-19)}$$

The candidate system, for which ΔX = (line length)/760, has a measurement error of approximately $(0.2\rho/d)$ percent of the true range.

Figure IV-9 is a graph of percent error versus range for a rangefinder with 256 line resolution. Although the accuracy is ±15% at the start of the rendezvous operation, the system achieves accuracy better than 5% at ranges where accuracy is critical.

ORIGINAL PAGE IS
OF POOR QUALITY

Table IV-1 Derivation of Formulas (IV-18) and (IV-19)

1. Range - Consider the diagram in Figure IV .. The maximum possible range, R_1, can be determined by noting the similar triangles ABC and ADE. Since the ratios among the sides of similar triangles are equal,

$$\overline{AE} = (R_1)(X_\ell - \Delta x)/f \qquad \text{(IV-20)}$$

The same reasoning for the right camera gives

$$d - \overline{AE} = (R_1)(X_r - \Delta x)/f \qquad \text{(IV-21)}$$

Combining (IV-20) and (IV-21) gives

$$\frac{(R_1)(X_\ell - \Delta x)}{f} + \frac{(R_1)(X_r - \Delta x)}{f} = \overline{AE} + d - \overline{AE} = d \qquad \text{(IV-22)}$$

which can be manipulated algebraically to give the maximum possible range:

$$R_1 = fd/(X_\ell + X_r - 2\Delta x) \qquad \text{(IV-23)}$$

The same reasoning gives the minimum possible range:

$$R_2 = fd/(X_\ell + X_r + 2\Delta x). \qquad \text{(IV-24)}$$

The average of R_1 and R_2 is the estimated range:

$$\rho = \tfrac{1}{2}(R_1 + R_2) = \frac{fd(X_\ell + X_r)}{(X_\ell + X_r)^2 - 4(\Delta x)^2} \qquad \text{(IV-25)}$$

2. Range Tolerance - The range tolerance is half the difference between R_1 and R_2:

$$\Delta R = \tfrac{1}{2}(R_1 - R_2) = \frac{2fd\Delta x}{(X_\ell + X_r)^2 - 4(\Delta x)^2} \qquad \text{(IV-26)}$$

Dividing (IV-26) by (IV-25) gives the relative error

$$\frac{\Delta R}{\rho} = \frac{2\Delta x}{X_\ell + X_r} \qquad \text{(IV-27)}$$

At moderate ranges, $(X_\ell + X_r)^2 \gg 4(\Delta x)^2$. [In the candidate system, $(X_\ell + X_r)^2$ is over 40 times as large as $4(\Delta x)^2$ for ranges to 300 m.]

So Δx can be ignored in (IV-25) with little error:

$$\rho \simeq \frac{fd}{X_\ell + X_r} \qquad \text{(IV-28)}$$

Table IV-1 Concl

Substitution of (IV-28) into (IV-27) gives
$\frac{\Delta R}{\rho} \approx \frac{2\rho\Delta x}{fd}$ (IV-29)

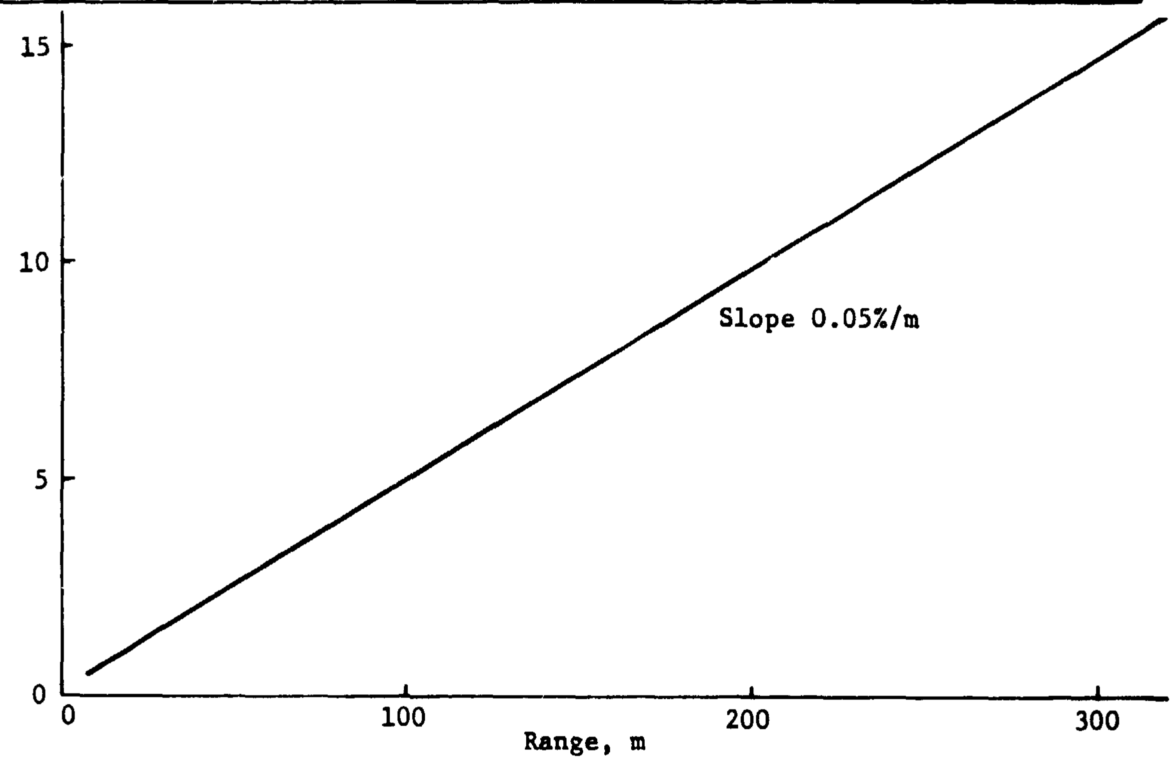


Figure IV-9 Percent Error vs Range for Stereo Rangefinding System With a 30-degree Field of View and 256 Pixels Per Line. The Percentage Error Increases Linearly with Range at the Rate of 0.05%/m.

Because the two cameras used for stereo ranging are not gimballed, they cannot look "cross-eyed" at the beacon. As the beacon moves betweeen the cameras in the last moments before contact, it leaves the fields of view of both cameras. A third camera is therefore used to allow at least pointing, if not ranging, during the last few seconds of the docking operation. This camera is mounted very close to the docking fixture so that the beacon will always be in its field of view.

The third camera has a circuit like the one in Figure IV-2 to compute the beacon's center of brightness. The coordinates from this circuit are used in the flight computer to compute a vector $\underline{c}$, which defines the camera's position in a coordinate system parallel to the primary coordinate system but centered at the beacon. The equation used is identical to equation (IV-6), except that the variable ρ (range) now comes from the stereo rangefinding system. The vector $\underline{c}$ plays the same role in this guidance system as in the three-light system.

From the data provided by the beacon color analyzer, the system "knows" its angular separation from the docking axis. The analyzer provides two angles, one for horizontal error ("yaw") and one for vertical error ("pitch"). These angles are combined with the range information (ρ) to give $\underline{r}$, the camera's position in a reference frame that is parallel to the target's body axes but centered at the beacon. In the simulation program in Appendix B, the calculations are found in subroutine POSIT, and $\underline{r}$ is represented by the variable RELPOS.

The flight computer therefore has available to it vectors that are analogous to the $\underline{r}$ and $\underline{c}$ vectors of the three-light system.

Unfortunately, the computer has no information to help it resolve the ambiguity in orientation about the line of sight. Unless it can resolve this ambiguity, it cannot determine which way to steer to get to the docking axis.

The solution to this problem is found in combining two separate measurements. Suppose, for example, that one measurement gives $\underline{r}$ and $\underline{c}$ and that the second measurement gives $\underline{r}'$ and $\underline{c}'$. If the measurements are closely spaced in time, the target's attitude will not have time to change appreciably. Therefore only two coordinate systems are involved: the primary system, which is known, and the target system, which is to be computed. If A_t is the direction cosine matrix (see Appendix D) giving the target's attitude with respect to the primary frame,

$$\underline{r} = A_t \, \underline{c} \text{ , and} \qquad \text{(IV-30)}$$

$$\underline{r}' = A_t \, \underline{c}' \text{ .} \qquad \text{(IV-31)}$$

This is true because $\underline{r}$ and $\underline{r}'$ are expressed in a frame parallel to the target frame and the frame in which $\underline{c}$ and $\underline{c}'$ are expressed is parallel to the primary frame and has the same origin as the frame for $\underline{r}$ and $\underline{r}'$.

The relationship illustrated in equations (IV-30) and (IV-31) holds for any pair of vectors expressed in these two frames. For example, the cross product of $\underline{c}$ and $\underline{c}'$ is related to the cross product of $\underline{r}$ and $\underline{r}'$ by the same direction cosine matrix. So the relationship holds between the two matrices:

$$M_r = \left[\underline{i}_r \,\vdots\, \underline{j}_r \,\vdots\, \underline{k}_r\right] \text{ and} \qquad \text{(IV-32)}$$

$$M_c = \left[\underline{i}_c \,\vdots\, \underline{j}_c \,\vdots\, \underline{k}_c\right] \qquad \text{(IV-33)}$$

in which (IV-34)

$$\underline{i}_r = \frac{\underline{r}}{|\underline{r}|} \qquad \underline{i}_c = \frac{\underline{c}}{|\underline{c}|} \text{ ,}$$

ORIGINAL PAGE IS
OF POOR QUALITY

$$\underline{j}_r = \frac{\underline{r} \times \underline{r}'}{|\underline{r} \times \underline{r}'|} \qquad \underline{j}_c = \frac{\underline{c} \times \underline{c}'}{|\underline{c} \times \underline{c}'|} , \tag{IV-35}$$

$$\underline{k}_r = \underline{i}_r \times \underline{j}_r \qquad \underline{k}_c = \underline{i}_c \times \underline{j}_c \tag{IV-36}$$

Because of the way M_r and M_c are constructed (they are 3x3 matrices in which each column is a unit vector of a rectangular coordinate system), these matrices have the special property that their inverses are equal to their transposes. The flight computer can therefore find A_t from these matrices by the formula:

$$A_t = M_r M_c^T . \tag{IV-37}$$

In the simulation, this formula is evaluated in subroutine QUATRN. The validity and usefulness of the formula depends on two requirements:

1) The vector r must differ significantly in orientation from r', and c must differ significantly from c'. Otherwise, small errors in these vectors may make the results totally random.

2) The target spacecraft must not rotate much between the time r and c are measured and the time r' and c' are measured.

Unfortunately these two requirements conflict: if the two measurement sets are taken in rapid succession to minimize target rotation between readings, the measurement sets will differ by very little. However, if there is much delay between the two sets, the target may have rotated enough to invalidate the assumptions behind the formula. The delay also requires the chase vehicle to use stale data for an extended period of time.

In the simulation program we tried only one approach to dealing with this conflict: we attempted to optimize the time interval between measurements. There are alternative strategies.

One strategy is to move the chase vehicle in a path perpendicular to the line of sight from time to time to maximize the difference between the measurement sets. This strategy would be combined with an expanded Kalman filter that estimates target tumble parameters. The additional cost in system complexity would be small, but the approach wastes fuel and places a much heavier burden on the flight computer.

An alternate strategy is to use two color-decoding circuits instead of one. One would be placed midway between the middle camera and the left camera; the other would be placed midway between the middle camera and the right camera. The resulting system would provide two separate color/rangefinder systems that would allow simultaneous measurement of both sets of data. Because of the close spacing of the instruments, the vectors in the two sets would be almost parallel until the chase vehicle gets close to the target. This implies that good attitude information is not available at great distances. If the target's tumble rate is small enough so that elaborate trajectory planning is not required, this should cause few problems.

We did not simulate these approaches, because we felt that they would result in a system that was so much more complex and expensive than the three-light system that even if they worked well, they would not be recommended.

3. Mission Constraints and Compatibility

Camera placement is a bigger concern with this approach than with the three-light system. Although the beacon, center camera, and color analyzer could be offset from the docking fixtures, there is a risk in doing so. The problem is that the algorithm for computing target attitude is not very robust. The factors discussed previously (too little chase vehicle motion or excessive target attitude change betweeen measurements) can cause errors in attitude computations. This problem is compounded when the stereo ranging system becomes "blind" during the critical seconds just before docking as the beacon moves between the cameras, out of their fields of view.

The loss of stereo ranging might be cured by making the outboard cameras "crosseyed." Although time did not permit a simulation of this modification, we believe it will greatly improve performance at close range. However, this change tightens constraints on camera placement and spacecraft design, since it will be harder to avoid obstructing the fields of view of the cameras.

Because the "rainbow" beacon approach requires a significant delay between observations to properly derive target attitude, the approach is inherently slower than the three-light approach. We do not recommend it for use with rapidly tumbling targets. With modifications to the Kalman filter, it may be able to cope with modest tumble rates, but this has not been demonstrated.

The burden on the onboard computer is greater with this approach than it was with the three-light approach, and it requires more hardware on the target and chase vehicle. If one considers only the hardware that would not be required without the video guidance system, the difference in quantity is approximately a factor of five. The power requirement and weight can also be expected to increase by a factor of five. Furthermore, the hardware in the "rainbow" system is more complex and less rugged.

The beacon cannot use flash lamps, because it must produce a continuous spectrum. Although careful design may allow a low-power rainbow projection system, we doubt that power consumption can be reduced to within a factor of ten of what is possible with a three-light system.

4. Measurement Model for Simulation

The measurement model for simulating the television cameras in the "rainbow" system was identical to the model used for the three-light system. The same resolution was modeled, and the same assumptions were made. The calculations are in subroutine CENTRD.

The model for the rainbow itself, however, is unique. The program first computes the true angular error from the docking axis about the target z and y axes. This calculation fully takes into consideration the offset between the beacon and the target spacecraft center of mass and the offset between the color analyzer and the center of mass of the chase vehicle.

The validity of the measurement is then tested. If the color decoder cannot "see" the beacon, the program recognizes that the measurement is unuseable, and the guidance system ignores the measurement.

Otherwise, the program adds a random number to the computed angle to simulate imperfections in the system. The accuracy of a "pitch" measurement can be expected to deteriorate when the "yaw" angle becomes large. Similarly, "yaw" measurements will be poor when the "pitch" angle is large. As the colors get crowded close together near the axis about which they are spread out, it becomes harder to separate them. The program models this effect by making the corruption to "yaw" measurements proportional to "pitch" and by making the corruption to "pitch" measurements proportional to "yaw."

The program then rounds off the simulated measurements to the nearest multiple of 0.01 radian and limits the range of measurements to $\pm$ 1 radian. This procedure models the inherent quantization in the angle decoding circuit and the limited rainbow width achievable with a practical rainbow projector.

The computations for "yaw" measurements are in subroutine GETYAW. The computations for "pitch" measurements are found in subroutine GETPCH. The structures of these subroutines are virtually identical. Subroutine DOCK selects one or the other of these subroutines for each measurement to model the fact that both measurement types cannot be taken simultaneously.

C. IMAGE OF RING OF LIGHTS LEAVES ROLL UNDEFINED IN THIRD SYSTEM

1. Description of Technique and Equations Used

Figure IV-10 shows a third docking aid that can be analyzed with a small dedicated video signal processor. The required circuit (Fig. IV-11) uses the equations in Figure IV-12 to analyze a television image of the docking aid in 1/30 second. In the simulation program of Appendix C, this hardware analysis is simulated in subroutines LOCATE

ORIGINAL PAGE IS
OF POOR QUALITY

and MCALC. The circuit computes parameters defining an ellipse tha approximates the appearance of the image. These parameters can be interpreted in software to get position and attitude, except that target rotations about the target x axis produce no visible effect. This ambiuguity does not matter if the docking fixture is at the center of the ring of lights and allows mating with an arbitrary roll misalignment and if the camera is mounted at the center of the chase vehicle's docking fixture.

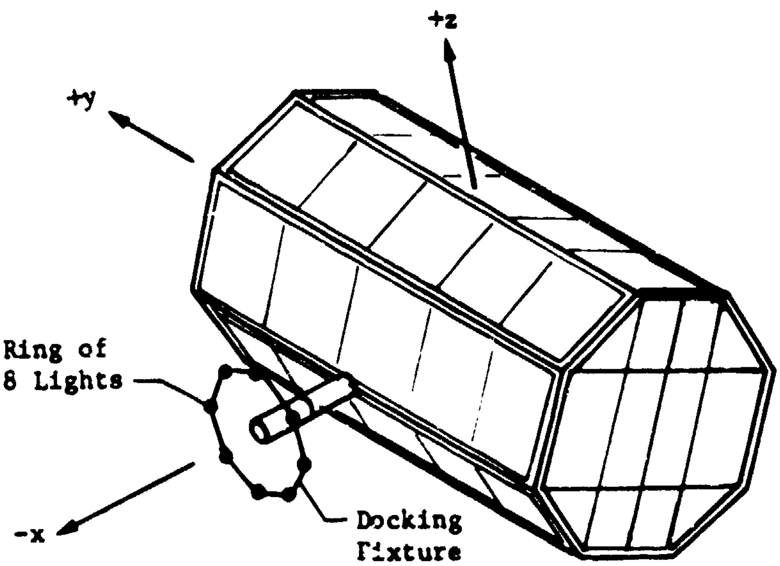


Figure IV-10 Ring-of-Lights Docking Aid

Ring Encircles Docking Fixture. Spacecraft Body Coordinate System Is Shown. Docking-Aid Coordinate System Is Parallel to Body Frame But Centered at the Middle of the Ring of Lights.

Because the video processor analyzes the image as a whole, it is important for the ring of lights to be the only significant source of light in the image. Flashing-light and color-keying approaches can be used with this docking aid in the same way they were used in the three-light system. All of the lights in the ring must flash simultaneously if flash tubes are used.

The parameters X_c and Y_c from the scene analysis circuit are the coordinates of the center of the ellipse on the camera's focal plane. The moments I_{xx}, I_{xy}, and I_{yy} are intermediate results that must be processed further in the flight computer. The computer calculates the ellipse semimajor and semiminor axes (a and b, respectively) and the rotation of the semimajor axis from horizontal (θ). The formulas are:

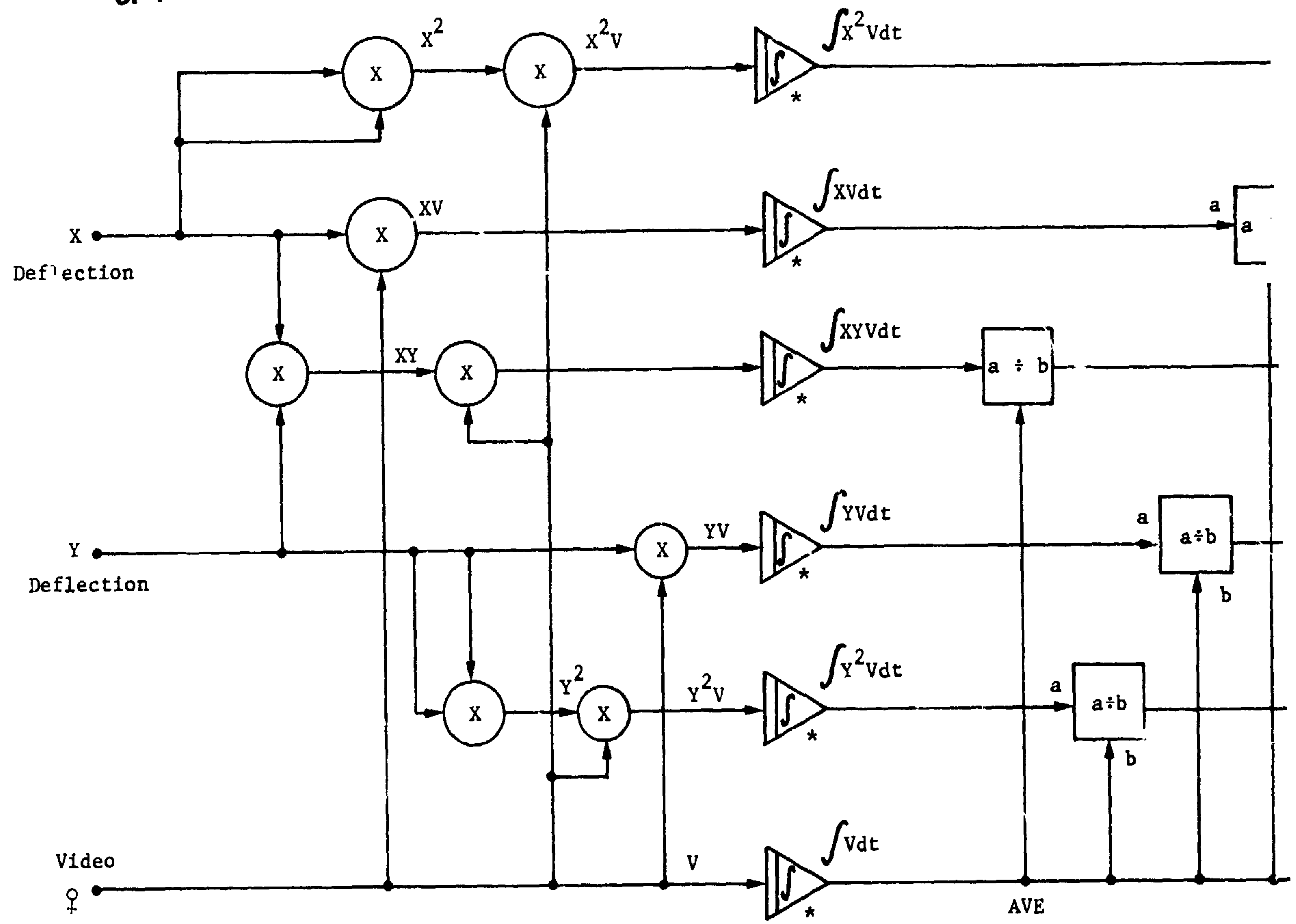


Figure IV-11 Possible Hardware to Compute Centroid and Ellipse Parameters

$$AVE = \int_{Frame} V\,dt$$

$$X_c = \frac{\int_{Frame} X \bullet V\,dt}{AVE} \qquad Y_c = \frac{\int_{Frame} Y \bullet V\,dt}{AVE}$$

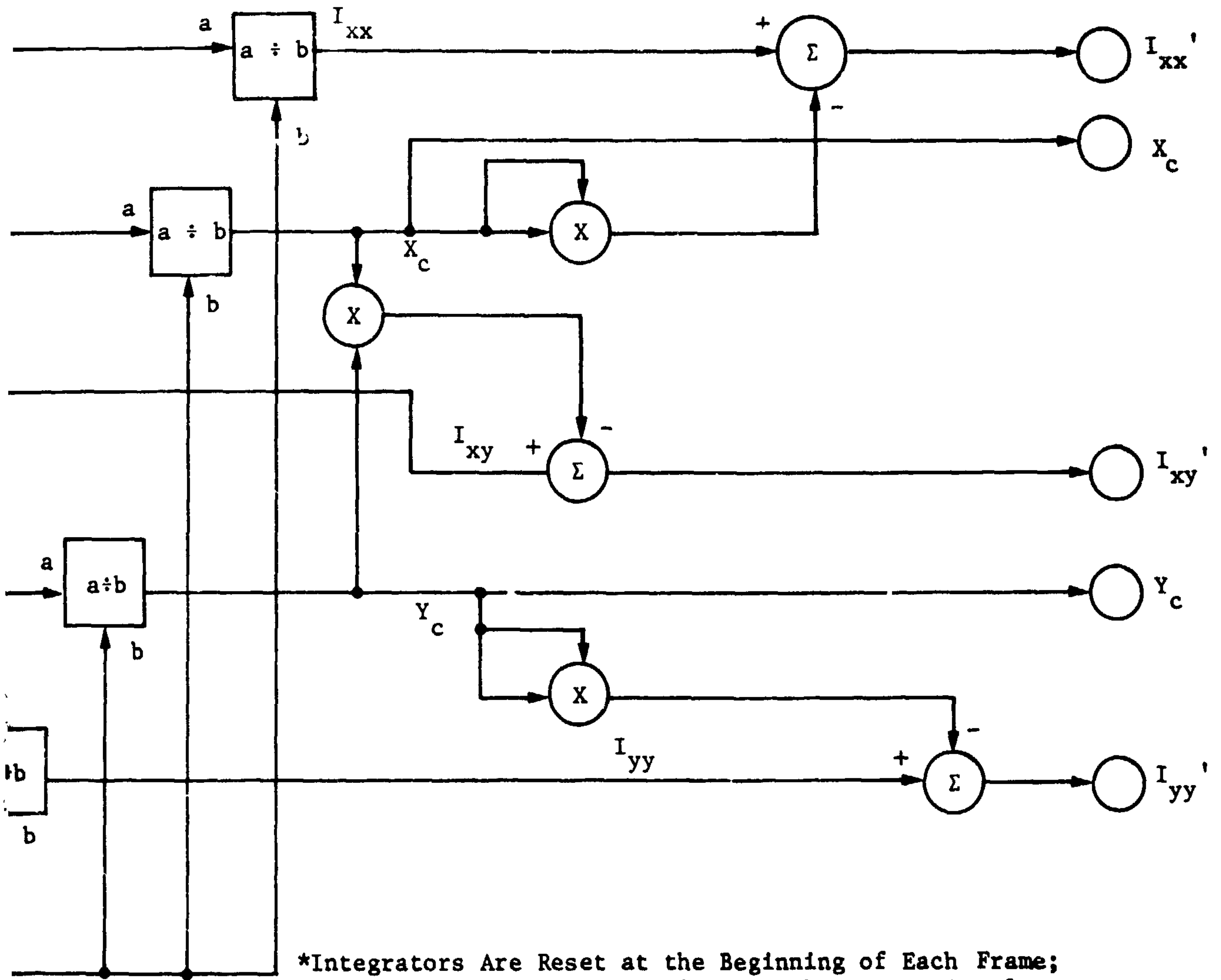
$$I_{xx} = \int_{Frame} V \bullet X^2\,dt/AVE \qquad I_{xx}'$$

$$I_{yy} = \int_{Frame} V \bullet Y^2\,dt/AVE \qquad I_{yy}'$$

$$I_{xy} = \int_{Frame} V \bullet X\,Y\,dt/AVE \qquad I_{xy}'$$

Figure IV-12 Equations Solved by Circuit of Figure IV-11

FOLDOUT FRAME



*Integrators Are Reset at the Beginning of Each Frame;
The Integration May Be Replaced with a Summation for a Digital Implementation;
♀ Thresholding of Video Signal Is Optional

$$I_{xx}' = I_{xx} \quad X_c^{\,2}$$

$$I_{yy}' = I_{yy} - Y_c^{\,2}$$

$$I_{xy}' = I_{xy} - X_c \bullet Y_c$$

$$\theta = \tfrac{1}{2}\tan^{-1}\left(\frac{2\,I'_{xy}}{I'_{xx} - I'_{yy}}\right) \qquad \text{(IV-38)}$$

$$\lambda_1 = \tfrac{1}{2}\left(I'_{xx} + I'_{yy} + \sqrt{(I'_{xx} - I'_{yy})^2 + 4I'^2_{xy}}\right) \qquad \text{(IV-39)}$$

$$\lambda_2 = \tfrac{1}{2}\left(I'_{xx} + I'_{yy} - \sqrt{(I'_{xx} - I'_{yy})^2 + 4I'^2_{xy}}\right) \qquad \text{(IV-40)}$$

$$a = k\sqrt{\max(\lambda_1, \lambda_2)} \qquad \text{(IV-41)}$$

$$b = k\sqrt{\max(\lambda_1, \lambda_2)} \qquad \text{(IV-42)}$$

where k is a constant that depends slightly on the exact target geometry. In this simulation, which models eight-point light sources, $k = \sqrt{2}$. These calculations are performed in subroutine EPAR in the simulation program.

Because the roll component of the target's attitude is not observable, the computer assigns it an arbitrary value. Specifically, roll orientation is assumed to place the camera in the X-Z plane of the docking aid coordinate system. This coordinate system is parallel to the target spacecraft system but is centered at the middle of the docking aid.

With this arbitrary roll value assigned, a vector $\underline{r}$ can be computed, which is analogous to the vector $\underline{r}$ in the other two video guidance systems discussed previously. The formulas are:

$$\rho = 0.5df/a \qquad \text{(IV-43)}$$

$$\beta = b/a \qquad \text{(IV-44)}$$

$$\underline{r} = \begin{bmatrix} -\rho\beta \\ 0 \\ \rho\sqrt{1 - \beta^2} \end{bmatrix} \qquad \text{(IV-45)}$$

where:

ρ is the range to the target,

β is the ratio of the lengths of the semiminor axis and semimajor axis of the ellipse,

$\underline{r}$ is the position of the camera in a reference frame that is parallel to the target spacecraft frame but centered at the center of the ring of lights.

In the simulation program, these equations are implemented in subroutine ELIP, and $\underline{r}$ is represented by the array RELPOS.

This system also computes a vector $\underline{c}$, analogous to the vector $\underline{c}$ in the other two systems. This vector specifies the camera's location in a reference frame that is parallel to the "primary" reference frame used for navigation. The origin of this frame is at the center of the ring of lights. Equation (IV-6) is used to compute this vector as in the other two systems, except that X_c and Y_c from the video-processing hardware fill the role of u_2 and v_2 in the equation.

In the other two systems, $\underline{r}$ and $\underline{c}$ were combined to determine the attitude of the target. The same approach is used in this system, but the details of their use differ.

In the three-light system, there was no ambiguity in the interpretation of an observation. A single measurement was therefore sufficient to define the chase vehicle's position. In the "rainbow" system, there was an ambiguity because target rotations about the line of sight were not observable. Two different measurements were therefore combined to determine the target's attitude uniquely.

In the ring-of-lights system, there are two sources of ambiguity. First, target rotation about its x axis is undetectable. This ambiguity is unresolvable but causes no problems if the ring of lights encircles the docking fixture as was discussed previously. The second ambiguity is of greater concern; the target's center of mass may be on either side of the major axis of the ellipse (Fig. IV-13). If the guidance system simply guesses, it will drive the chase vehicle farther from the docking axis. Instead of correcting position errors, it will make them worse.

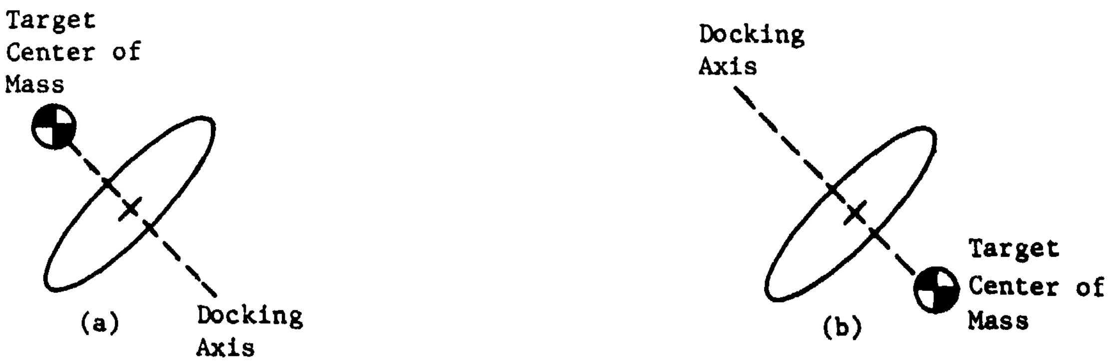


Figure IV-13 Two Ellipse Interpretations

In (a) the Chase Vehicle Should Steer Down and to the Right to Reach the Docking Axis. In (b) the Chase Vehicle Should Steer Up and to the Left. A Wrong Guess Results in an Unstable Control System.

This problem is also solved by combining two measurements, but the method for combining the measurements is completely different.

The procedure starts by picking a plausible interpretation from the equations:

$$\underline{w} = \underline{r} \times \underline{c} \quad \text{(cross product)}, \tag{IV-46}$$

$$\phi = \tan^{-1}\left(\frac{|w|}{\underline{r}\cdot\underline{c}}\right), \tag{IV-47}$$

$$\underline{q}' = \begin{bmatrix} \underline{w}\sin(\phi/2)/|w| \\ \cos(\phi/2) \end{bmatrix} \tag{IV-48}$$

These equations are identical to equations (IV-7) through (IV-9), which were used in the first two systems, and they serve the same purpose here. The quaternion $\underline{q}'$ defines a target attitude that aligns the $\underline{r}$ and $\underline{c}$ vectors. This quaternion must now be corrected for rotation about the line of sight. In the first system this was done by comparing the observed target -y axis with a predicted observation that was based on $\underline{q}'$. With the ring-of-lights docking aid, the -y axis cannot be observed. The -x axis, however, is known to lie along the ellipse minor axis in the image, and this fact can be used to form the correction for rotation about the line of sight. The reasoning is identical to that used in deriving equations (IV-10) and (IV-11), but the use of a different axis results in slightly different formulas:

$$\underline{S} = A_c \begin{bmatrix} -q_1'^2 + q_2'^2 + q_3'^2 - q_4'^2 \\ -2(q_1' q_2' + q_3' q_4') \\ 2(q_2' q_4' - q_1' q_3') \end{bmatrix}, \tag{IV-49}$$

$$\gamma = \tan^{-1}\left(\frac{S_2}{S_3}\right) - \theta . \tag{IV-50}$$

where $\underline{s}$ is the projection of the target -x axis unit vector onto the chase vehicle's y-z plane. The angle γ is the amount of rotation about the line of sight required to make the predicted ellipse appearance match the observed ellipse. The angle θ, the orientation of the ellipse in the image, is computed by the video signal processing hardware shown in Figure IV-11.

The corresponding quaternion is computed with the equations:

$$\underline{q}'' = \begin{bmatrix} \underline{r}\sin(\gamma/2)/|\underline{r}| \\ \cos(\gamma/2) \end{bmatrix}, \tag{IV-51}$$

$$\underline{q} = \underline{q}' \, \underline{q}'' \; . \qquad \text{(IV-52)}$$

The alternate interpretation, in which the target spacecraft's center of mass lies on the opposite side of the major axis of the ellipse, is formed by rotating the target 180 degrees about the line of sight. This rotation is done with the formulas:

$$q''' = \begin{bmatrix} \underline{r}/|\underline{r}| \\ 0 \end{bmatrix} , \qquad \text{(IV-53)}$$

$$\underline{q}^* = \underline{q} \, \underline{q}''' \; . \qquad \text{(IV-54)}$$

All these equations are implemented in the simulation program in subroutine QUATRN, where $\underline{q}$ and $\underline{q}^*$, the possible target attitude quaternions, are represented by the first and second columns of the 4x2 array QT, respectively.

After the first observation of the target, the guidance system has no basis for preferring one attitude interpretation over the other, but after the second and subsequent observations, it can compare $\underline{q}$ and $\underline{q}^*$ from the current observation with the values from the previous observation. If the target is not tumbling too rapidly, one pair of attitude measurements will match closely. This pair can be assumed to represent the proper image interpretation.

One problem in implementing this method is that the quaternions all contain arbitrary components representing roll about the target x axis. Some means is required for comparing two quaternions while ignoring this roll component in each. The guidance system simulated in this study makes this comparison with the equations:

$$a = q_1Q_1 + q_2Q_2 + q_3Q_3 + q_4Q_4 \; , \qquad \text{(IV-55)}$$

$$b = q_1Q_4 + q_2Q_3 - q_3Q_2 - q_4Q_1 \; , \qquad \text{(IV-56)}$$

$$\delta = \cos^{-1}\sqrt{a^2 + b^2} \qquad \text{(IV-57)}$$

where $\underline{q}$ and $\underline{Q}$ are any two quaternions to be compared, and δ is smallest possible Euler angle that will account for the difference between $\underline{q}$ and $\underline{Q}$, making the most generous assumptions about the arbitrary roll angle. Here variables a and b are intermediate results and are used only to simplify equation (IV-57). They should not be confused with the ellipse parameters used in other formulas.

Equations (IV-55) through (IV-57) are implemented in the simulation program in subroutine TSTATT. This subroutine is called by subroutine SELEC1 to test all possible combinations of old and new target attitude interpretations to find the best match. SELEC1 returns pointers that indicate which pair is assumed to represent the true target attitude.

The guidance system uses the selected interpretation of the current observation for control, but it remembers both interpretations for testing the next observation and for propagating the state estimate. Two sets of parameters are maintained for the onboard mathematical dynamics model. After each observation it throws away the parameter set that was based on the target attitude interpretation that the current observation shows to be false. This maintenance of two parameter sets minimizes errors caused by operation with a tumbling target.

2. Mission Constraints and Compatibility

Because this system cannot detect roll misalignment of the two spacecraft, it is essential that such a misalignment makes no difference. This requires the camera to be mounted very close to, or even within, the docking fixture. The ring of lights must encircle the docking fixture of the target spacecraft. Also, the docking fixtures must be designed to operate properly with an arbitrary roll misalignment.

The hardware power requirement for this system will probably be only slightly more than that of the three-light system, and the hardware is not much more complex than that system's hardware. The burden on the flight computer is also slightly greater.

In all other respects, the mission constraints imposed by this system will be essentially the same as those imposed by the three-light system.

3. Measurement Model for Simulation

The simulation program used the perspective projection techniques described in Chapter IX to provide "measurements' from this system. The center of brightness coordinates for each light were corrupted to represent 380-line television resolution. The method used to do this was the same as the method used with the three-light system. The equations in Figure IV-12 were simulated with summations that approximated the required integrals. These calculations are in MCALC of the simulation program.

V. Simulation Results and Discussion

V. Simulation Results and Discussion

V. SIMULATION RESULTS AND DISCUSSION

A. THE THREE-LIGHT SYSTEM WORKS BEST

In more than 50 simulations run with the computer programs in Appendices A through C, the three-light version consistently outperformed the other two systems. Figure V-1 shows six typical trajectories produced with the three-light system. Each simulation started with the chase vehicle at a random position approximately 300 meters from the target spacecraft and was assigned a random initial velocity. Target attitude varied from one simulation to the next, but the target's attitude rate was zero. In each of these simulations, and several others not shown, docking was successful.

Figure V-2 and V-3 show the results of similar simulations with the "rainbow" and ring-of-lights systems, respectively. Both systems performed reasonably well at distances over 150 meters from the target, but both had trouble at close range. The "rainbow" system successfully docked or came very close most of the time, but the final alignment was never as good as the three-light system achieved. More serious is the fact that the system occasionally became "confused" and wandered away from the target. This is what the control system is supposed to do if the target leaves the field of view of the television cameras. It is a preprogrammed maneuver to avoid a collision when the system is "blind." However, the system should not have had to take this action with such benign starting conditions.

In some simulations where the simulation time limit was increased, the chase vehicle made another pass at the target when this happened. The onboard mathematical model, described in Chapter VIII, was provided to allow recovery from a temporary loss of imagery. We believe, however, that it is significant that the three-light version did not have the problem.

Three explanations can account for this behavior of the "rainbow" system:

1) The projected rainbow does not produce valid data when it is viewed from more than one radian from the docking axis. This restriction models what we believe to be a reasonable estimate of the limitations of a practical system;

2) The stereo rangefinder's cameras are mounted at the sides of the chase vehicle, and at close range the image of the beacon moves toward the edge of each camera's field of view. Because of these two limitations, small errors in position and attitude at close range can cause loss of ranging information or incorrect angle measurements;

3) The chase vehicle may be moving too little between successive observations, causing errors in determining the target's attitude.

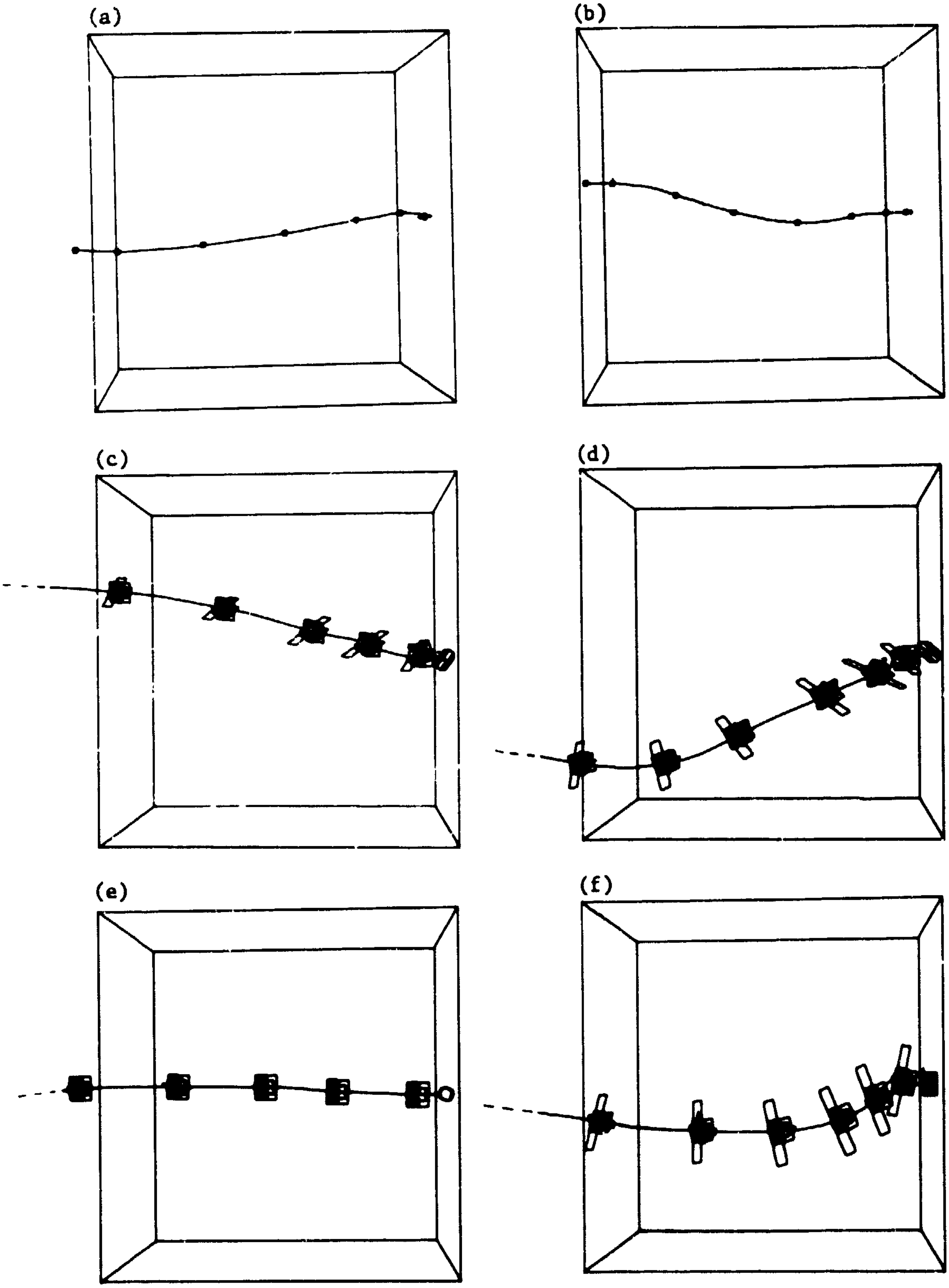


Figure V-1
Typical trajectories with three-light system: a) and b) show entire trajectory; c) through f) are closeup views of the last 60 m. All trajectories had an initial range of approximately 300 m.

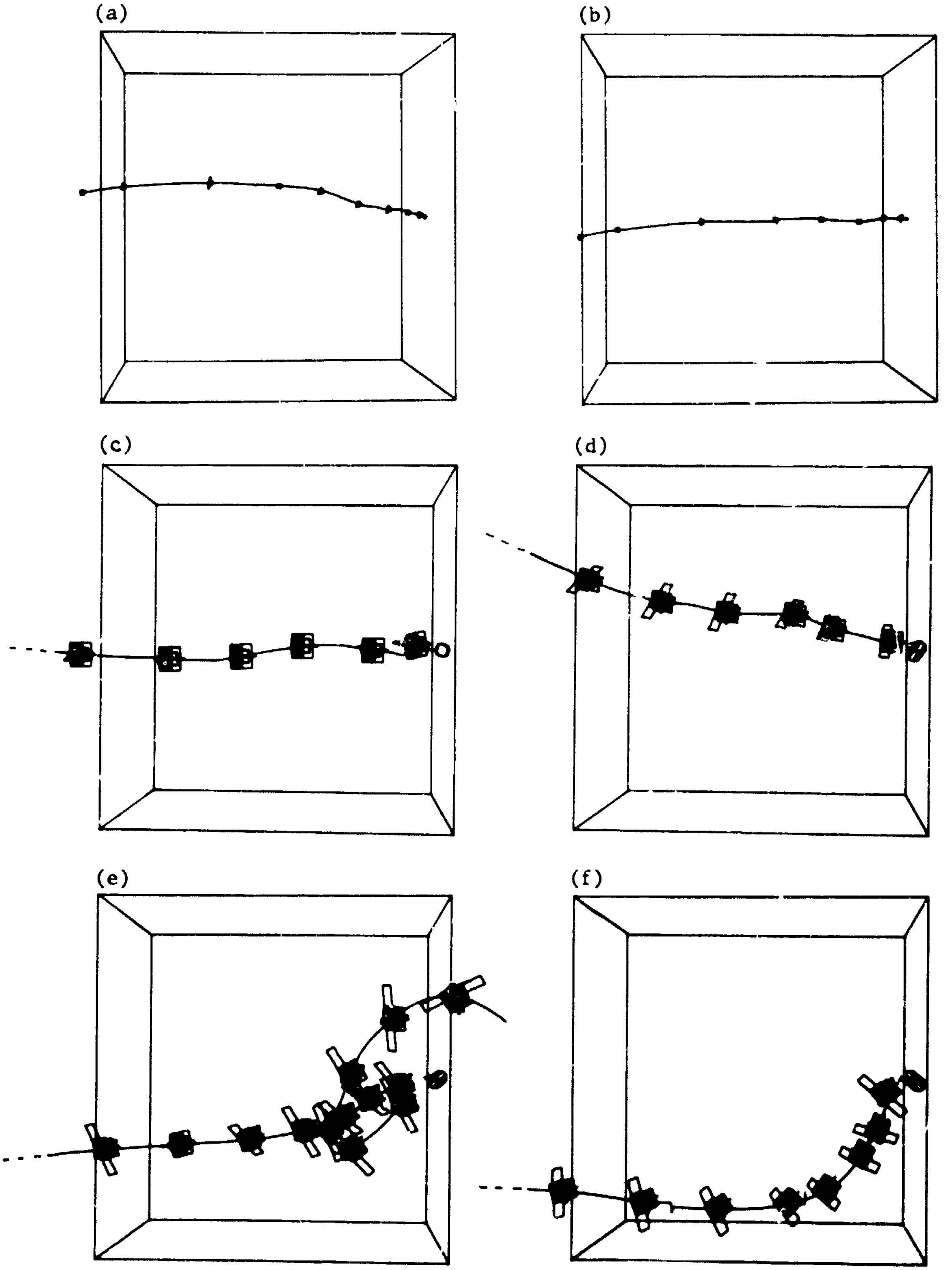


Figure V-2
Typical Trajectories with rainbow system: a) and b) show entire trajectory; c) through f) are closeup views of the last 60 m. All trajectories had an initial range of approximately 300 m.

ORIGINAL PAGE IS
OF POOR QUALITY

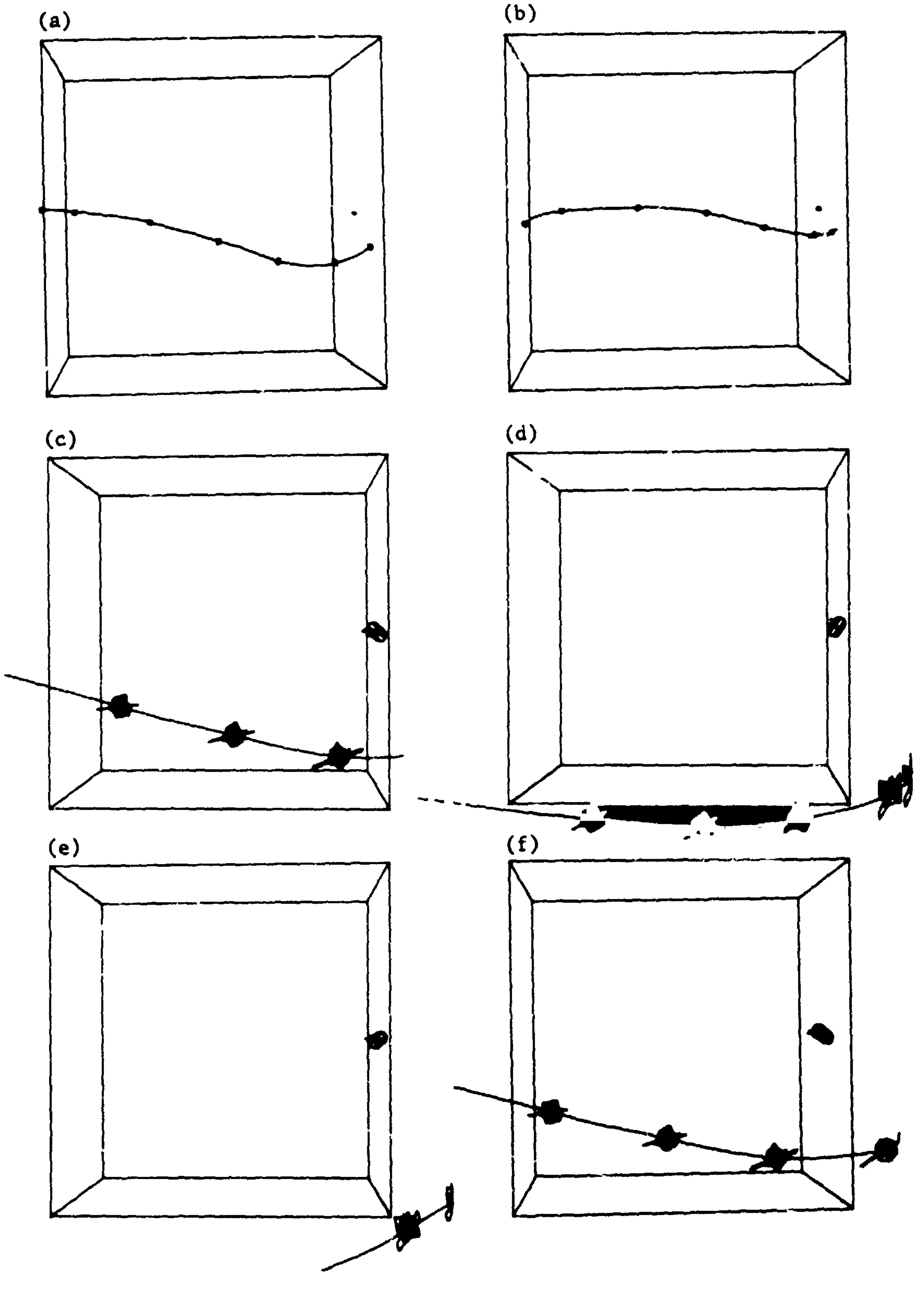


Figure V-3
Typical Trajectories with Ellipse system: a) and b) show entire trajectory; c) through f) are closeup views of the last 60 m. All trajectories had an initial range of approximately 300 m.

The ring-of-lights system never docked successfully. In fact, in most of the simulations it appeared to actively avoid the target spacecraft. The problem appears to be in the method it uses to resolve ambiguity in interpreting the image of the ring of lights. This ambiguity (Fig. IV-13), is resolved by combining pairs of successive observations, and the algorithm used is not robust enough to cope with modeled imperfections in the measurements and control system. We have found no simple algorithm change that measurably improves performance.

B. THE THREE-LIGHT SYSTEM WORKS WITH TUMBLING TARGETS

Because the "rainbow" and ring-of-lights systems did not work well even with an inertially stable target, we believed it was pointless to test them with tumbling targets. We therefore concentrated on testing the three-light system.

Figure V-4 shows trajectories for target roll rates of 540 to 40,000 degrees per hour. The control system appears to handle rates up to 20,000 degrees per hour with good accuracy. Above this rate, performance is degraded by gyroscopic torques caused by attempting to maintain roll alignment with the target at all times. If the chase vehicle did not rotate in synchronism with the target, much better performance could be expected, because the accuracy of the measurements is not measurably degraded at 20,000 degrees per hour. (Rotation affects the measurement accuracy only by moving the lamps, and the motion during the time an observation takes is only 2 percent of the spacing between lamps.) However, if the chase vehicle does not rotate, the docking fixture must be designed to accept relative rotation between the two spacecraft.

Figures V-5 and V-6 show the performance of the three-light system with target spacecraft that rotate about their pitch and yaw axes. Again the limiting factor was the control system, not measurement accuracy. If a control system is to operate with target attitude rates over 700 degrees per hour above these axes, some form of attitude prediction will probably be needed. This can be implemented without changing the structure of the control system but will require more parameters in the onboard mathematical dynamics model described in Chapter VIII. It may also require an increase in the number of state variables used in the Kalman filter portion of the control system. The result will be an increase of approximately 20 percent in the computational burden on the flight computer. We do not believe this will be a major problem in system design.

ORIGINAL PAGE IS
OF POOR QUALITY

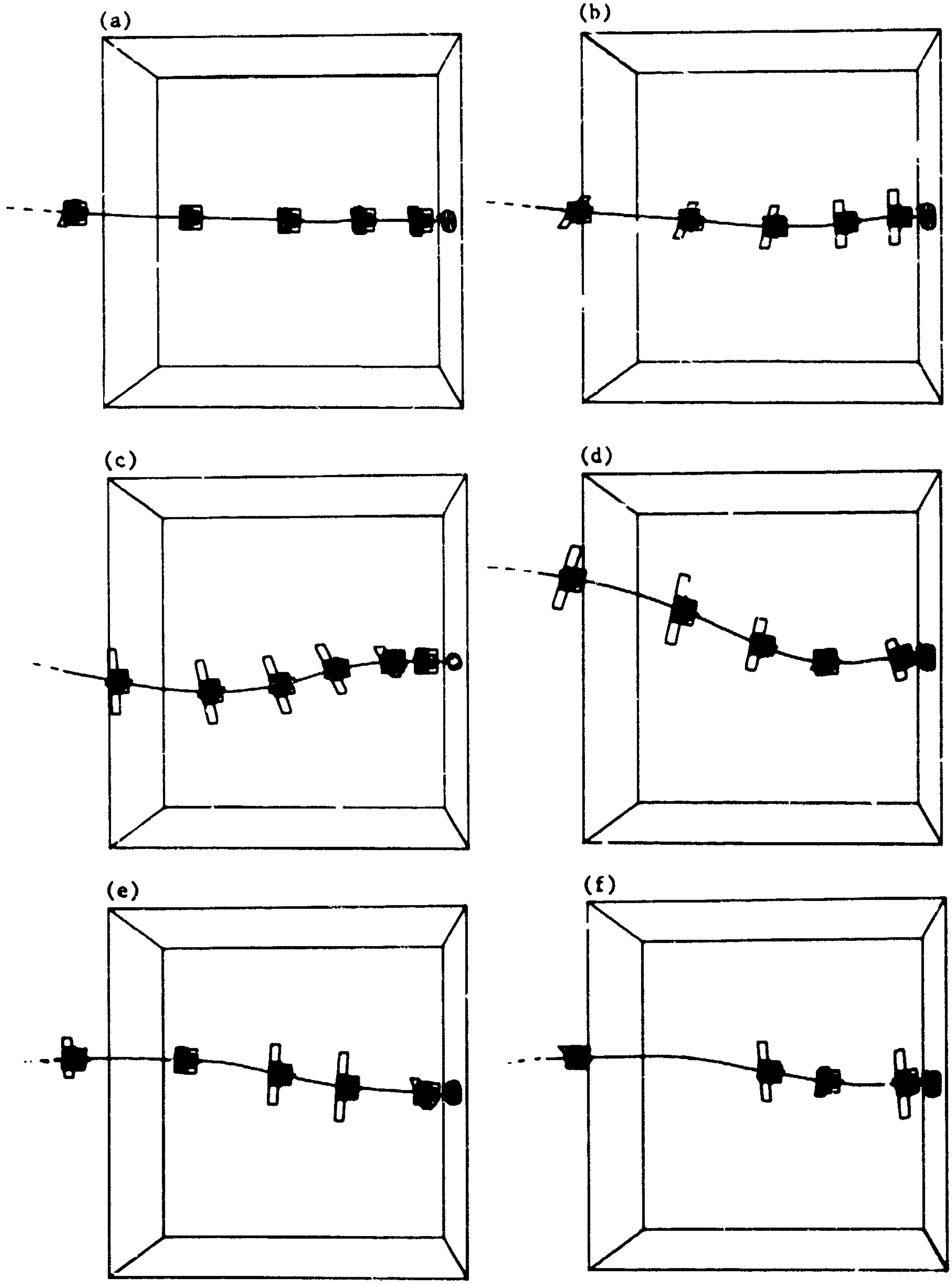


Figure V-4
Trajectories with three-light system with various target tumble rates about the docking axis: a) with 540°/hr; b) with 1000°/hr; c) with 4000°/hr; d) with 10,000°/hr; e) with 12,000°/hr; and f) with 20,000°/hr.

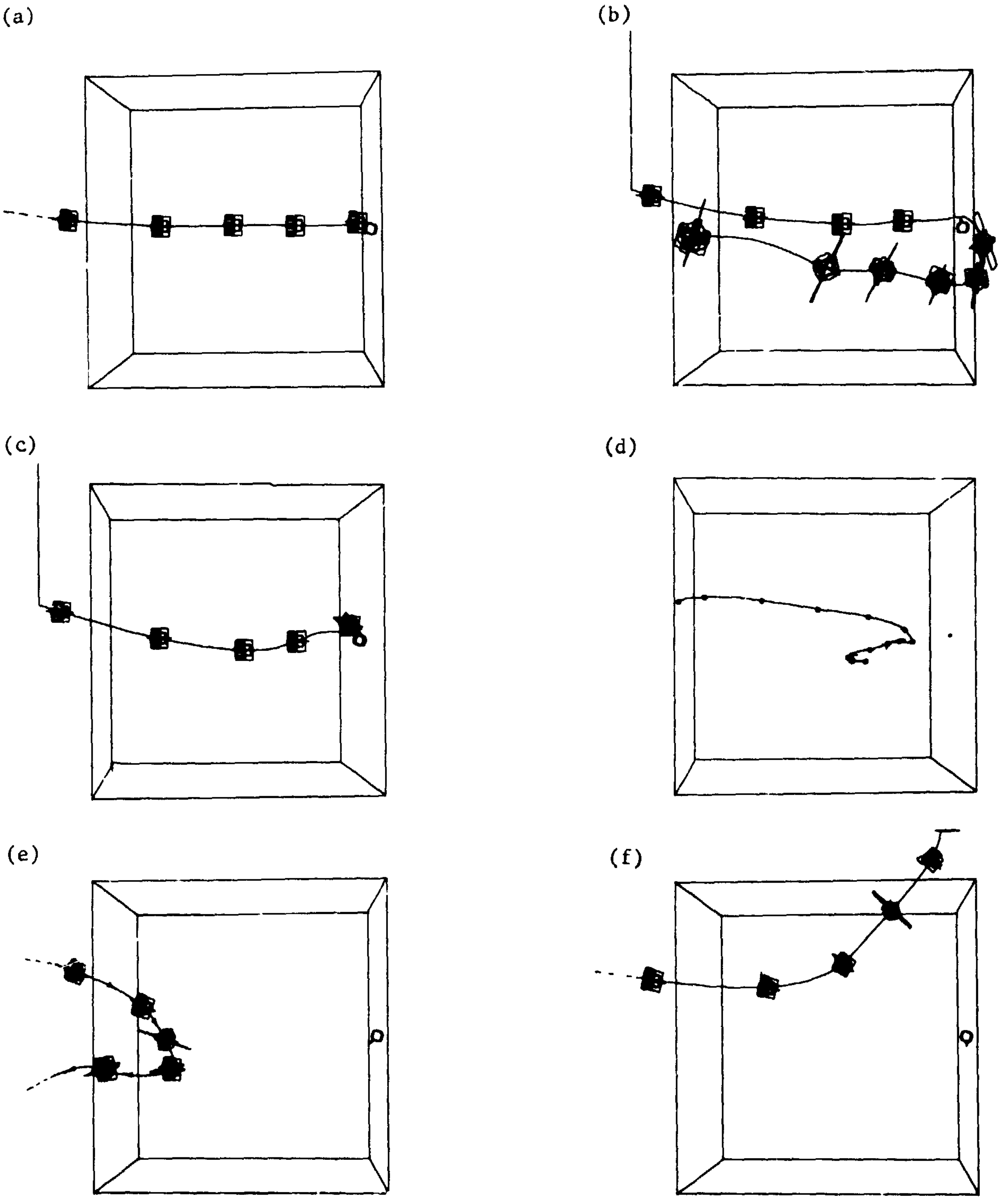


Figure V-5
Trajectories with three-light system with various target tumble rates about the pitch axis: a) with 540°/hr; b) with 1000°/hr; c) with 2000°/hr; d) with 4000°/hr; e) a closeup of d); and f) with 8000°/hr.

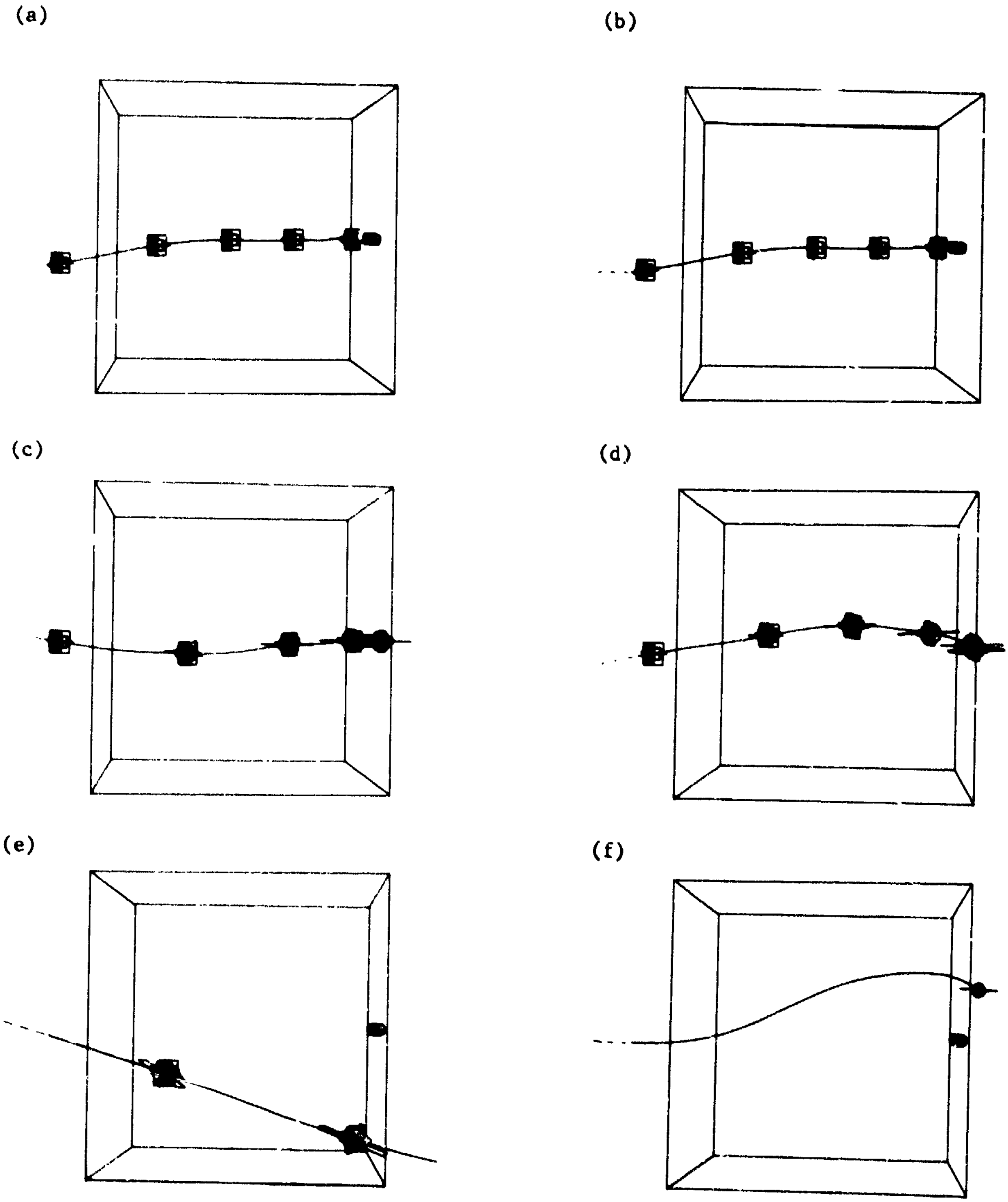


Figure V-6
Trajectories with three-light system with various tumble rates about the yaw axis: a) and b) are a stereo pair with 540°/hr tumble rate; c) with 1000°/hr; d) with 2000°/hr; e) with 4000°/hr; and f) with 8000°/hr.

C. THERE IS ONLY ONE OPTION

The selection of the best system does not require a tradeoff matrix because only the three-light system worked reliability. It is interesting to note, however, that the system would be selected even if the other systems had performed as well. Consider how the systems compare in the 10 characteristics summarized in Table V-1. It is better than the other two systems in almost every column of the table.

The table entries shown in the cost column are only for comparison and are based on the known costs of hardware of similar complexity. Although the figures do not include such indirect costs as software development and impact on other systems, these costs can be expected to vary roughly in proportion to the hardware costs. None of the systems requires the expensive packaging associated with high voltage, none requires a zoom lens, and all can operate in the red/near-infrared portion of the spectrum for which inexpensive sensors are available. All of the systems will require some form of iris or threshold control, and this control will have comparable complexity in all three systems. Because the color sensor and rainbow beacon of the "rainbow" system are the only elements in any of the systems based on totally untried technology, the "rainbow" system represents by far the greatest risk.

All three systems have an algorithm hazard in common: they must be able to determine when the target is not within the television cameras' field of view. When the distance to the target is great, the video signal amplitude will fall with the square of the distance to the target, and it may be difficult to determine whether the target is further away than expected or is out of the field of view. We recommend a physical simulation to demonstrate that a hardware system can solve this problem.

In addition to this common hazard, the "rainbow" and ring-of-lights systems have the problems discussed previously that caused them to perform poorly in the simulations.

The speed of the three-light system is set primarily by the video frame rate, and the figure shown in Table V-1 is based on standard video rates. A special-purpose camera might be developed for higher speed. The other two systems, in contrast, are limited in speed by more fundamental constraints. In both of these systems, two successive observations are combined to define target attitude. Enough time must be allowed between observations to ensure that the observations are significantly different from one another. Therefore these two systems can never achieve the speed of the three-light system.

Accuracy and television resolution are closely related. The resolution specified for the three-light system was based on simulations that showed the system works with resolution three times poorer. A factor of three safety margin was added to increase confidence while adding little to the cost; a 380-line resolution is approximately that of standard television cameras.

ORIGINAL PAGE IS
OF POOR QUALITY

Table V-1 Tradeoff Matrix for Three Rendezvous and Docking Systems

Criterion / System	Hardware Cost (1)	Algorithm Hazards (2)	Accuracy	Speed	Size and Weight (equip (4)	Power Required	Min/Max Range	Resolution Required	Ruggedness and Hardware Reliability	System Compatibility
3-Light System	$300,000	Moderate	3% at 20 Meters 47% at 300 Meters	Up to 10 Measurements per Second	$0.02\ m^3$ 5 kg	10-15 Watts	0-300m	380 Television Lines	Good	Good
Rainbow System	$500,000	Severe	4% at 20 Meters 7% at 300 Meters	Up to 1 Measurement per Second (3)	$0.04\ m^3$ 12 kg	25 Watts	0-300m	380 Television Lines	Fair	Fair
Ellipse System	$300,000	Severe	41% at 20 Meters 23% at 300 Meters	Up to 1 Measurement per Second (3)	$0.03\ m^3$ 8 kg	20 Watts	0-300m	380 Television Lines	Good	Severe Problems

(1) For Comparison Only - Based on Cost per kg of Similar Hardware

(2) See Text

(3) Constraint Imposed by Having to Take Two Measurements; If Measurements Are Too Close, Noise Will Be Dominant

(4) Chase Vehicle System

The resolution for the other two systems was specified at a level high enough to convince us that accuracy was not the dominant cause of failure. Simulations with much better resolution demonstrated no better performance.

All three systems are rugged, but the "rainbow" system uses a rainbow projector and a color detector that may require fragile parts and precise alignment. The types and the quantity of components make this system rate the lowest in terms of hardware ruggedness and reliability. The differences among the systems are so great in other respects, however, that this factor is insignificant.

All three systems place similar constraints on the system:

1) Nothing must be allowed to block the field of view of the cameras or color detector;

2) The propulsion system must be able to support the added mass of the guidance system;

3) The power system must provide power for the additional hardware:

4) The coarse rendezvous system must provide a sufficiently accurate estimate of the relative positions and velocities of the two spacecraft to give the video system time to search for the target, lock on, and begin to track.

Additional constraints are discussed in Chapter IV. However, of the three systems the three-light system imposes the fewest constraints; it has only one camera, and the camera does not have to be placed at the docking fixture, it requires the smallest amount of power and weighs the least. Even if the other systems worked well, they would place more constraints on the two spacecraft.

D. THE HARDWARE TECHNOLOGY IS AVAILABLE

The three-light system does not require any new developments in hardware to make it practical. It can use television cameras of standard resolution and speed operating in a convenient spectral band. The burden on the flight computer is modest and will not require a special computer design.

We do not recommend, however, that fully autonomous operation be tried with the first such system built. We recommend that the system be used first as an aid in remote piloting of the chase vehicle. As the system proves its ability, it can be given greater autonomy from mission to mission. This approach of gradually increasing autonomy minimizes risks and allows the collection of valuable data that can be used to improve system performance.

VI. Two Other Candidate Systems

VI. TWO OTHER CANDIDATE SYSTEMS

A. CORRELATION SYSTEM WAS TOO EXPENSIVE

In most correlation guidance schemes, a reference scene is compared to the live scene from a television camera. The center of the reference scene is placed at different locations on the live scene, and the correlator computes the quality of the match with each alignment. The system uses the coordinates that give the best match between scenes for horizontal and vertical error signals.

The rendezvous system cannot afford the time to do that because the misalignment can be in six degrees of freedom, not just two. The rendezvous system would have to match scenes with several different horizontal displacements for several individual vertical displacements, for each zoom lens setting for each of several rotations of the camera about the optical axis, for each of several reference scenes showing the target from different angles. The number of comparisons required for each measurement is astounding, perhaps ten billion at the very least, and each comparison may require 2000 or more operations.

The number of comparisons must be reduced, and this is possible. The guidance system can use the centroid-tracking and ellipse-fitting techniques described in Chapter IV to greatly reduce the number of degrees of freedom. The system we considered used these techniques to control the camera's zoom lens and gimbal set. This was done to keep the target image centered in the camera's field of view, to maintain a constant image size, and to keep the major axis of the best-fit ellipse horizontal. The eccentricity of the best-fit ellipse was to be used to select a subset of reference scenes for comparison. The correlator would then need only to select the reference scene that best matched the live scene.

Without question this system was the most complex of the five systems that were evaluated, though it was less complex than pattern interpretation systems and more tolerant of sensor imperfections. It had the advantage of requiring no special docking aids on the target spacecraft. With modifications to its database of comparison images, it could, in principle, be used with any target. However, the scheme could not compete with the other concepts in accuracy or cost.

B. CURVE-FITTING SCHEME DUPLICATED MSFC EFFORT

R. Dabney of NASA's George C. Marshall Spaceflight Center describes a video guidance scheme in MSFC memorandum ED15-81-71. His system used the same ring-of-lights docking aid as one of the systems described in Chapter IV, but its method for fitting an ellipse to the ring's image was considerably different, and a different approach was used to derive guidance information from ellipse parameters.

This system was not evaluated under this study, because more could be gained from the study of systems that had not already been analyzed and simulated, as this one had.

VII. Other Techniques That Were Investigated

VII. Other Techniques That Were Investigated

VII. OTHER TECHNIQUES THAT WERE INVESTIGATED

Table VII-1 summarizes the approaches that were considered for use in the video guidance system. Many of these are discussed in detail in Chapters IV and VI. This chapter discusses the rejected approaches.

Table VII-1 Techniques Investigated

- Interpreting an Image of Three Lights
- Fitting an Ellipse to the Image of a Ring of Lights
- Correlation
- Recognition of Corners and Edges
- Projected Pattern from Target
- Recognition of a Bar Pattern on the Target Spacecraft
- Detection of Target Color, which Varies with Observer's Position
- Stereo Rangefinding
- Stadimetry
- Rangefinding by Optical Focusing

A. CORNER/EDGE RECOGNITION IS TOO EXPENSIVE

Many of the studies on robotic vision have concentrated on the detection of edges and corners. While this approach holds promise in many robotics applications, it is not recommended for an autonomous video rendezvous system. There are two reasons:

1) Fast algorithms are easily fooled. The guidance system must contend with an image of extremely high contrast and with shadows that are almost totally black. Although contrast aids in the detection of corners and edges when the target's geometry is simple, the variable collection of instruments on the target spacecraft will result in a complex pattern of shadows that will make image interpretation very difficult.

2) Better algorithms cannot compete in speed or cost. Speed is vital when the system must operate with tumbling targets, and software analysis of a complex image cannot come close to the speed of the other algorithms considered. Further, the better algorithms store the entire image, which may require as much as a quarter of a million bytes of memory. A detailed cost analysis is not required to conclude that such a system will be far more expensive than one that uses one of the selected algorithms.

B. BAR PATTERN REQUIRES HIGH RESOLUTION AND GOOD LIGHTING

The Universal Product Code has been highly successful in automating checkout lines in supermarkets. In principle, a similar bar-code pattern could be used for locating key reference points on a spacecraft. Although this approach would greatly simplify the target, it greatly complicates the overall system. First, it requires good lighting to ensure that the bar patterns are visible. Because the target may be tumbling, sunlight will be unreliable. While the problem is not insurmountable (a backlit pattern might be used, for example), the approach remains more susceptible to lighting problems than the selected approaches.

The hardest problem to solve, however, is resolving the details of the pattern. The pattern must be small if it is to be effective for identifying reference points, and the small size makes it difficult to use, especially when the system must operate at all ranges from zero to over 300 meters. Other techniques such as those discussed in Chapter IV, circumvent these problems and provide the same capabilities at a lower cost.

C. RANGEFINDING BY OPTICAL FOCUSING IS HARD TO USE

Optical focusing, at first glance, appears to be a simple method for determining range, but a number of practical problems make it very difficult to implement:

1) The system does not know what is in focus. A practical guidance system needs to determine the range to a known reference point on the target spacecraft to adjust approach speed for a soft contact with the target. A simple focusing system attempts to adjust for maximum sharpness of the image as a whole, and there is no guarantee that a known reference point will be in sharp focus. While this is not a problem at great distances from the target, it may cause severe problems in deriving velocity at close range, and proper alignment of the docking fixture may require additional hardware that partially duplicates the function of the rangefinder.

2) A telephoto lens is required at long range. Lenses with wide fields of view and modest F numbers have great depth of focus; one lens setting may produce sharp focusing for any range from a few meters to infinity. A large-aperture telephoto lens is therefore required to make effective use of optical focusing. Such lenses, however, are expensive and their narrow fields of view introduce pointing problems. A gimballed pointing mount may be required, along with a complex state estimation algorithm to keep the target centered in the field of view of such a lens.

3) Automatic focusing is a noisy operation. Sharp focusing is detected by measuring the high-frequency content in the video signal,

because sharpest focusing is associated with the highest high-spatial-frequency content in an image. Finding the point of sharpest focus is analogous to finding a point of zero slope on a curve of sharpness versus lens setting. Both operations involve the calculus operation of differentiation. This operation is always "noisy" in that it emphasizes signal corruptions, and the double differentiation is especially noisy. The result is that an accurate system will be slow, expensive, or both.

Although it may be argued that accurate information is not required at great distances from the target, the simulations have shown that systems which are not accurate do not work well. The problem is that the guidance system cannot determine the approach rate. Therefore, it allows the chase vehicle to drift away from the target or drives the vehicle toward the target at such a high speed that it cannot stop.

VIII. A Generic Video Guidance System

VIII. A Generic Video Guidance System

VIII. GENERIC VIDEO GUIDANCE SYSTEM

The simulation programs in Appendices A thru C are adaptations of a single original program called DSIM, which was developed by Martin Marietta to test our autonomous video rendezvous guidance system. The program and the guidance system were developed under IR&D project D-11R. Because a major portion of this guidance system is common to all three simulations, a detailed explanation of the system is presented.

Figure VIII-1 is a block diagram of the control system, and each block corresponds, in general, with a subroutine or set of subroutines in the simulation programs. The exceptions to this rule are:

1) Telemetry uplink and downlink have not been implemented in these simulations;

2) Several subroutines simulate television images of the docking aid and process them to obtain position and target attitude. These subroutines, which differ among the simulation programs, were written to perform specific functions rather than to correspond to specific hardware modules in the block diagram;

3) Subroutine DOCK in each simulation is the "wiring" among the blocks; it sequences operations and passes data among blocks.

Table VIII-1
Relationship between Block-Diagram Blocks and Subroutines in the Simulation Programs

Block Name	Subroutines*
Kalman Filter	INCORP, (COMPG, ESTCOV, KALGAN, UPDSTA, UPDCOV)
Mathematical Dynamics Model	PROPES, RPY, ESTRPY
Inertial Measurement Unit	IMU, (DIRMAT, ANGVEC)
Goal-Setting Logic	SETGOL
Control Law	THRUST, (CNTLAW, ACCEL, FIRTHR)
Thrusters	SELECT, (TABLE1, TABLE2, TABLE3)
Chase Vehicle Dynamics	PROPTR, (COMPK1, COMPK2, COMPK3, COMPK4, POINT, STPRIM, ANGVEC, LINACL, MPRIME, FORCE, DIRMAT, TORQUE, LPRIME, MAKROT, QPRIME)
Target Spacecraft Dynamics	TRGATT

* Library routines such as matrix arithmetic routines are not listed.

ORIGINAL PAGE IS
OF POOR QUALITY

Table VIII-1 shows the correspondence between program subroutines and blocks in the block diagram. Subroutine names are shown in parentheses if they are not the primary routines involved in implementing a block. These routines are called on by the primary routines to implement low-level details of a function.

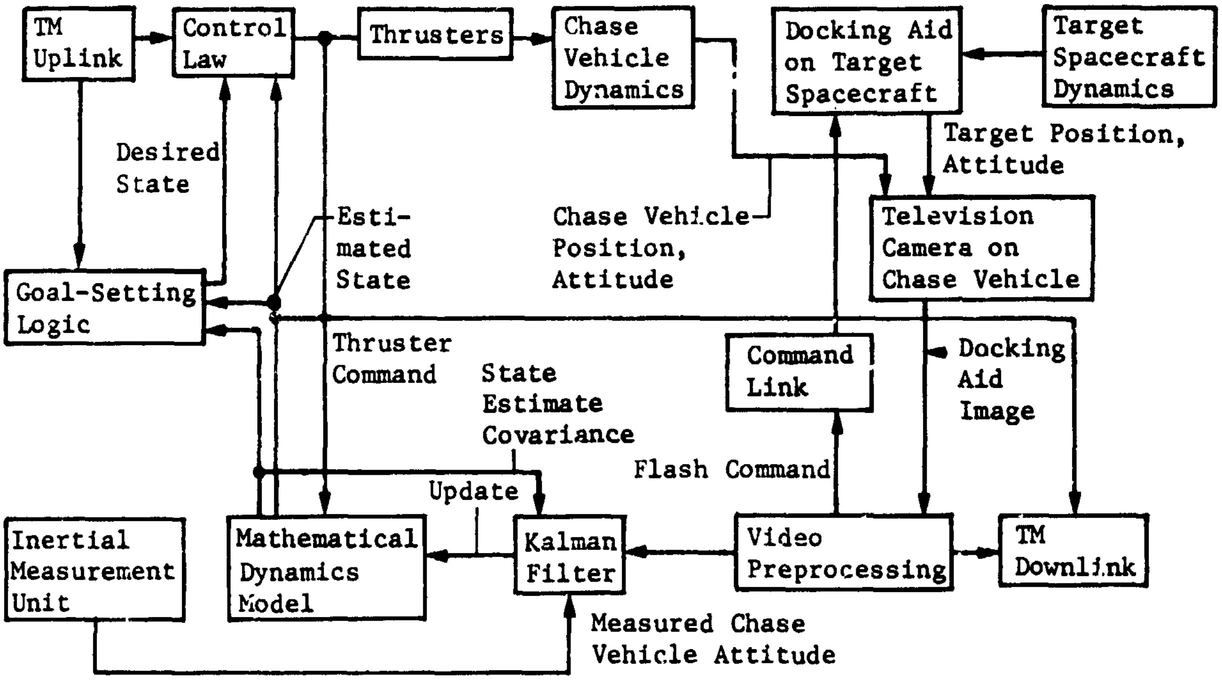


Figure VIII-1 Genera Video Rendezvous Guidance Systems

Only Major Signal Paths Are Shown. The Simulations Reported Here Do Not Use TM Uplink and Downlink, and Only Minimal Forms of the Kalman Filter, Mathematical Model, and Goal-Setting Logic Were Implemented.

A. THE KALMAN FILTER IMPROVES ACCURACY AND DERIVES RATES

The Kalman filter combines observations to fine-tune the parameters in a mathematical dynamics model, providing accurate estimates of position and velocity. In these simulations, the filter adjusts six parameters, the x, y, and z components of position and velocity of the chase vehicle relative to the target. It can also be expanded to estimate such other parameters as target tumbling characteristics.

Although a mathematical dynamics model is an essential feature of a Kalman filter, the model is considered to be a separate entity in this control system for two reasons:

1) The model would be used in the control system even if the filter were not. The primary reason the system maintains the model is to allow it to operate properly even when feedback is interrupted by obstruction of the camera's view of the target. The model is not simply a piece of the Kalman filter;

2) The model contains parameters that are not adjusted by the filter. The parameters include the latest chase vehicle attitude quaternion and angular velocity vector from the inertial measurement unit, the latest target attitude measurement, information about the geometry of both spacecraft, an estimate of the mass of the chase vehicle, optical system parameters, and other data.

The subroutine that implements the Kalman filter is called INCORP. Each time a new measurement is taken, this subroutine updates the state estimate ($\underline{x}$) and the state covariance matrix (P) by calculating

$$K \leftarrow PG^T (R+GPG^T)^{-1} \quad , \qquad \text{(VIII-1)}$$

$$\underline{x} \leftarrow \underline{x}+K(\underline{z}-\underline{g}) \quad , \qquad \text{(VIII-2)}$$

$$P \leftarrow (I-KG)P \quad , \qquad \text{(VIII-3)}$$

where

K, represented in the programs as KGAIN, is a 6x3 matrix referred to as the Kalman gain matrix,

P, represented in the program as P, is a 6x6 matrix, the state covariance matrix,

G, represented in the programs as G, is a 3x6 matrix, the partial derivative of the predicted measurement with respect to the state. Since the measurements in all three simulations are the first three elements of the state vector, G is a constant with the value

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

It is, however, "computed" in subroutine COMPG to maximize the usefulness of the program for other simulations in which G is not constant,

R, represented in the programs as R, is a 3x3 matrix, the measurement covariance matrix. This matrix is calculated from an empirical formula in subroutine ESTCOV. The formula was derived by observing the measurement errors at various ranges in several rendezvous simulations and fitting a curve to a graph of the mean square error versus range. In the design of a flight system a similar approach could be used, but test data and analytically derived tolerances would be used instead of simulation results,

$\underline{x}$, represented in the programs as ESTATE, is the estimated chase vehicle state. This is a six-element vector, the first three elements of which represent position along the x, y, and z axes of the so-called "primary" reference frame. The remaining elements are the velocity components along these axes. (The primary frame is a nonrotating right-handed rectangular coordinate system centered at the target spacecraft's center of mass but aligned with the body axes of the chase vehicle at the instant the video guidance system takes control),

$\underline{z}$, represented in the programs as CVPOS, is a three-element measurement vector representing the measured position of the chase vehicle's center of mass in the primary reference frame,

$\underline{g}$, which is not explicitly represented in the programs, is the predicted observation. In these implementations of the control system, $\underline{g}$ is simply the first three elements of $\underline{x}$,

I, the 3x3 identity matrix, is not explicitly represented in the programs.

The Kalman filter form used here does not have the best roundoff and stability characteristics for flight software, but is ideal for simulations because it is easy to modify.

B. THE MATHEMATICAL DYNAMICS MODEL ALLOWS DEAD RECKONING

The mathematical model embodies the guidance system's knowledge of the target spacecraft and chase vehicle. Some of this information--positions of the cameras, docking aids and fixtures with respect to the spacecraft centers of mass, the optical focal lengths, the thruster locations, orientations and thrust levels, and similar information--is known in advance. In the simulations, this information is passed to all subroutines that need it through common blocks. All these common blocks appear in subroutines INIPAR, where the variables are initialized and the variables' definitions, which vary from one simulation to the next, are given in comments with the type declarations in that subroutine.

A second class of information in the mathematical model is measured data that does not pass through the Kalman filter. Although the nature and use of this information vary among the three simulations, all three programs maintain target attitude, chase vehicle attitude, and the coordinates of the lamp images' centers of brightness.

The third class of information, which is identical in all three simulations, is the information processed by the Kalman filter. In the three simulations this class included only the position and velocity of the chase vehicle. These collectively referred to as the state estimate (ESTATE) and the state estimate covariance matrix (P). Additional parameters could be added to improve performance with tumbling targets.

In addition to a data base, the mathematical model has procedures for updating and using the data. The largest of these is subroutine PROPES, which uses numerical integration to propagate the state estimate and covariance matrix between observations. The formulas it uses are

$$\underline{\ddot{x}} = A_c^T K_2 \, \underline{f} \, / \, m \qquad \text{(VIII-4)}$$

$$\underline{\Delta x} = (\underline{\dot{x}} + \tfrac{1}{2} \, \underline{\ddot{x}} \, \Delta t) \, \Delta t \qquad \text{(VIII-5)}$$

$$\underline{\Delta \dot{x}} = \underline{\ddot{x}} \Delta t \qquad \text{(VIII-6)}$$

$$\Delta P = (FP + PF^T + NVN^T)\Delta t \qquad \text{(VIII-7)}$$

where

$\underline{x}$, $\underline{\dot{x}}$ and $\underline{\ddot{x}}$ represent the position, velocity, and acceleration vectors, respectively. The position vector is the first three elements of the array ESTATE, and the velocity vector is the second three elements. The program variable name for $\underline{x}$ is ACCEL,

A is the transpose of the chase vehicle direction cosine matrix, as measured by the inertial measurement unit, which is modeled in subroutine IMU. The variable name in the program is ACVT,

K2 is a constant matrix relating the force magnitudes of all the thrusters to the force vectors they produce. The variable name in the programs is AK2,

$\underline{f}$ is a 14-element array in which each element corresponds to the magnitude of the force currently produced by one thruster. The programs refer to this array as F,

m is an estimate of the chase vehicle mass. The programs refer to this variable as AVGMAS,

t is the time interval over which the state estimate is to be propagated. The variable is referred to in the programs as STEP,

P is the state estimate covariance matrix. The correspo ng variable name in the programs is P,

F is the partial derivative of d(ESTATE)/dt with respect to ESTATE, which, in this context, is the constant matrix.

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

This matrix does not appear explicitly in the programs because its effect is simply to select elements of P to be added together. The selection and addition were done more efficiently by avoiding matrix arithmetic,

NVN^T is a matrix that gives the covariance of stray (unmodeled) accelerations of the chase vehicle. These accelerations arise from uncertainty in thruster force magnitude from attitude changes between the time A^T is measured and the time it is used, from gravity gradient acceleration, from thruster misalignment, and from roundoff errors in computation. Because this matrix is sparse (only three elements out of 36 are nonzero), it is handled without using matrix arithmetic and is not explicitly represented in the programs.

The mathematical model is also responsible for attitude control. Subroutine RPY is used if the docking aid is within the field of view. It returns the changes in roll, pitch, and yaw required to align the chase vehicle's x axis with the line of sight between the camera and the docking aid. Subroutine ESTRPY is used when the docking aid cannot be seen. It estimates the changes in roll, pitch, and yaw required to align the chase vehicle's x axis with the line connecting the chase vehicle's center of mass and the target spacecraft's center of mass. ESTRPY operates on the state estimate (ESTATE) whereas RPY uses the coordinates of the center of brightness of the docking aid's image. ESTRPY is the same in all three simulations, but RPY is slightly different in the three versions because of the differences in the docking aids among the three systems.

C. THE INERTIAL MEASUREMENT UNIT PROVIDES ATTITUDE AND ATTITUDE RATE INFORMATION

The guidance system uses the so-called "primary" reference frame to maintain estimates of relative position and velocity, and of the attitudes of both spacecraft. The primary frame is a nonrotating coordinate system that is initially aligned with the chase vehicle's body axes at the instant the video guidance system takes control. The center of the coordinate system is at the target spacecraft's center of mass. Since both spacecraft rotate with respect to this frame, the guidance system must measure the chase vehicle's attitude with each video measurement to properly interpret the imagery. Subroutine IMU supplies simulated attitude measurements from the inertial measurement unit for this purpose. The attitude is returned in the form of a direction cosine matrix defining the attitude with respect to the primary frame.

The transpose of the direction cosine matrix and an angular velocity vector, used for control system damping, are also returned from the IMU.

The attitude is determined by examining the "true" state of the chase vehicle, which is maintained in the 14-element array STATE. Elements 10 through 13 of this array are the current attitude quaternion and define the attitude with respect to the "truth" coordinate system. Subroutine IMU subtracts the initial attitude from the current attitude to compute attitude with respect to the "primary" reference frame used for control. It then converts from quaternion notation to direction cosine notation.

The angular velocity vector is found from elements 7 through 9 of STATE, which are the angular momentum vector, and from element 14 of STATE, which is the chase vehicle mass. The equations are

$$I = I_o + (m-m_o)dI/dm \qquad \text{(VIII-8)}$$

$$\underline{\omega}_b = I^{-1} A_c \underline{L} \qquad \text{(VIII-9)}$$

where

I, represented in the programs as the 3x3 matrix INERTA, is the total chase vehicle inertia,

I_o, represented in the programs as the 3x3 matrix INERCV, is the inertia of the chase vehicle without fuel,

m, m_o, are the mass of the chase vehicle with the current fuel load and the mass with no fuel, respectively. In the programs m is represented as STATE (10), and m_o is MEMPTY,

dI/dm, represented in the programs by the 3x3 matrix FULDIS, is the amount of additional inertia added for each unit of fuel,

represented in the programs by the three-element array ATRATE (in subroutine IMU) or BODVEL (in subroutine ANGVEC), is the angular velocity vector expressed in the chase vehicle's body coordinate system,

$\underline{L}$, represented in the programs by elements 7 through 9 of STATE, is the angular momentum vector of the chase vehicle, expressed in the "truth" coordinate system,

A_c, represented in the program by the 3x3 matrix TACV, is the direction cosine matrix that defines the current chase vehicle attitude with respect to the "truth" coordinate system.

In these simulations the IMU returns exact values for attitude and angular momentum. Although it would be easy to add corruptions to these quantities, this has not been done because the inertial measurement unit of a real spacecraft can be very accurate for the few minutes a rendezvous operation requires, and IMU errors will not be a major contributor to the total error.

D. GOAL-SETTING LOGIC PROVIDES INTELLIGENCE

The goal-setting logic is a software module in the flight computer and is responsible for planning, safety, and hazard recognition. For example, it examines the state estimate and covariance matrix provided by the mathematical dynamics model, decides how close it should come to the target, and where its aim point should be with respect to the current position. The simulations use a minimal version of this software;

a practical flight system might also decide what trajectory will minimize fuel use, determine if the target is tumbling so fast that docking is impossible, or decide to ask for human assistance via telemetry.

The subroutine that embodies the goal-setting logic is SETGOL. The three simulation programs use slightly different versions of this routine because the locations of lights and cameras differ among the simulations. The essential logic, however, is identical.

First, the subroutine picks a desired location for the docking fixture of the chase vehicle. This point (Fig. VIII-2) will be just beyond the end of the target spacecraft's docking fixture. It can be expressed in target coordinates as $\underline{r} = \underline{h}_{dt} - \underline{d})$, where $\underline{h}_{dt}$ is the location of the end of the target docking fixture with respect to the target's center of mass and $\underline{d} = (d, 0, 0)^T$ is a safety margin.

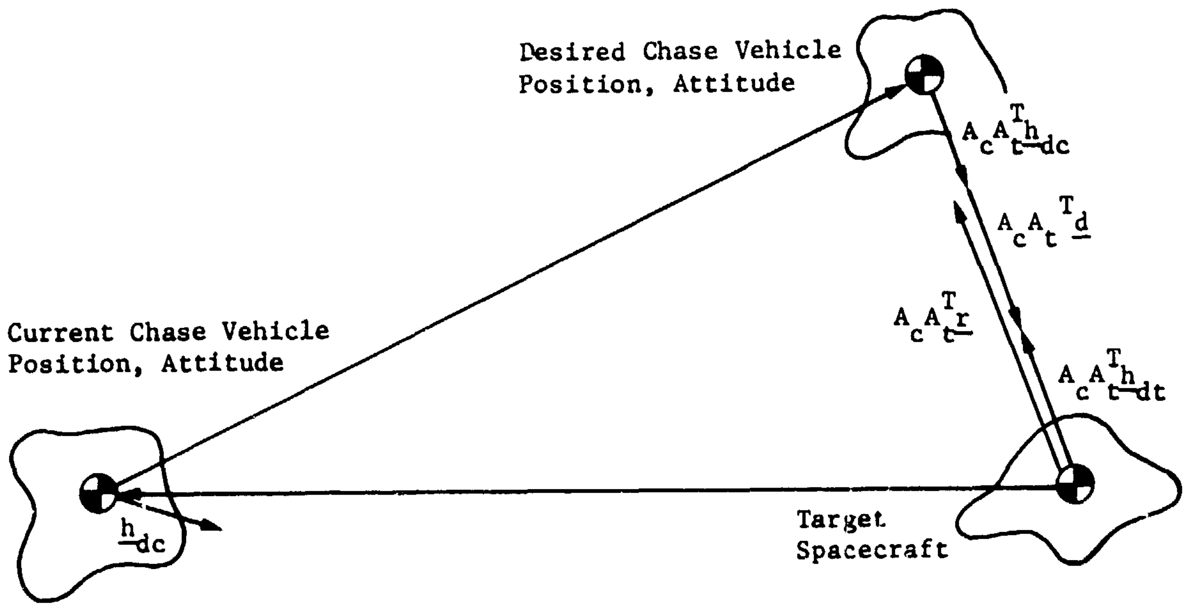


Figure VIII-2 Physical Significance of Equation (8-10) Multiplications by A_c and A_t^T Are Done to Convert All Vectors to the Chase Vehicle's Current Body Coordinate System. Attitude Control, a Separate Operation, Keeps $\underline{h}_{dc}$ Pointed Toward the Target So That, by the Time the Chase Vehicle Reaches Its Goal the Attitude Is Correct.

The value of d is computed from the state estimate covariance matrix (P) and has a value of approximately three times the standard deviation of the position estimate. If the chase vehicle is estimated to be more than 15 meters from the target's x axis, d is increased to encourage an approach along the x axis.

When range and velocity are sufficiently small, d is set to zero. If this were not done, the chase vehicle might hover indefinitely a few centimeters from contact.

After $\underline{r}$ has been determined, the subroutine finds a goal for the chase vehicle's center of mass, expressed in chase vehicle coordinates, with the formula

$$\underline{v} = A_c \, (A_t^T(\underline{r} - \underline{h}_{dc}) - \underline{x}) \qquad \text{(VIII-10)}$$

where

$\underline{v}$, represented in the programs as V3, is the goal for the chase vehicle's center of mass,

A_c, represented in the programs as ACV, is the chase vehicle's attitude direction cosine matrix, as measured by the inertial measurement unit,

A_t, represented in the programs as TRNAT, is the transpose of the target's attitude direction cosine matrix, which is derived from the appearance of the target in the television images,

$\underline{x}$, represented in the programs as the first three elements of ESTATE, is the chase vehicle's estimated position in the "primary" reference frame,

$\underline{h}_{dc}$, represented in the programs as HDC, is the location of the chase vehicle's docking fixture with respect to the chase vehicle's center of mass.

Finally, the subroutine defies a "box" or region of space around the goal. The control law will use this box in selecting thrusters. If the chase vehicle is already within the box, the control law will attempt to prevent it from leaving. If it is not within the box, the control law will attempt to force it to enter the box before a deadline time has been reached. If the chase venicle is within the box and is not in danger of immediately drifting out of the box, the control law will not activate any thrusters.

The box concept has two advantages. First, it allows the control law to work on a single axis at a time and to ignore any interaction. Second, it allows control tolerances to be adjusted as a function of the distance to the target. This prevents waste of fuel at greater distances from the target where accurate positioning is not important. The subroutines implement only a very simple version of the box-sizing strategy.

E. THE CONTROL LAW DETERMINES IDEAL THRUST VECTORS

The control law, subroutine CNTLAW, converts the velocity estimate (elements 4 through 6 of ESTATE) to the chase vehicle body coordinates and then calls on the function ACCEL six times, once for each translational and rotational axis. ACCEL returns a value of zero or ± 1.0

ORIGINAL PAGE IS
OF POOR QUALITY

to indicate the state of an idealized thruster (off, on with positive thrust, or on with negative thrust). The value is selected to ensure that limit cycling along the axis the thruster controls is confined to the "box" defined by the goal-setting logic. The algorithm behind ACCEL is difficult to state in plain English, but the "pseudocode" flowchart in Figure VIII-3 shows the essential logic. For a given set of input parameters, ACCEL defines a simple phase-plane control law. The input parameters have the effect of adjusting the phase-plane decision boundaries to accommodate different thrust authorities, decision intervals, limit cycle envelopes and control bandwidths.

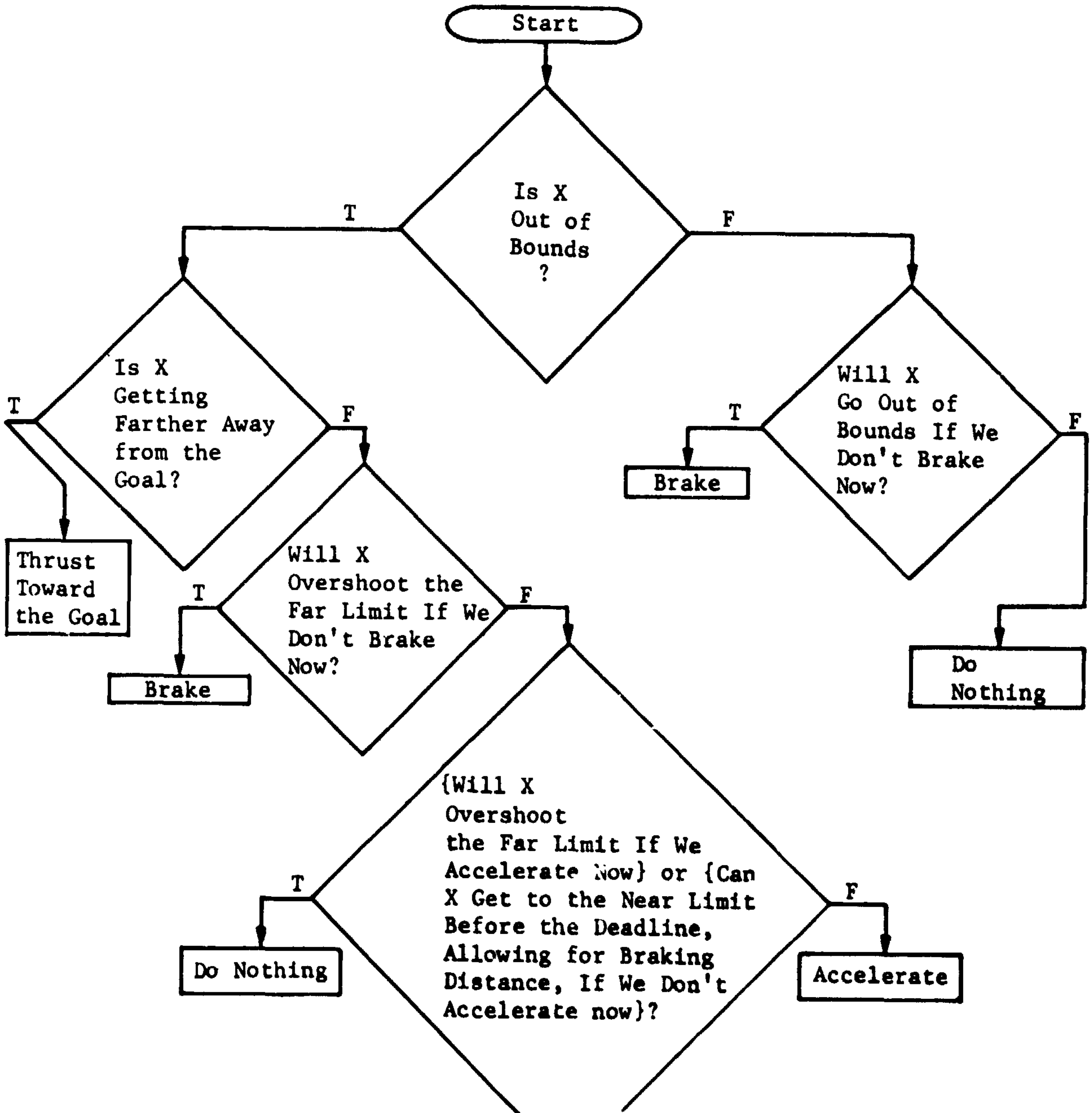


Figure VIII-3 Essential Logic of Function ACCEL
The Logic of the Subroutine is Complicated Slightly by the Fact That The Thrust Directions Required to Implement "Brake," "Accelerate," and "Thrust Toward Goal" Depend on the Current Position and Velocity.

After $\underline{r}$ has been determined, the subroutine finds a goal for the chase vehicle's center of mass, expressed in chase vehicle coordinates, with the formula

$$\underline{v} = A_c \, (A_t^T(\underline{r} - \underline{h}_{dc}) - \underline{x}) \qquad \text{(VIII-10)}$$

where

$\underline{v}$, represented in the programs as V3, is the goal for the chase vehicle's center of mass,

A_c, represented in the programs as ACV, is the chase vehicle's attitude direction cosine matrix, as measured by the inertial measurement unit,

A_t, represented in the programs as TRNAT, is the transpose of the target's attitude direction cosine matrix, which is derived from the appearance of the target in the television images,

$\underline{x}$, represented in the programs as the first three elements of ESTATE, is the chase vehicle's estimated position in the "primary" reference frame,

$\underline{h}_{dc}$, represented in the programs as HDC, is the location of the chase vehicle's docking fixture with respect to the chase vehicle's center of mass.

Finally, the subroutine defies a "box" or region of space around the goal. The control law will use this box in selecting thrusters. If the chase vehicle is already within the box, the control law will attempt to prevent it from leaving. If it is not within the box, the control law will attempt to force it to enter the box before a deadline time has been reached. If the chase vehicle is within the box and is not in danger of immediately drifting out of the box, the control law will not activate any thrusters.

The box concept has two advantages. First, it allows the control law to work on a single axis at a time and to ignore any interaction. Second, it allows control tolerances to be adjusted as a function of the distance to the target. This prevents waste of fuel at greater distances from the target where accurate positioning is not important. The subroutines implement only a very simple version of the box-sizing strategy.

E. THE CONTROL LAW DETERMINES IDEAL THRUST VECTORS

The control law, subroutine CNTLAW, converts the velocity estimate (elements 4 through 6 of ESTATE) to the chase vehicle body coordinates and then calls on the function ACCEL six times, once for each translational and rotational axis. ACCEL returns a value of zero or ± 1.0

to indicate the state of an idealized thruster (off, on with positive thrust, or on with negative thrust). The value is selected to ensure that limit cycling along the axis the thruster controls is confined to the "box" defined by the goal-setting logic. The algorithm behind ACCEL is difficult to state in plain English but the "pseudocode" flowchart in Figure VIII-3 shows the essential logic. For a given set of input parameters, ACCEL defines a simple phase-plane control law. The input parameters have the effect of adjusting the phase-plane decision boundaries to accommodate different thrust authorities, decision intervals, limit cycle envelopes and control bandwidths.

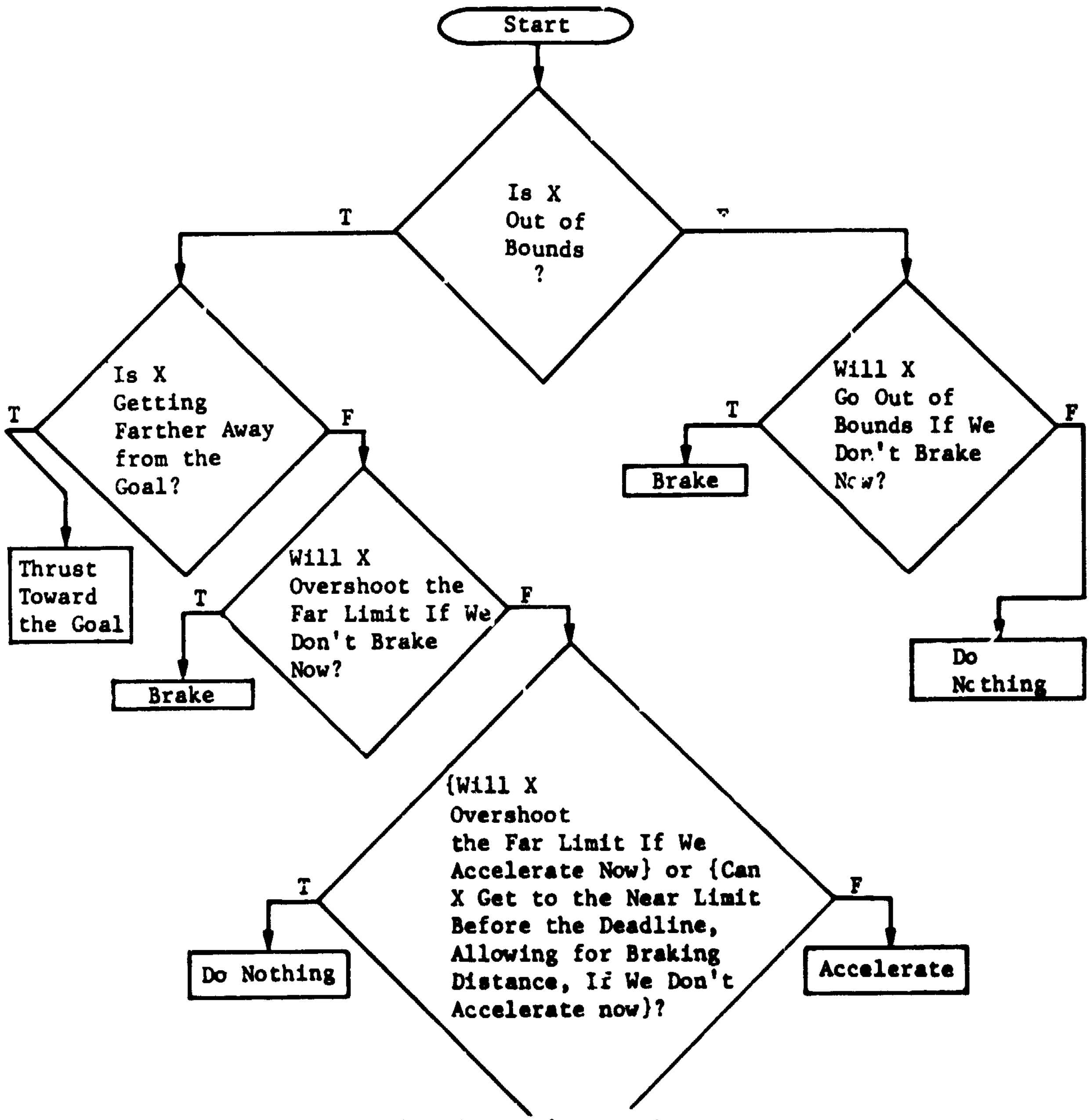


Figure VIII-3 Essential Logic of Function ACCEL
The Logic of the Subroutine is Complicated Slightly by the Fact That The Thrust Directions Required to Implement "Brake," "Accelerate," and "Thrust Toward Goal" Depend on the Current Position and Velocity.

F. THRUSTERS APPROXIMATE REQUIRED FORCES AND TORQUES

A practical spacecraft has a limited number of thrusters and cannot command an arbitrary acceleration in an arbitrary direction. The simulation models this restriction by giving the chase vehicle only 14 thrusters. (Physically there are 15, but two of them are controlled as a unit.) Combinations of the 14 thrusters can be selected to provide accelerations in any combination of the three translational axes and torques about any combination of these axes. The magnitudes of the accelerations and torques may, however, significantly differ from the ideal values. The logic in the control law accounts for this fact by knowing both the maximum and the minimum possible value for each acceleration and torque.

There are three possible commands (-1, 0, and +1) for each axis, and there are six axes for which commands can be given (x, y, z, yaw, roll, and pitch). This means there are only 3^6 or 729 different command combinations that can be given. Subroutine SELECT converts the set of commands to an index and then looks up an appropriate set of thrusters in a table. In flight hardware this could be done by using read-only memory to convert acceleration commands to thruster commands.

G. CHASE VEHICLE DYNAMICS RESPOND TO THRUSTER COMMANDS

The simulation programs use a set of five coupled differential equations to compute the chase vehicle's response to the thrusters. These equations propagate the vehicle's so-called "true state", which includes position, velocity, angular momentum, attitude, and mass. Because some of these quantities are vectors, the number of elements in the state is 14. These quantities are represented in the programs by the array STATE, whose elements are defined in Table VIII-2.

The array STATE differs significantly from the array ESTATE, the state estimate used for navigation. ESTATE contains only position and velocity estimates and therefore has only six elements. These six elements are not simply estimates of the first six elements of STATE because two different coordinate systems are used.

The true state is measured with respect to a nonrotating coordinate system centered at the target spacecraft's center of mass. The orientation of this coordinate system is fixed at the instant the simulation begins, with the +z axis pointing away from the center of the earth, the +y axis aligned with the orbit angular momentum vector, and the +x axis defined by the cross product of unit vectors along the +y and +z axes. This orientation was chosen to simplify the calculation of gravity gradient accelerations.

ORIGINAL PAGE IS
OF POOR QUALITY

Table VIII-2
Definitions of Elements of True-State Array STATE

Element Number	Definition
1	X Position (m)
2	Y Position (m)
3	Z Position (m)
4	X Velocity (m/s)
5	Y Velocity (m/s)
6	Z Velocity (m/s)
7	Angular Momentum about X Axis (kg m^2/s)
8	Angular Momentum about Y Axis (kg m^2/s)
9	Angular Momentum about Z Axis (kg m^2/s)
10	q_1 of Attitude Quaternion
11	q_2 of Attitude Quaternion
12	q_3 of Attitude Quaternion
13	q_4 of Attitude Quaternion
14	mass (kg)

The state estimate (ESTATE) uses a different coordinate system to demonstrate that the guidance system has no need for the target-orbit information that defines the truth coordinate system. In a flight situation the guidance system may, in fact, have no access to this information.

The differential equations, summarized in Table VIII-3, are integrated in subroutine PROPTR with fourth-order Runge-Kutta numerical integration. This integration algorithm calculates the state at a future time from the current state and the first derivative of the state at various points within the integration interval. Since a minimum of four evaluations of the first derivative are used for any interval, a separate subroutine, STPRIM, is used to calculate the derivatives. This subroutine, in turn, calls subroutines that evaluate the equations in Table VIII-3. Table VIII-4 shows the meanings of the symbols in the formulas and the names of the variables used in the simulation program to represent them.

The orbit is modeled as a 300-kilometer circular earth orbit. Because of the choice of coordinate system, the orbit has only a minor second-order effect; it determines the magnitude and direction of the gravity gradient acceleration, which is a relative acceleration between the two spacecraft caused by the fact that gravity acts most strongly on the spacecraft closest to the earth. The maximum magnitude of this differential acceleration is approximately 0.0001 g.

ORIGINAL PAGE IS
OF POOR QUALITY

Table VIII-3 Major Equations for True-State Computation

Equation		Subroutine Where Computed
$\underline{f} = A^T K_2 \underline{F}$	(VIII-11)	FORCE
$\dot{m} = -k_1^T \underline{F}$	(VIII-12)	MPRIME
$\underline{\dot{v}} = \underline{a}_{gg} + \underline{f}/m$	(VIII-13)	LINACL
$\underline{\omega}_b \equiv \begin{bmatrix} \omega_{b_1} \\ \omega_{b_2} \\ \omega_{b_3} \end{bmatrix} = I^{-1} AL$	(VIII-14)	ANGVEC
$\Omega = \begin{bmatrix} 0 & \omega_{b_3} & -\omega_{b_2} & \omega_{b_1} \\ -\omega_{b_3} & 0 & \omega_{b_1} & \omega_{b_2} \\ \omega_{b_2} & -\omega_{b_1} & 0 & \omega_{b_3} \\ -\omega_{b_1} & -\omega_{b_2} & -\omega_{b_3} & 0 \end{bmatrix}$	(VIII-15)	MAKROT
$\underline{N} = A^T K_3 F$	(VIII-16)	TORQUE
$\underline{\dot{L}} = \underline{N} - (A^T \underline{\omega}_b) \times L$	(VIII-17)	LPRIME
$\underline{\dot{q}} = 0.5\ \Omega\ \underline{q}$	(VIII-18)	QPRIME
$\underline{\dot{x}} = \underline{v}$	(VIII-19)	STPRIM

ORIGINAL PAGE IS
OF POOR QUALITY

Table VIII-4 Definition of Variables in Table VIII-3

Algebraic Symbol	Dimension	Program Variables	Units	Coordinate System	Meaning
A, A^T	3 x 3	A, TRNA	--	Truth	Direction Cosine Matrix Corresponding to $\underline{q}$
$\underline{f}$	3 x 1	NTFORC	N	Truth	Net Force Vector from Thrusters
$\underline{F}$	14 x 1	$\underline{F}$	N	--	Element i is the Magnitude of Thrust Produced by Thruster i
I, I^{-1}	3 x 3	INERTA, INVIN	$kg \cdot m^2$, $kg^{-1} \cdot m^{-2}$	Body	Moment of Inertia and Its Inverse
dI/dm	3 x 3	FULDIS	m^2	Body	Sensitivity of I to Additional Fuel Mass
k_1^T	1 x 14	AK1	$kg \cdot N^{-1} s^{-1}$	--	Element i is Fuel Mass Consumed, per Newton-Second of Thrust, for Thruster i
K_2	3 x 14	AK2	--	Body	Column i is a Unit Vector in the Direction of the Thrust Produced by Thruster i
K_3	3 x 14	AK3	m	Body	Column i is a Vector Corresponding to Thruster i. Its Direction is the Direction of the Torque Produced by the Thruster. Its Magnitude is the Amount of Torque per Newton of Force.
$\underline{L}$, $\dot{\underline{L}}$	3 x 1	STATE(7) (et seq.), DSTATE(7) (et seq.) DLDT	$kg \cdot m^2/s$, $kg \cdot m^2/s^2$	Truth	Angular Momentum about Center of Mass and its Rate of Change

Table VIII-4 (concl)

Algebraic Symbol	Dimension	Program Variables	Units	Coordinate System	Meaning
$m, \dot{m}$	Scalar	STATE(14), DSTATE(14) or DMDT	kg, kg/s	--	Total Chase Vehicle Mass and its Rate of Change
$\underline{N}$	3 x 1	N	N·m	Truth	Torque about Center of Mass
$\underline{q}, \dot{\underline{q}}$	4 x 1	STATE(10) (et seq), DSTATE(10) (et seq) or QPRM	--, s^{-1}	Truth	Allitude Quaternion and its Rate of Change
$\underline{v}, \dot{\underline{v}}$	3 x 1	STATE(4) (et seq), DSTATE(4) (et seq) or ACCEL	m/s, m/s^2	Truth	Velocity and its Rate of Change
$\underline{x}, \dot{\underline{x}}$	3 x 1	STATE(1) (et seq), DSTATE(1) (et seq)	m, m/s	Truth	Position and its Rate of Change
$\underline{\omega}_b$	3 x 1	BODVEL	s^{-1}	Body	Angular Velocity
Ω	. 4	OMEGA	s^{-1}	Body	Matrix Formed from Elements of $\underline{\omega}_b$ to Implement Equation (D-6)

H. TARGET ATTITUDE IS MODELED AS DETERMINISTIC

Subroutine TRGATT computes the target spacecraft's attitude as a function of time. In the simulations several versions of this subroutine were used to simulate different initial attitudes and tumbling at different rates about different axes. In each case a simple tumbling was assumed so that system performance could be evaluated as a function of simple parameters.

I. DIFFERENT SPACECRAFT CAN BE MODELED

The dynamics model used in these programs is readily adapted to different targets and chase vehicles. For example, different masses, thruster orientations, number of thrusters, thruster forces, docking fixture locations, camera locations, fuel loads, and engine types can be modeled without changing the program logic. Different control laws can be used by changing the logic of subroutine CNTLAW, and different methods for deriving rates can be investigated by replacing the Kalman filter subroutines. These investigations were beyond the scope of the study reported here.

IX. Computer Models of Measurements and Noise

IX. Computer Models of Measurements and Noise

IX. COMPUTER MODELS OF MEASUREMENTS AND NOISE

A. LIGHT POSITIONS ARE MODELED WITH PERSPECTIVE PROJECTION

In each program, lights are assumed to be point light sources, and the images from the cameras are simulated by perspective projection. The calculations are identical for each light in every program, except that the vectors defining camera and lamp positions vary in value. The basic strategy is to compute the position of the lamp in the chase vehicle's camera coordinate system. This system is parallel to the chase vehicle's body coordinate system but offset by $\underline{h}_c$, which is the separation between the camera and the chase vehicle's center of mass. The formula for determining a lamp's position in the camera coordinate system is

$$\underline{r} = A_c (A_t^T \underline{h}_t - \underline{x}) - \underline{h}_c \qquad \text{(IX-1)}$$

The physical interpretation of this formula is shown in Figure IX-1.

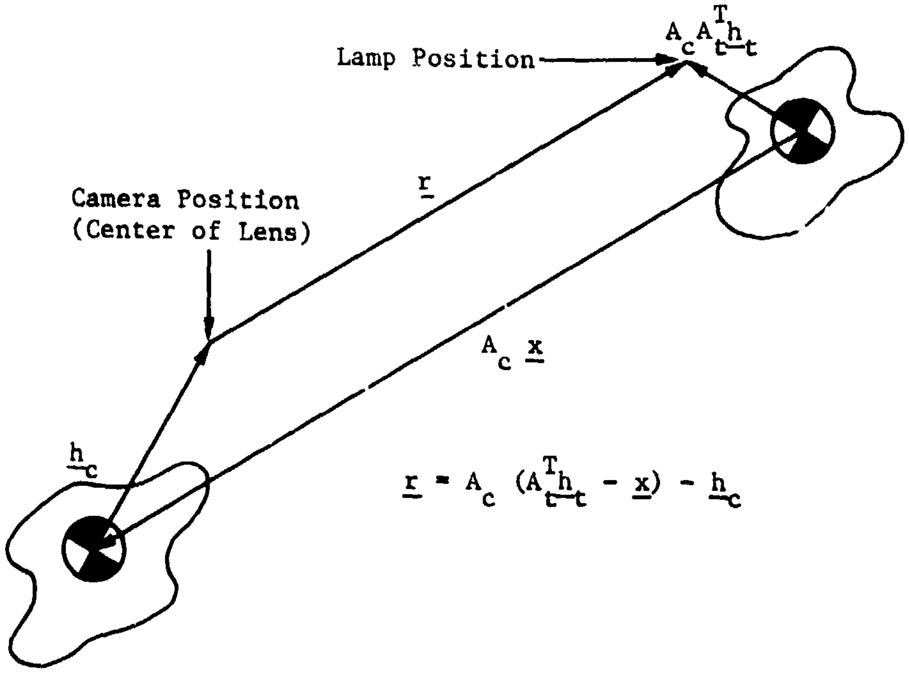


Figure IX-1 Physical Interpretation of Equation (IX-1)

The position of the lamp's image on the television screen can be determined from

$$u = -r_2 f / r_1 \qquad \text{(IX-2)}$$

for the horizontal component and from

$$v = r_3 f/r_1 \qquad \text{(IX-3)}$$

for the vertical component, where r_1, r_2, and r_3 are the x, y, and z components of $\underline{r}$. The direction cosine matrix A_c is derived from the true state, which is represented in the programs as the array STATE. Elements 10 through 13 of this array are the attitude quaternion. It is converted to a direction cosine matrix with equation (D-5) from Appendix D. The vector $\underline{x}$ is the current position of the chase vehicle's center of mass with respect to the "truth" coordinate system defined in Chapter VIII. This vector is the first three elements of the array STATE. The vectors $\underline{h}_c$ and $\underline{h}_t$, which vary from one simulation to the next, are the positions of the camera and lamp, respectively: $\underline{h}_c$ is the camera position with respect to the chase vehicle's center of mass, and $\underline{h}_t$ is the docking aid's position with respect to the target spacecraft's center of mass. The direction cosine matrix A_t is the target's attitude, which is computed as a function of time in subroutine TRGATT, as described in Chapter VIII.

This measurement model's accuracy depends on the validity of several assumptions:

1) The focal plane of the camera is flat and parallel to the y-z plane of the chase vehicle's coordinate system;

2) The lens and the scanning of the camera do not significantly distort the image;

3) The lamp's image is small enough to be considered a point;

4) The lamp is bright enough to be considered the only significant object in the image.

Equations (IX-1) through (IX-3) are used several times in each simulation. In the program that simulates the system with three flashing lights, for example, the equations are used three times per observation with different values of $\underline{h}_t$. Similarly, in the program that simulates the system with stereo ranging, the equations are used three times with different values of $\underline{h}_c$.

B. RANDOM NOISE CORRUPTS IMAGE COORDINATES

To simulate the imperfections of a practical guidance system, the simulation programs add random numbers to the coordinates computed by equations (IX-2) and (IX-3). The random numbers have a Gaussian distribution with a mean of zero and a standard deviation selected to represent the root-mean-square uncertainty in the ideal locations of the lamp images.

This uncertainty arises from several factors:

1) Imperfect electronics are used to compute the center of brightness of the image. The operational amplifiers, analog multipliers, and other components corrupt the video and deflection signals with biases and random errors;

2) Scanning is not perfect. In some cameras a sawtooth analog deflection voltage or current is used for scanning. Misalignment of deflection coils, imperfections in deflection drive amplifiers and similar factors may cause errors in the center-of-brightness calculations. Some cameras use discrete picture elements, which may cause the center of brightness to shift due to quantization errors. Finally, solid-state cameras use no analog deflection voltage, and the use of a synthetic deflection voltage in the center of brightness computation may introduce errors caused by errors in deflection amplitude and synchronization with the actual scanning of the camera;

3) Lenses introduce errors. The distortions, aberrations, and focusing limitations of practical lenses can add additional corruptions to the center of brightness calculations;

4) Camera positioning is not perfect. The mathematical model of equation (IX-1) does not allow for misalignment of the camera axes from the spacecraft axes, errors in the camera's mounting location, or the transient changes in position and alignment caused by thermal expansion and vibration;

5) Camera sensitivity varies from picture element to picture element. In some solid-state cameras the variation is as much as 15 percent. Further, some solid-state cameras have blemishes--picture elements that always give full output or no output no matter how much light shines on them. Charge-coupled device (CCD) cameras tend to "bleed" from one scanning line to the next when a bright light is in the field of view, while other cameras are troubled with blooming and after-images. Many cameras have significant "dark" output that varies from picture element to picture element and may vary greatly with temperature.

Meaningful numbers cannot be assigned to all these error sources, especially because there are no detailed designs for the spacecraft, guidance system, and mission. The noise model should therefore be considered a specification of the maximum allowable signal corruption rather than a prediction of system performance with some specified set of hardware.

Appendix A
Computer Program to Simulate Three-Light System

APPENDIX A--COMPUTER PROGRAM TO SIMULATE THREE-LIGHT SYSTEM

The program listing in this appendix is provided to document the simulation methods used in analyzing the three-light videc guidance system. It was written to run on a Prime 550 computer under the PRIMOS operating system, but it has few hardware-dependent subroutines. If it is to be run on another computer, the following information will prove useful:

1) Several library routines are used, and these are not shown in the listing. The routines include ACOS and ASIN, which compute the inverse trigonmetric functions $\cos^{-1}x$ and $\sin^{-1}x$, and a random number generator, RANFN. The latter function computes normally distributed random values with a specified mean and standard deviation. In addition, the matrix arithmetic routines MADD (addition), MSUB (subtraction), MMLT (multiplication), MINV (inversion), MSCL (multiplication by a scalar), MIDN (setting an array to the identity matrix), and MTRN (forming the transpose of a matrix) are used from the Prime library MATHLB;

2) File handling may present conversion problems even if the program is to be run on another Prime 550 computer because logical unit numbers, file names, and amount of disk storage vary from installation to installation. Standard Prime subroutines are used to open and close files. These subroutines (TSRC$$, EXST$A, CLOS$A, DELE$A) are from the Prime library APPLIB;

3) Run time is approximately real time if the computer is dedicated to a single user. That is, a simulation of three minutes of flight will take approximately three minutes;

4) The perspective drawings shown in this report are not created directly by this program. They are drawn by a second program that uses the data file created by this program. This allows the creation of stereo plots and views from different perspectives;

5) Several WRITE statements in subroutine DOCK are rendered inactive by a character C in the first column of the text. Removing this character will provide a printout at the user's terminal for monitoring the progress of a simulation, but this will make the simulation run more slowly;

6) The goal in writing the program was to make the program easy to modify. As a result, many of the operations are done inefficiently, and a speed improvement by a factor of two or three might be achieved.

The first part of the listing is the text of a terminal session, which includes compilation, loading, and execution of the program.

APPENDIX A -- THREE-LIGHT VERSION OF SIMULATION PROGRAM

SAMPLE RUN

```
OK, FIN MSEC DSIM1
0000 ERRORS [<MAIN >FTN-REV17 4]
0000 ERRORS [<OPEN  >FTN-REV17 4]
0000 ERRORS [<INIPAR>FTN-REV17 4]
0000 ERRORS [<INISIM>FTN-REV17 4]
0000 ERRORS [<DOCK  >FTN-REV17 4]
0000 ERRORS [<CENTRD>FTN-REV17 4]
0000 ERRORS [<PROPTR>FTN-REV17 4]
0000 ERRORS [<COMPK1>FTN-REV17 4]
0000 ERRORS [<COMPK2>FTN-REV17 4]
0000 ERRORS [<COMPK3>FTN-REV17 4]
0000 ERRORS [<COMPK4>FTN-REV17 4]
0000 ERRORS [<POINT >FTN-REV17 4]
0000 ERRORS [<POSIT >FTN-REV17 4]
0000 ERRORS [<ATTITUD>FTN-REV17 4]
0000 ERRORS [<FINDCV>FTN-REV17 4]
0000 ERRORS [<QUATRN>FTN-REV17 4]
0000 ERRORS [<TDQUAT>FTN-REV17 4]
0000 ERRORS [<INCORP>FTN-REV17 4]
0000 ERRORS [<COMPG >FTN-REV17 4]
0000 ERRORS [<ESTCOV>FTN-REV17 4]
0000 ERRORS [<KALGAN>FTN-REV17 4]
0000 ERRORS [<UPDSTA>FTN-REV17 4]
0000 ERRORS [<UPDCOV>FTN-REV17 4]
0000 ERRORS [<PROPES>FTN-REV17 4]
0000 ERRORS [<SETGOL>FTN-REV17 4]
0000 ERRORS [<THRUST>FTN-REV17 4]
0000 ERRORS [<CNTLAW>FTN-REV17 4]
0000 ERRORS [<ACCEL >FTN-REV17 4]
0000 ERRORS [<SELECT>FTN-REV17 4]
0000 ERRORS [<TABLE1>FTN-REV17 4]
0000 ERRORS [<TABLE2>FTN-REV17 4]
0000 ERRORS [<TABLE3>FTN-REV17 4]
0000 ERRORS [<FIRTHR>FTN-REV17 4]
0000 ERRORS [<IMU   >FTN-REV17 4]
0000 ERRORS [<RPY   >FTN-REV17 4]
0000 ERRORS [<ESTRPY>FTN-REV17 4]
0000 ERRORS [<GETPCH>FTN-REV17 4]
0000 ERRORS [<GETYAW>FTN-REV17 4]
0000 ERRORS [<DIRMAT>FTN-REV17 4]
0000 ERRORS [<BTPRIM>FTN-REV17 4]
0000 ERRORS [<FORCE >FTN-REV17 4]
0000 ERRORS [<MPRIME>FTN-REV17 4]
0000 ERRORS [<LINACL>FTN-REV17 4]
0000 ERRORS [<TORQUE>FTN-REV17 4]
0000 ERRORS [<LPRIME>FTN-REV17 4]
0000 ERRORS [<MAKROT>FTN-REV17 4]
0000 ERRORS [<QPRIME>FTN-REV17 4]
0000 ERRORS [<SQR1  >FTN-REV17 4]
0000 ERRORS [<ANGVEC>FTN-REV17 4]
0000 ERRORS [<TRGATT>FTN-REV17 4]
0000 ERRORS [<XPRD  >FTN-REV17 4]
0000 ERRORS [<DPRD  >FTN-REV17 4]

OK, LOAD
$ LO B_MSEC_DSIM2
$ LO 'TIETZ>WORKING NERD>B_RANF
$ LO 'TIETZ>WORKING NERD>B_ASIN
$ LI APPLIB
$ LI MATHLB
$ LI
LOAD COMPLETE
$ SA RUNSIM
$ Q
OK, R RUNSIM
ENTER PATHNAME FOR OUTPUT
DOCFILE
 ENTER THREE ROTATION ANGLES TO SPECIFY INITIAL TARGET
 ATTITUDE FOR A YAW-PITCH-ROLL SEQUENCE---ANGLES MUST BE
 IN DEGREES
10 20 -15
 ENTER TUMBLE AXIS-- ENTER 1, 2, OR 3 FOR YAW, PITCH,
 OR ROLL, RESPECTIVELY
1
 ENTER TUMBLE RATE IN DEGREES PER HOUR
240
 INPUT 16-DIGIT RANDOM INTEGER
2097614296354618
DO YOU WANT ANOTHER SIMULATION? NO
OK,
```



```
C
C            TRUE STATE
C
C               STATE( 1)   X POSITION
C               STATE( 2)   Y POSITION
C               STATE( 3)   Z POSITION
C               STATE( 4)   X VELOCITY
C               STATE( 5)   Y VELOCITY
C               STATE( 6)   Z VELOCITY
C               STATE( 7)   ANGULAR MOMENTUM ABOUT X
C               STATE( 8)   ANGULAR MOMENTUM ABOUT Y
C               STATE( 9)   ANGULAR MOMENTUM ABOUT Z
C               STATE(10)  Q1  \
C               STATE(11)  Q2   \
C               STATE(12)  Q3    > ATTITUDE  .  'TRUTH' AXES
C               STATE(13)  Q4   /
C               STATE(14)   MASS, INCL FUEL
C
C                  (X,Y,Z) = 'TRUTH' AXES
C
C             STATE ESTIMATE
C
C               ESTATE(1)   X' POSITION
C               ESTATE(2)   Y' POSITION
C               ESTATE(3)   Z' POSITION
C               ESTATE(4)   X' VELOCITY
C               ESTATE(5)   Y' VELOCITY
C               ESTATE(6)   Z' VELOCITY
C
C                  (X',Y',Z') = 'PRIMARY REF  FRAME' AXES
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
C        ROUTINE  PAGE ROUTINE-NBR
C
C        <MAIN>     1    000
C        ACCEL     39    481 1
C        ANGVEC    62    904
C        ATITUD    23    440
C        CNTLAW    38    481
C        COMPG     28    431
C        COMPK1    17    421
C        COMPK2    18    422
C        COMPK3    19    423
C        COMPK4    20    424
C        DIRMAT    52    901
C        DOCK      11    400
C        ESTCOV    29    432
C        ESTRPY    51    480
C        FINDCV    24    441
C        FIRTHR    48    483
C        FLASH     14    410
C        FORCE     54    902 1
C        IMU       49    490
C        INCORP    27    430
C        INIPAR     5    200
C        INISIM    10    300
C        KALGAN    30    453
C        LINACL    56    902 3
C        LPRIME    55    902 5
C        MAKROT    59    902 6
C        MPRIME    53    902 2
C        OPEN       4    100
C        POINT     21    425
C        POSIT     22    430
C        PROPES    33    460
C        PRCPTR    16    420
C        QPRIME    60    902 7
C        QUATRN    25    442
C        RPY       50    440
C        SELECT    41    482
C        SETGOL    35    470
C        SGAI      61    903
C        STPRIM    53    902
C        TABLE1    42    482 1
C        TABLE2    44    482 2
C        TABLE3    46    482 3
C        THRUST    37    480
C        TORQUE    57    902 4
C        TRGATT    63    905
C        UPDCOV    32    455
C        UPDSTA    31    454
```



```
C
C        PROGRAM MSFC DSIM1 - DOCKING SIMULATION PROGRAM, THREE-LIGHT
C
C        CALLS
C           CLOS$A, DOCK, EXIT, INIPAR, INISIM, OPEN, YSNO$A
C
C
         LOGICAL CLOS$A
C           LIBRARY FUNCTION TO CLOSE FILES (FUNCTION VALUE INDICATES SUCCESS)
         REAL ESTATE(6)
C           ESTIMATE OF STATE VECTOR
         REAL F(14)
C           FORCES FROM THRUSTERS AFTER SELECTION
         INTEGER OUT
C           FORTRAN UNIT NUMBER OF OUTPUT FILE
         REAL P(6,6)
C           COVARIANCE OF STATE ESTIMATE
         REAL STATE(14)
C           CHASE VEHICLE STATE VECTOR
         REAL T
C           ELAPSED TIME
         INTEGER TERMNL
C           FORTRAN UNIT NUMBER FOR TERMINAL
         LOGICAL DOCKED
C           CODE SET TO ' TRUE ' IF THE CHASE VEHICLE IS WITHIN DOCKING RADIUS
         LOGICAL FILERR
C           ERROR CODE RETURNED FROM 'OPEN' SUBROUTINE
         LOGICAL YSNO$A
C           FUNCTION TO GET YES/NO ANSWER FROM USER AT TERMINAL
C
         COMMON/SIMUL/TERMNL,OUT
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
         OUT=6
         TERMNL=1
C        (* LOOP UNTIL USER WANTS NO MORE SIMULATIONS *)
   10    CONTINUE
C           (* 100 - OPEN DATA FILE *)
               CALL OPEN(FILERR,TERMNL,OUT)
C           (* 200 - INITIALIZE PROBLEM PARAMETERS *)
               CALL INIPAR(T,STATE,P,ESTATE,F)
C           (* 300 - INITIALIZE SIMULATOR *)
               CALL INISIM(STATE)
            DOCKED=.FALSE.
C           (* LOOP WHILE NOT DOCKED *)
   20       IF (DOCKED) GO TO 30
C              (* 400 - SIMULATE OBSERVATION & INTERVAL BETWEEN OBSERVATIONS *)
                  CALL DOCK(P,ESTATE,STATE,T,DOCKED,F)
               GO TO 20
   30       CONTINUE
C           (* CLOSE DATA FILE *)
               IF(.NOT.CLOS$A(OUT-4)) GO TO 40
   40       CONTINUE
C           (* ASK OPERATOR WHETHER HE WANTS ANOTHER SIMULATION *)
               IF(YSNO$A('DO YOU WANT ANOTHER SIMULATION',30,-1)) GO TO 10
         CALL EXIT
C
         END
```

```
C
      SUBROUTINE OPEN(FILERR, TERMNL, OUT)
C     (* 100 - OPEN FILES FOR OUTPUT  WARNING  HIGHLY INSTALLATI(N DEPENDENT *)
C
C     CALLS
C        DELE$A, EXST$A, TSRC$$
C     CALLED BY
C        <MAIN>
C
C     INPUT
C        IN, OUT
C     OUTPUT
C        FILERR
C
      INTEGER CHRPOS(2)
C        CHARACTER POSITION/COUNT CODES FOR SYSTEM ROUTINE TSRC$$
      INTEGER CODE
C        ERROR CODE RETURNED BY TSRC$$ (=0 IF NO ERROR)
      INTEGER I
C        DO LOOP INDEX
      INTEGER OUT, TERMNL
C        FORTRAN LOGICAL UNIT NUMBER FOR OUTPUT
      INTEGER PATH(16)
C        ARRAY OF ASCII DEFINING PATH NAME
      INTEGER TERMNL
C        FORTRAN LOGICAL U''T NUMBER OF USER TE''MINAL
      INTEGER TYPE
C        TYPE CODE RETURNED BY TSRC$$ (UNUSED HERE)
      LOGICAL DELE$A
C        SYSTEM ROUTINE TO DELETE A FILE (RETURNS TRUE IF SUCCESSFUL)
      LOGICAL EXST$A
C        SYSTEM ROUTINE TO CHECK EXISTENCE OF FILE (.'UE .. IT EXISTS)
      LOGICAL FILERR
C        ERROR CODE RETURNED TO CALLING ROUTINE
      LOGICAL YSNO$A
C        FUNCTION TO GET YES/NO ANSWER FROM USER AT TERMINAL
C
$INSERT SYSCOM>?YS F
C - - - - - - - - - - - - - - - - - - - - - - - -   - - - - - - - - - - -
      CHRPOS(2)=32
      TYPE=0
   10 CONTINUE
            WRITE(TERMNL, 902)
            READ(TERMNL, 901)(PATH(I), I=1, 16)
         IF( NOT EXST$A(PATH, 32)) GO TO 20
            IF( NOT YSNO$A('***FILE ALREADY EXISTS   OK TO OVERWRITE',
     A        40, -1)) GO TO 10
      IF( NOT DELE$A(PATH, 32)) GO TO 20
   20 CONTINUE
      CHRPOS(1)=0
      CALL TSRC$$(K$WRIT+K$NSAM, PATH, OUT-4, CHRPOS, TYPE, CODE)
      IF(CODE NE 0) GO TO 30
      FILERR= FALSE
      RETURN
C
   30 FILERR= TRUE
      RETURN
C
  901 FORMAT(16A2)
  902 FORMAT('ENTER PATHNAME FOR OUTPUT     ')
C
      END
```



```
C
      SUBROUTINE INIPAR(T, STATE, P, ESTATE, F)
C     (* 200 - INITIALIZE PROBLEM PARAMETERS *)
C
C     CALLS
C        SQRT, RANFN
C     CALLED BY
C        <MAIN>
C
C     INPUT
C        <NONE>
C     OUTPUT
C        AK1, AK2, AK3, DOKRAD, ESTATE, FOCLEN, FULDIS, F1, INERCV, MEMPTY, P, TPHIH,
C        TPHIV, STATE, TERMNL, T, F, HC, HT, A, B
C
      REAL A, B
C        LAMP SEPARATION ON DOCKING AID (A=BASE TO EITHER SIDE LAMP, B=BASE
C        TO LAMP ON DOCKING AXIS) - VALUES IN METERS
      REAL AK1(14), AK2(3, 14), AK3(3, 14)
C        MATRICES SPECIFYING CHASE VEHICLE
C           AK1 GIVES FUEL USE PER NEWTON-SECOND OF THRUST FOR EACH THRUSTER
C           AK2 GIVES FORCE DIRECTION FOR EACH THRUSTER
C           AK3 GIVES TORQUE/NEWTON & TORQUE DIRECTION FOR EACH THRUSTER
      REAL DOKRAD
C        RADIUS FROM TARGET SPACECRAFT WITHIN WHICH THE SPACECRAFT ARE
C        CONSIDERED DOCKED
      REAL ESTATE(6)
C        ESTIMATE OF STATE VECTOR
      REAL FOCLEN
C        LENS FOCAL LENGTH IN METERS
      REAL FULDIS(3, 3)
C        CHASE VEHICLE FUEL-DISTRIBUTION INERTIA TENSOR
      REAL F(14)
C        FORCES FROM THRUSTERS AFTER SELECTION
      REAL F1(14)
C        LIST OF MAXIMUM THRUST MAGNITUDES FROM EACH THRUSTER OF CHASE VEHICLE
      REAL HC(3) HT(3)
C        CAMERA POSITION IN CHASE VEHICLE BODY FRAME AND DOCKING AID POSITION
C        IN TARGET SPACECRAFT BODY FRAME, RESPECTIVELY
      REAL HDC(3), HDT(3)
C        DOCKING FIXTURE' POSITIONS IN THEIR RESPECTIVE SPACECRAFT COORDI-
C        NATE SYSTEMS
      INTEGER I, J
C        DO LOOP INDICES
      REAL INERCV(3, 3)
C        INERTIA OF EMPTY CHASE VEHICLE
      INTEGER ITUMBL
C        TUMBLE AXIS, VALUE OF 1, 2, OR 3 INDICATES YAW, PITCH, OR R( .
C        TUMBLE, RESPECTIVELY
      REAL MEMPTY
C        MASS OF CHASE VEHICLE WITHOUT FUEL
      INTEGER OUT
C        FORTRAN UNIT NUMBER FOR OUTPUT FILE
      REAL P(6, 6)
C        COVARIANCE OF STATE ESTIMATE
      REAL PHI, PSI, THETA
C        ANGLES SPECIFYING INITIAL TARGET ATTITUDE FOR A YAW-PITCH-ROLL
C        SEQUENCE
      REAL QO(4)
C        INITIAL ATTITUDE QUATERNION OF CHASE VEHICLE
      REAL TPHIH, TPHIV
C        TANGENTS OF HALF-FIELD-OF-VIEW ANGLES IN HORIZONTAL AND VERTICAL
C        DIRECTIONS FOR CAMERA SCANNING, RESPECTIVELY
      REAL STATE(14)
C        CHASE VEHICLE STATE VECTOR
      INTEGER TERMNL
C        FORTRAN LOGICAL UNIT NUMBER OF USER TERMINAL
      REAL T
C        ELAPSED TIME
      REAL TRNATO(3, 3)
C        TRUE INITIAL TARGET DIRECTION COSINE MATRIX
C
      COMMON/MASPRP/FULDIS, INERCV, MEMPTY
      COMMON/OPTSYS/TPHIH, TPHIV, FOCLEN, A, B
      COMMON/VEHIC/AK1, AK2, AK3, DOKRAD, F1
```

```
      COMMON/SIMUL/TERMNL,OUT
      COMMON/HNGOFF/HC,HT,HDC,HDT
      COMMON/REF/GO
      COMMON/ATINFO/TRNATO,ITUMBL,TUMRAT
```



```
C
C     (* INPUT INITIAL ATTITUDE ANGLES *)
      WRITE(TERMNL,901)
      READ(TERMNL,*) PHI,PSI,THETA
C     (* CONVERT ANGLES TO RADIANS *)
         PHI=PHI* 01743329
         PSI=PSI* 01743329
         THETA=THETA* 01745329
C     (* COMPUTE TRANSPOSE OF TRUE INITIAL TARGET ATTITUDE *)
         TRNATO(1,1)=COS(PSI)*COS(PHI)
         TRNATO(2,1)=COS(PSI)*SIN(PHI)
         TRNATO(3,1)=-SIN(PSI)
         TRNATO(1,2)=SIN(THETA)*SIN(PSI)*COS(PHI)-COS(THETA)*SIN(PHI)
         TRNATO(2,2)=SIN(PHI)*SIN(THETA)*SIN(PSI)+COS(THETA)*COS(PHI)
         TRNATO(3,2)=SIN(THETA)*COS(PSI)
         TRNATO(1,3)=COS(THETA)*SIN(PSI)*COS(PHI)+SIN(THETA)*SIN(PHI)
         TRNATO(2,3)=SIN(PHI)*COS(THETA)*SIN(PSI)-SIN(THETA)*COS(PHI)
         TRNATO(3,3)=COS(THETA)*COS(PSI)
C     (* INPUT TUMBLE AXIS *)
        WRITE(TERMNL,902)
    3   READ(TERMNL,*) ITUMBL
          IF(ITUMBL EQ 1 OR ITUMBL EQ 2 OR ITUMBL EQ 3) GO TO 5
          WRITE(TERMNL,904)
          GO TO 3
    5 CONTINUE
C     (* INPUT TUMBLE RATE *)
        WRITE(TERMNL,903)
        READ(TERMNL,*) TUMRAT
C     (* CONVERT TUMBLE RATE FROM DEGREES/HOUR TO RADIANS/SECOND
        TUMRAT=TUMRAT*4 84813E-6
      A=1
      B=1
      DOKRAD= 4
      FOCLEN= 05
      F1(1)=360
      F1(2)=80
      F1(3)=80
      F1(4)=80
      F1(5)=80
      F1(6)=20
      F1(7)=20
      F1(8)=20
      F1(9)=20
      F1(10)=80
      F1(11)=60
      F1(12)=80
      F1(13)=80
      F1(14)=360
      DO 10 I=1,3
         DO 10 J=1,3
            INERCV(I,J)=0
   10 CONTINUE
      INERCV(1,1)=1055
      INERCV(2,2)=1905
      INERCV(3,3)=1905
      TPHIH= 5
      TPHIV= 5
      T=0 0
      DO 20 I=1,14
         F(I)=0
   20 CONTINUE
      DO 30 I=1,3
         DO 30 J=1,3
            FULDIS(I,J)=0.
   30 CONTINUE
      FULDIS(1,1)= 781
      FULDIS(2,2)= 706
      FULDIS(3,3)= 706
      DO 40 I=1,14
         AK1(I)=4 636E-4
   40 CONTINUE
      DO 50 I=1,3
         DO 50 J=1,14
            AK2(I,J)=0
            AK3(I,J)=0
```

```
50 CONTINUE
   AK2(1,1)=-1
   AK2(2,2)=1
   AK2(2,3)=1
   AK2(2,4)=-1
   AK2(2,5)=-1
   AK2(2,6)=-1
   AK2(2,7)=1
   AK2(2,8)=-1
   AK2(2,9)=1
   AK2(3,10)=-1
   AK2(3,11)=-1
   AK2(3,12)=1
   AK2(3,13)=1
   AK2(1,14)=1
   AK3(3,2)=- 375
   AK3(3,3)= 375
   AK3(3,4)=- 375
   AK3(3,5)= 375
   AK3(1,6)=1 25
   AK3(1,7)=-1 25
   AK3(1,8)=-1 25
   AK3(1,9)=1 25
   AK3(2,10)=- 375
   AK3(2,11)= 375
   AK3(2,12)= 375
   AK3(2,13)=- 375
   MEMPTY=.800
   DO 60 I=1,6
      DO 60 J=1,6
         P(I,J)=0
60 CONTINUE
   P(1,1)=400
   P(2,2)=563
   P(3,3)=563
   P(4,4)= 03
   P(5,5)= 03
   P(6,6)= 03
   ESTATE(1)=-300
   ESTATE(2)=0
   ESTATE(3)=0
   ESTATE(4)= 1
   ESTATE(5)= 1
   ESTATE(6)= 1
   DO 70 I=1,6
      STATE(I)=RANFN(2,ESTATE(I),SQRT(P(I,I)))
70 CONTINUE
   STATE(7)=0 0
   STATE(8)=0 0
   STATE(9)=0 0
   STATE(10)=0
   STATE(11)=0
   STATE(12)=0
   STATE(13)=1
   STATE(14)=3700
   QQ(1)=STATE(10)
   QQ(2)=STATE(11)
   QQ(3)=STATE(12)
   QQ(4)=STATE(13)
   HC(1)= 58
   HC(2)=2 0
   HC(3)=0 0
   HT(1)=-1 25
   HT(2)=2 0
   HT(3)=0 0
   HDC(1)=3 49
   HDC(2)=0
   HDC(3)=0
   HDT(1)=-2
   HDT(2)=0
   HDT(3)=0
   RETURN
901 FORMAT(/' ENTER THREE ROTATION ANGLES TO SPECIFY INITIAL TARGET'/
  A     ' ATTITUDE FOR A YAW-PITCH-ROLL SEQUENCE---ANGLES MUST BE'/
  B     ' IN DEGREES')
```



```
902 FORMAT(/' ENTER TUMBLE AXIS-- ENTER 1, 2, OR 3 FOR YAW, PITCH, '/
  A     ' OR ROLL, RESPECTIVELY')
903 FORMAT(/' ENTER TUMBLE RATE IN DEGREES PER HOUR ')
904 FORMAT(/' A VALUE OF 1, 2 OR 3 MUST BE ENTERED FOR TUMBLE AXIS')
C
    END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE INISIM(STATE)
C     (* 300 - INITIALIZE SIMULATOR *)
C
C     CALLS
C        <NONE>
C     CALLED BY
C        <MAIN>
C
C     INPUT
C        <NONE>
C     OUTPUT
C        <NONE>
C
C     (* DUMMY ROUTINE - ACCOMODATES PHYSICAL SIMULATION *)
      RETURN
C
      END
```

C-2


```
C
      SUBROUTINE DOCK(P,ESTATE,STATE,T,DOCKED,F)
C     (* 400 - RUN SIMULATION FOR ONE MEASUREMENT INTERVAL *)
C
C     CALLS
C        SETGOL,INCORP,PROPST,SQRT,FLASH,PROPIN,POSIT,ATITUD,PROPES,THRUST,
C        IMU,RPY,MMLT,MSUB,MADD
C     CALLED BY
C        <MAIN>
C
C     INPUT
C        P,ESTATE,DOKRAD,STATE,T,F,INERCV
C     OUTPUT
C        STATE,T,F,DOCKED,ESTATE,P
C
      REAL ACV(3,3),ACVT(3,3)
C        DIRECTION COSINE MATRIX DESCRIBING CURRENT CHASE VEHICLE ATTITUDE
C        (RELATIVE TO PRIMARY REFERENCE FRAME) AND TRANSPOSE OF ACV
      REAL AT(3,3)
C        MEASURED DIRECTION COSINE MATRIX OF TARGET SPACECRAFT
      REAL ATRATE(3)
C        MEASRED ATTITUDE RATES ABOUT CHASE VEHICLE AXES
      REAL CVPOS(3)
C        MEASURED CHASE VEHICLE POSITION IN PRIMARY REFERENCE FRAME
      REAL DEDLIN
C        TIME BY WHICH STRATEGY LOGIC WANTS CHASE VEHICLE IN GOAL 'BOX'
      REAL DOKRAD
C        RADIUS FROM TARGET AT WHICH CHASE VEHICLE IS CONSIDERED DOCKED
      REAL DU,DV
C        CHANGE IN CENTER-LAMP IMAGE COORDINATES BETWEEN FLASHES
      REAL ESTATE(6)
C        ESTIMATE OF STATE VECTOR
      REAL F(14)
C        FORCES FROM CHASE VEHICLE THRUSTERS AFTER SELECTION
      REAL GXL,GXH,GYL,GYH,GZL,GZH
C        X,Y,Z UPPER,LOWER LIMITS OF GOAL 'BOX'
      REAL HDC(3),HDT(3)
C        DOCKING FIXTURES' POSITIONS IN RESPECTIVE SPACECRAFT COORDINATE
C        SYSTEMS
      INTEGER OUT
C        FORTRAN UNIT NUMBER FOR OUTPUT FILE
      REAL P(6,6)
C        COVARIANCE OF STATE ESTIMATE
      REAL R1(3),R2(3),R3(3),R4(3)
C        INTERMEDIATE RESULTS WITH NO SIMPLE PHYSICAL INTERPRETATION
      REAL RELPOS(3)
C        RELATIVE CAMERA COORDINATES IN CURRENT DOCKING AID FRAME CENTERED
C        AT CENTER LIGHT
      REAL RHO
C        MEASURED RANGE, CAMERA TO TARGET DOCKING AID CENTER LIGHT
      REAL RPYERR(3)
C        MEASURED ROLL, PITCH, AND YAW ERRORS IN RADIANS
      REAL STATE(14)
C        CHASE VEHICLE STATE VECTOR
      REAL T
C        ELAPSED TIME
      REAL TADOUT(3,3)
C        TRANSPOSE OF TARGET SPACECRAFT DIRECTION COSINE MATRIX
C        (WITH RESPECT TO TRUTH AXES) FOR OUTPUT ONLY
      REAL U(4),V(4)
C        COORDINATES OF LAMPS' IMAGES
      LOGICAL DOCKED
C        TRUE IF CHASE VEHICLE IS WITHIN DOCKING RADIUS
      LOGICAL VALID
C        TRUE IF MEASUREMENT VALID (LIGHT IN FIELD OF VIEW)
C
      COMMON/SIMUL/IDUMMY,OUT
      COMMON/HNGOFF/DUMMY3(6),HDC,HDT
      COMMON/VEHIC/DUMMY1(98),DOKRAD,DUMMY2(14)
```

```
C
C     (* TAKE MEASUREMENT *)
        VALID= TRUE
C         (* 410 - FLASH CENTER LIGHT *)
            CALL FLASH(U(4),V(4),2,T,STATE,VALID)
C         (* 420 - PROPAGATE TRUE STATE 1/30 SEC  *)
            CALL PROPTR( 03333333,F,STATE,T)
C         (* 410 - FLASH LEFT LIGHT *)
            CALL FLASH(U(1),V(1),1,T,STATE,VALID)
C         (* 420 - PROPAGATE TRUE STATE 1/30 SEC *)
            CALL PROPTR( 03333333,F,STATE,T)
C         (* 410 - FLASH RIGHT LIGHT *)
            CALL FLASH(U(3),V(3),3,T,STATE,VALID)
C         (* 420 - PROPAGATE TRUE STATE 1/30 SEC *)
            CALL PROPTR( 03333333,F,STATE,T)
C         (* 410 - FLASH CENTER LIGHT SECOND TIME *)
            CALL FLASH(U(2),V(2),2,T,STATE,VALID)
C         (* 490 - TAKE IMU MEASUREMENT *)
            CALL IMU(STATE,ACV,ACVT,ATRATE)
C         (* 420 - PROPAGATE TRUE STATE TO TIME OF THRUSTER COMMAND CHANGE *)
            CALL PROPTR( 53333333,F,STATE,T)
      IF( NOT VALID) GO TO 10
C         (* ADJUST CENTROIDS FOR VEHICLE MOTION *)
            DU=U(2)-U(4)
            DV=V(2)-V(4)
            U(1)=U(1)+ 6666667*DU
            V(1)=V(1)+ 6666667*DV
            U(3)=U(3)+ 3333333*DU
            V(3)=V(3)+ 3333333*DV
C         (* 4A0 - CALCULATE ROLL, PITCH AND YAW ERRORS *)
            CALL RPY(U,V,RPYERR)
C         (* 430 - CALCULATE POSITONS AT TIME OF 4TH FLASH *)
            CALL POSIT(U,V,RELPOS,RHO)
C         (* 440 - CALC  TARGET ATTITUDE AND CHASE VEHICLE POSITION AT 4TH
C           FLASH *)
            CALL ATITUD(ACV,ACVT,RELPOS,RHO,U,V,AT,CVPOS)
C         (* 450 - USE MEASUREMENT TO UPDATE STATE ESTIMATE *)
            CALL INCORP(CVPOS,P,ESTATE)
          GO TO 20
   10 CONTINUE
C         (* 4B0 - ESTIMATE ATTITUDE ERROR FROM STATE *)
            CALL ESTRPY(CSTATE,ACVT,RPYERR)
   20 CONTINUE
C       (* WRITE PARAMETERS TO DISK FILE *)
          CALL TRGATT(TAOUT,T)
          WRITE(OUT) T,STATE,TAOUT
C       (* 460 - PROPAGATE STATE ESTIMATE TO TIME OF THRUSTER COMMAND CHANGE *)
          CALL PROPES(P,ESTATE,F, 5333333,ACVT)
C       (* 470 - SET GOAL FOR NEXT MEASUREMENT INTERVAL *)
          CALL SETGOL(ESTATE,P,T,ACV,AT,GXL,GXH,GYL,GYH,GZL,GZH,DEDLIN,
     A      VALID)
C       (* 480 - CONTROL THRUSTERS *)
          CALL THRUST(F,ESTATE,GXL,GXH,GYL,GYH,GZL,GZH,AMAX1(
     A        DEDLIN-T,0 ),ACV,ATRATE,RPYERR)
C       (* 420 - PROPAGATE TRUE STATE TO TIME OF FIRST FLASH OF NEXT SEQUENCE *)
          CALL PROPTR(0 6,F,STATE,T)
C       (* 460 - PROP  STATE ESTIMATE TO TIME OF 4TH FLASH OF NEXT SEQUENCE *)
          CALL PROPES(P,ESTATE,F,0 7,ACVT)
C       (* SEE IF CHASE VEHICLE HAS DOCKED YET *)
          CALL MMLT(R1,ACVT,HDC,3,3,1)
          CALL MADD(R2,R1,STATE,3,1)
          CALL MMLT(R3,AT,R2,3,3,1)
          CALL MSUB(R4,HDT,R3,3,1)
          DOCKED=(SQRT(R4(1)**2+R4(2)**2+R4(3)**2) LE DOKRAD)  OR
     A       (T GT 280 )  OR
     B       (STATE(1) LT -380 )  OR  (STATE(1) GT 15 )  OR
     C       (AMAX1(ABS(STATE(2)),ABS(STATE(3))) GT 200 )
C       (* WRITE PARAMETERS TO DISK FILE *)
          CALL TRGATT(TAOUT,T)
          WRITE(OUT) T,STATE,TAOUT
C         WRITE(1,901) T,STATE(1),STATE(2),STATE(3)
C         WRITE(1,902) STATE(4),STATE(5),STATE(6)
C         WRITE(1,903) STATE(10),STATE(11),STATE(12),STATE(13)
C         WRITE(1,904) STATE(7),STATE(8),STATE(9)
C         WRITE(1,905) STATE(14)
```



```
        RETURN
C
  901 FORMAT(/' -----TIME=',F10 2/' POSITION IS ',3G14 7)
  902 FORMAT(' VELOCITY IS ',3G14 7)
  903 FORMAT(' QUATERNION IS ',4G14 7)
  904 FORMAT(' ANGULAR MOMENTUM IS ',3G14 7)
  905 FORMAT(' MASS IS ',G14 7)
C
        END
```

```
      SUBROUTINE FLASH(U,V,LAMP,T,STATE,VALID)
C     (* 410 - FLASH A LAMP *)
C
C     CALLS
C        TRGATT,MMLT,DIRMAT,MSUB,RANFN,MTRN
C     CALLED BY
C        DOCK
C
C     INPUT
C        FOCLEN,TPHIH,TPHIV,LAMP,STATE,A,B,VALID
C     OUTPUT
C        U,V,VALID
C
      REAL AC(3,3),TRNAC(3,3)
C        CHASE VEHICLE DIRECTION COSINE MATRIX AND ITS TRANSPOSE
      REAL A,B
C        LAMP SEPARATIONS (IN METERS)
      REAL AT(3,3),TRNAT(3,3)
C        TARGET DIRECTION COSINE MATRIX & ITS TRANSPOSE
      REAL HC(3),HT(3)
C        CAMERA POSITION IN CHASE VEHICLE BODY FRAME AND DOCKING AID POSITION
C        IN TARGET SPACECRAFT BODY FRAME, RESPECTIVELY
      REAL FOCLEN
C        LENS FOCAL LENGTH IN METERS
      INTEGER LAMP
C        LAMP NUMBER (1-3)    1=LEFT,2=CENTER,3=RIGHT
      REAL STATE(14)
C        TRUE CHASE VEHICLE STATE
      REAL T
C        ELAPSED TIME
      REAL TPHIH,TPHIV
C        TANGENTS OF HALF-FIELD-OF-VIEW ANGLES IN HORIZONTAL AND VERTICAL
C        DIRECTIONS FOR CAMERA SCANNING, RESPECTIVELY
      REAL U,V
C        COORDINATES OF CENTROID OF LAMP IMAGE IN METERS
      LOGICAL VALID
C        TRUE IF MEASUREMENT VALID (LIGHTS IN FIELD OF VIEW)
      REAL VLT(3)
C        COORDINATES OF LAMP WITH RESPECT TO CAMERA
      REAL VL(3),V1(3),V2(3),V3(3),V5(3),V6(3)
C        INTERMEDIATE RESULTS WITH NO PROBLEM-RELATED SIGNIFICANCE
C
      COMMON/OPTSYS/TPHIH,TPHIV,FOCLEN,A,B
      COMMON/HNGOFF/HC,HT,DUMMY(6)
```



```
C
C     (* FIND LAMP POSITION (=VLT) WITH RESPECT TO CAMERA *)
C        (* 905 - COMPUTE TARGET ATTITUDE *)
         CALL TRGATT(TRNAT,T)
      VL(3)=HT(3)
      GO TO (10,20,30), LAMP
10       CONTINUE
            VL(1)=HT(1)
            VL(2)=HT(2)+A
            GO TO 40
20       CONTINUE
            VL(1)=HT(1)-B
            VL(2)=HT(2)
            GO TO 40
30       CONTINUE
            VL(1)=HT(1)
            VL(2)=HT(2)-A
40    CONTINUE
      CALL MMLT(V1,TRNAT,VL,3,3,1)
      CALL MSUB(V2,V1,STATF,3,1)
      CALL DIRMAT(STATE(10),AC,TRNAC)
      CALL MMLT(V3,AC,V2,3,3,1)
      CALL MSUB(VLT,V3,HC,3,1)
C     (* FIND DEFLECTION *)
      U=-VLT(2)*FOCLEN/VLT(1)
      V=VLT(3)*FOCLEN/VLT(1)
C     (* BE SURE DOCKING AID IS IN FIELD OF VIEW OF CAMERA *)
      VALID=VALID AND VLT(1) GT 0
      IF( NOT VALID) RETURN
C     (* BE SURE DOCKING AID IS ON VISIBLE SIDE OF TARGET SPACECRAFT *)
      CALL MMLT(V5,TRNAC,VLT,3,3,1)
      CALL MTRN(AT,TRNAT,3)
      CALL MMLT(V6,AT,V5,3,3,1)
      VALID=VALID AND V6(1) GT 0
C     (* CORRUPT CENTROID COORDINATES WITH NOISE *)
      U=RANFN(6,U,TPHIH*FOCLEN/200 )
      V=RANFN(6,V,TPHIV*FOCLEN/200 )
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE PROPTR(DT,F,STATE,T)
C     (* 420 - PROPAGATE TRUE STATE BY 4TH-ORDER RUNGE-KUTTA INTEGRATION *)
C
C     CALLS
C        COMPK1,COMPK2,COMPK3,COMPK4,POINT
C     CALLED BY
C        DOCK
C
C     INPUT
C        DT,F,STATE,T
C     OUTPUT
C        STATE,T
C
      REAL DT
C        INTEGRATION INTERVAL SIZE
      REAL F(14)
C        FORCES FROM THRUSTERS AFTER SELECTION
      INTEGER I
C        DO LOOP INDEX
      REAL K1(14),K2(14),K3(14),K4(14)
C        VECTORS K1 THRU K4 USED IN RUNGE-KUTTA INTEGRATION
      REAL QM
C        SQUARE ROOT OF SUM OF SQUARES OF QUATERNION ELEMENTS
      REAL STATE(14)
C        CHASE VEHICLE STATE VECTOR
      REAL STEP
C        STEP SIZE FOR INTEGRATION
      REAL T,TO
C        ELAPSED TIME AND TIME AT START OF INTEGRATION INTERVAL
      REAL TLEFT
C        TIME LEFT TO GO OUT OF DT
      PARAMETER STEPMX=0.2
C        MAXIMUM INTEGRATION STEP SIZE
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      TLEFT=DT
      TO=T
   10 IF(TLEFT.LE.0) GO TO 40
         STEP=AMIN1(STEPMX,TLEFT)
         TLEFT=TLEFT-STEP
C        (* 421 - COMPUTE K1 *)
            CALL COMPK1(F,K1,STATE,STEP,T)
C        (* 422 - COMPUTE K2 *)
            CALL COMPK2(F,K1,STATE,STEP,T,K2)
C        (* 423 - COMPUTE K3 *)
            CALL COMPK3(F,K2,STATE,STEP,T,K3)
C        (* 424 - COMPUTE K4 *)
            CALL COMPK4(F,K3,STATE,STEP,T,K4)
C        (* COMPUTE NEW STATE *)
            DO 20 I=1,14
               STATE(I)=STATE(I)+(K1(I)+2.*K2(I)+2.*K3(I)+K4(I))/6
   20       CONTINUE
C        (* NORMALIZE QUATERNION *)
            QM=SQRT(STATE(10)**2+STATE(11)**2+STATE(12)**2+
     A         STATE(13)**2)
            DO 30 I=10,13
               STATE(I)=STATE(I)/QM
   30       CONTINUE
C        (* 425 - POINT SIMULATION CAMERA *)
            CALL POINT(STATE)
         T=T+STEP
         GO TO 10
   40 CONTINUE
      T=TO+DT
      RETURN
C
      END
```



```
C
      SUBROUTINE COMPK1(F,K1,STATE,STEP,T)
C     (* 421 - COMPUTE VECTOR K1 FOR RUNGE-KUTTA INTEGRATION *)
C
C     CALLS
C        STPRIM,MSCL,MADD
C     CALLED BY
C        PROPTR
C
C     INPUT
C        F,STATE,STEP,T,FULDIS,INERCV,MEMPTY
C     OUTPUT
C        K1
C
      REAL DSTATE(14)
C        TIME DERIVATIVE OF STATE VECTOR
      REAL F(14)
C        FORCES FROM THRUSTERS AFTER SELECTION
      REAL FULDIS(3,3)
C        CHASE VEHICLE FUEL-DISTRIBUTION INERTIA TENSOR
      INTEGER I
C        DO LOOP INDEX
      REAL INERFL(3,3)
C        INERTIA OF FUEL
      REAL INERTA(3,3)
C        INERTIA OF CHASE VEHICLE (LOADED)
      REAL K1(14)
C        VECTOR K1 USED IN RUNGE-KUTTA INTEGRATION
      REAL STATE(14)
C        CHASE VEHICLE STATE VECTOR
      REAL STEP
C        STEP SIZE FOR INTEGRATION
      REAL T
C        ELAPSED TIME
C
      COMMON/MASPRP/FULDIS,INERCV,MEMPTY
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* COMPUTE INERTIA *)
         CALL MSCL(INERFL,FULDIS,3,3,STATE(14)-MEMPTY)
         CALL MADD(INERTA,INERCV,INERFL,3,3)
C     (* 902 - COMPUTE STATE DERIVATIVE *)
         CALL STPRIM(STATE,F,INERTA,T,DSTATE)
C     (* COMPUTE K1=STEP*(STATE DERIVATIVE) *)
         DO 10 I=1,14
            K1(I)=STEP*DSTATE(I)
   10    CONTINUE
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE COMPK2(F,K1,STATE,STEP,T,K2)
C     (* 422 - DETERMINE VECTOR K2, FOR RUNGE-KUTTA INTEGRATION *)
C
C     CALLS
C        MADD,MSCL,STPRIM
C     CALLED BY
C        PRUPTR
C
C     INPUT
C        F,FULDIS,INERCV,K1,MEMPTY,STATE,STEP,T
C     OUTPUT
C        K2
C
      REAL DSTATE(14)
C        TIME DERIVATIVE OF STATE VECTOR
      REAL F(14)
C        FORCES FROM THRUSTERS AFTER SELECTION
      REAL FULDIS(3,3)
C        CHASE VEHICLE FUEL-DISTRIBUTION INERTIA TENSOR
      INTEGER I
C        DO LOOP INDEX
      REAL INERCV(3,3)
C        INERTIA OF EMPTY CHASE VEHICLE
      REAL INERFL(3,3)
C        INERTIA OF FUEL
      REAL INERTA(3,3)
C        INERTIA OF CHASE VEHICLE (LOADED)
      REAL K1(14),K2(14)
C        VECTORS K1 AND K2 USED IN RUNGE-KUTTA INTEGRATION
      REAL MEMPTY
C        MASS OF CHASE VEHICLE WITHOUT FUEL
      REAL STATE(14)
C        CHASE VEHICLE STATE VECTOR
      REAL STEP
C        STEP SIZE FOR INTEGRATION
      REAL T
C        ELAPSED TIME
      REAL TEMPST(14)
C        TEMPORARY STATE VECTOR
C
      COMMON/MASPRP/FULDIS,INERCV,MEMPTY
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* COMPUTE TEMPORARY STATE *)
      DO 10 I=1,14
         TEMPST(I)=STATE(I)+0.5*K1(I)
   10 CONTINUE
C     (* DETERMINE TEMPORARY INERTIA AS FUNCTION OF TEMPORARY
C     STATE MASS *)
      CALL MSCL(INERFL,FULDIS,3,3,TEMPST(14)-MEMPTY)
      CALL MADD(INERTA,INERCV,INERFL,3,3)
C     (* 902 - COMPUTE DERIVATIVE AT TEMPORARY STATE *)
      CALL STPRIM(TEMPST,F,INERTA,T+.5*STEP,DSTATE)
C     (* COMPUTE K2=STEP*(DERIVATIVE AT TEMPORARY STATE *)
      DO 20 I=1,14
         K2(I)=STEP*DSTATE(I)
   20 CONTINUE
      RETURN
C
      END
```



```
C
      SUBROUTINE COMPK3(F,K2,STATE,STEP,T,K3)
C     (* 423 - DETERMINE VECTOR K3, FOR RUNGE-KUTTA INTEGRATION *)
C
C     CALLS
C        MADD,MSCL,STPRIM
C     CALLED BY
C        PROPTR
C
C     INPUT
C        F,FULDIS,INERCV,K2,MEMPTY,STATE,STEP,T
C     OUTPUT
C        K3
C
      REAL DSTATE(14)
C        TIME DERIVATIVE OF STATE VECTOR
      REAL F(14)
C        FORCES FROM THRUSTERS AFTER SELECTION
      REAL FULDIS(3,3)
C        CHASE VEHICLE FUEL-DISTRIBUTION INERTIA TENSOR
      INTEGER I
C        DO LOOP INDEX
      REAL INERCV(3,3)
C        INERTIA OF EMPTY CHASE VEHICLE
      REAL INERFL(3,3)
C        INERTIA OF FUEL
      REAL INERTA(3,3)
C        INERTIA OF CHASE VEHICLE (LOADED)
      REAL K2(14),K3(14)
C        VECTORS K2 AND K3 USED IN RUNGE-KUTTA INTEGRATION
      REAL MEMPTY
C        MASS OF CHASE VEHICLE WITHOUT FUEL
      REAL STATE(14)
C        CHASE VEHICLE STATE VECTOR
      REAL STEP
C        STEP SIZE FOR INTEGRATION
      REAL T
C        ELAPSED TIME
      REAL TEMPST(14)
C        TEMPORARY STATE VECTOR
C
      COMMON/MASPRP/FULDIS,INERCV,MEMPTY
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* COMPUTE TEMPORARY STATE *)
      DO 10 I=1,14
         TEMPST(I)=STATE(I)+0.5*K2(I)
   10 CONTINUE
C     (* DETERMINE TEMPORARY INERTIA AS A FUNCTION OF TEMPORARY
C              STATE MASS *)
      CALL MSCL(INERFL,FULDIS,3,3,TEMPST(14)-MEMPTY)
      CALL MADD(INERTA,INERCV,INERFL,3,3)
C     (* 902 - COMPUTE DERIVATIVE AT TEMPORARY STATE *)
      CALL STPRIM(TEMPST,F,INERTA,T+.5*STEP,DSTATE)
C     (* COMPUTE K3=STEP*(DERIVATIVE AT TEMPORARY STATE *)
      DO 20 I=1,14
         K3(I)=STEP*DSTATE(I)
   20 CONTINUE
      RETURN
C
      END
```

```
C
      SUBROUTINE COMPK4(F,K3,STATE,STEP,T,K4)
C     (* 424 - DETERMINE VECTOR K4, FOR RUNGE-KUTTA INTEGRATION *)
C
C     CALLS
C        MADD,MSCL,STPRIM
C     CALLED BY
C        PROPTR
C
C     INPUT
C        F,FULDIS,INERCV,K3,MEMPTY,STATE,STEP,T
C     OUTPUT
C        K4
C
      REAL DSTATE(14)
C        TIME DERIVATIVE OF STATE VECTOR
      REAL F(14)
C        FORCES FROM THRUSTERS AFTER SELECTION
      REAL FULDIS(3,3)
C        CHASE VEHICLE FUEL-DISTRIBUTION INERTIA TENSOR
      INTEGER I
C        DO LOOP INDEX
      REAL INERCV(3,3)
C        INERTIA OF EMPTY CHASE VEHICLE
      REAL INERFL(3,3)
C        INERTIA OF FUEL
      REAL INERTA(3,3)
C        INERTIA OF CHASE VEHICLE (LOADED)
      REAL K3(14),K4(14)
C        VECTORS K3 AND K4 USED IN RUNGE-KUTTA INTEGRATION
      REAL MEMPTY
C        MASS OF CHASE VEHICLE WITHOUT FUEL
      REAL STATE(14)
C        CHASE VEHICLE STATE VECTOR
      REAL STEP
C        STEP SIZE FOR INTEGRATION
      REAL T
C        ELAPSED TIME
      REAL TEMPST(14)
C        TEMPORARY STATE VECTOR
C
      COMMON/MASPRP/FULDIS,INERCV,MEMPTY
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* COMPUTE TEMPORARY STATE *)
         DO 10 I=1,14
            TEMPST(I)=STATE(I)+K3(I)
   10    CONTINUE
C     (* DETERMINE TEMPORARY INERTIA AS A FUNCTION OF TEMPORARY
C           STATE MASS *)
         CALL MSCL(INERFL,FULDIS,3,3,TEMPST(14)-MEMPTY)
         CALL MADD(INERTA,INERCV,INERFL,3,3)
C     (* 902 - COMPUTE DERIVATIVE AT TEMPORARY STATE *)
         CALL STPRIM(TEMPST,F,INERTA,T+STEP,DSTATE)
C     (* COMPUTE K4=STEP*(DERIVATIVE AT TEMPORARY STATE *)
         DO 20 I=1,14
            K4(I)=STEP*DSTATE(I)
   20    CONTINUE
      RETURN
C
      END
```



```
C
      SUBROUTINE POINT(STATE)
C     (* 425 - POSITION SIMULATION CAMERA *)
C     (* THIS DUMMY SUBROUTINE IS FOR LATER EXPANSION TO PHYSICAL SIMULATION*)
C
C     CALLS
C        <NONE>
C     CALLED BY
C        PROPTR
C
      RETURN
C
      END
```

```
C
      SUBROUTINE POSIT(U,V,RELPOS,RHO)
C     (* 430 -  COMPUTE CAMERA POSITION (IN DOCKING-AID COORD  SYSTEM) FROM
C        LIGHT IMAGE CENTROIDS *)
C
C     CALLS
C        SQR1,ABS
C     CALLED BY
C        DOCK
C
C     INPUT -
C        U,V
C     OUTPUT -
C        RELPOS,RHO
C
      REAL A,B
C        LIGHT SPACING (FIXED FOR ANY ONE TARGET SPACECRAFT)
      REAL FOCLEN
C        LENS FOCAL LENGTH IN METERS
      REAL RELPOS(3)
C        RELATIVE CAMERA COORDINATES IN CURRENT TARGET DOCKING AID REFERENCE
C        FRAME CENTERED AT CENTER LIGHT
      REAL RHO
C        DISTANCE BETWEEN LIGHTS AND CAMERA
      REAL U(4),V(4)
C        HORIZONTAL AND VERTICAL COMPONENTS FOR LEFT, 2ND CENTER, RIGHT,
C        AND FIRST CENTER CENTROID POSITIONS, RESPECTIVELY
      REAL AP,AK,BP,D,H,S,SP,UC,VC,XP,YP,ZP,VM,UM
C        INTERMEDIATE RESULTS. SEE ENGR NOTEBOOK #1035, P  44
C
      COMMON/OPTSYS/DUMMY(2),FOCLEN,A,B
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      VM= 5*(V(1)+V(3))
      UM= 5*(U(1)+U(3))
      IF(V(1) NE V(3)) GO TO 10
         AP= 5*ABS(U(3)-U(1))
         UC=U(2)
         VC=V(1)
         H=ABS(V(1)-V(2))
         GO TO 30
   10 IF(U(1) NE U(3)) GO TO 20
         AP= 5*ABS(V(3)-V(1))
         UC=U(1)
         VC=V(1)
         H=ABS(U(1)-U(2))
         GO TO 30
   20 CONTINUE
         AP= 5*SQR1((U(3)-U(1))**2+(V(3)-V(1))**2)
         S=(V(1)-V(3))/(U(1)-U(3))
         SP=-1 /S
         UC=(V(2)-V(1)+S*U(1)-SP*U(2))/(S-SP)
         VC=SP*(UC-U(2))+V(2)
         H=SQR1((UC-U(2))**2+(VC-V(2))**2)
   30 CONTINUE
      BP=SQR1((VM-V(2))**2+(UM-U(2))**2)
      D=(AP*B/(BP*A))**2
      AK=BP*(1 +D)/(2 *H*SQR1(D))
      ZP=AK-SQR1(AK**2-1 )
      XP=-SQR1((D-ZP**2)/(1 +D))
      YP=SQR1((1 -D*ZP**2)/(1 +D))
      IF(V(2) GT VC) ZP=-ZP
      IF(U(1)+U(3) GT 2 *UC) YP=-YP
      RHO=A*SQR1( (XP**2+ZP**2) * (UM**2+VM**2+FOCLEN**2) )/AP
      RELPOS(1)=XP*RHO+B
      RELPOS(2)=YP*RHO
      RELPOS(3)=ZP*RHO
      RHO=SQR1(RELPOS(1)**2+RELPOS(2)**2+RELPOS(3)**2)
      RETURN
C
      END
```



```
C
      SUBROUTINE ATITUD(ACV,ACVT,RELPOS,RHO,U,V,AT,CVPOS)
C     (* 440 - DETERMINE TARGET ATTITUDE AND CHASE VEHICLE POSITION IN PRIMARY
C        REFERENCE FRAME *)
C
C     CALLS
C        FINDCV,DIRMAT,QUATRN,MADD,MMLT,MSUB
C     CALLED BY
C        DOCK
C
C     INPUT
C        RELPOS,RHO,U,V,ACV,ACVT,HC,HT,B
C     OUTPUT
C        AT,CVPOS
C
      REAL ACV(3,3),ACVT(3,3)
C        DIRECTION COSINE MATRIX DESCRIBING CURRENT CHASE VEHICLE ATTITUDE
C        RELATIVE TO PRIMARY REFERENCE FRAME, AND ITS TRANSPOSE
      REAL AT(3,3),ATT(3,3)
C        DIRECTION COSINE MATRIX OF TARGET SPACECRAFT (IN PRIMARY REF  FRAME),
C        AND TRANSPOSE OF THIS MATRIX
      REAL B
C        DOCKING AID BASE-TO-CENTER-LIGHT DISTANCE
      REAL CAMPOS(3)
C        MEASURED CAMERA POSITION IN COORD  SYS  CENTERED AT CENTER DOCKING
C        AID LIGHT AND PARALLEL TO PRIMARY REFERENCE FRAME
      REAL CVPOS(3)
C        MEASURED CHASE VEHICLE POSITION IN PRIMARY REFERENCE FRAME
      REAL HC(3),HT(3)
C        CAMERA POSITION IN CHASE VEHICLE BODY FRAME AND DOCKING AID POSITION
C        IN TARGET SPACECRAFT BODY FRAME, RESPECTIVELY
      REAL HTA(3)
C        HT PLUS LENGTH OF CENTER LIGHT SUPPORT ROD
      REAL QT(4)
C        QUATERNION CORRESPONDING TO AT
      REAL RELPOS(3)
C        RELATIVE CHASE VEHICLE COORDINATES IN CURRENT TARGET SPACECRAFT
C        REFERENCE FRAME
      REAL RHO
C        DISTANCE BETWEEN LIGHTS AND CAMERA
      REAL U(3),V(3)
C        HORIZONTAL AND VERTICAL COMPONENTS OF CENTER-LIGHT CENTROID
      REAL V1(3),V2(3),V3(3)
C        INTERMEDIATE RESULTS WITH NO PROBLEM-RELATED SIGNIFICANCE
C
      COMMON/HNGOFF/HC,HT,DUMMY(6)
      COMMON/OPTSYS/DUMMY1(4),B
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* 441 - COMPUTE CAMERA POSITION WITH RESPECT TO DOCKING AID *)
         CALL FINDCV(ACVT,RHO,U(2),V(2),CAMPOS)
C     (* 442 - COMPUTE TARGET ATTITUDE IN PRIMARY REFERENCE FRAME *)
         CALL QUATRN(CAMPOS,RELPOS,QT,ACV,U,V)
C     (* 901 - COMPUTE TARGET DIRECTION COSINE MATRIX FROM QUATERNION *)
         CALL DIRMAT(QT,AT,ATT)
C     (* COMPUTE POSITION OF CHASE VEHICLE CENTER OF GRAVITY IN PRIMARY REF
C        FRAME *)
         HTA(1)=HT(1)-B
         HTA(2)=HT(2)
         HTA(3)=HT(3)
         CALL MMLT(V2,ATT,HTA,3,3,1)
         CALL MADD(V1,CAMPOS,V2,3,1)
         CALL MMLT(V3,ACVT,HC,3,3,1)
         CALL MSUB(CVPOS,V1,V3,3,1)
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE FINDCV(ACVT,RHO,UC,VC,CAMPOS)
C     (* 441 - COMPUTE POSITION OF CAMERA FRAME PARALLEL TO '   'ARY REF  FRAME
C        BUT CENTERED AT CENTER TARGET LIGHT *)
C
C     CALLS
C        MMLT SQRT
C     CALLED BY
C        ATITUD
C
C     INPUT
C        ACVT,RHO,UC,VC,FOCLEN
C     OUTPUT
C        CAMPOS
C
      REAL ACVT(3,3)
C        DIRECTION COSINE MARTIX DESCRIBING CURRENT CHASE VEHICLE ATTITUDE
C        RELATIVE TO PRIMARY REFERENCE FRAME
      REAL CAMPOS(3)
C        MEASURED CAMERA POSITION IN FRAME PARALLEL TO PRIMARY REF  FRAME
C        BUT CENTERED AT CENTER TARGET LIGHT
      REAL FOCLEN
C        CAMERA LENS FOCAL LENGTH, IN METERS
      REAL RATIO
C        INTERMEDIATE RESULTS, NO PROBLEM RELATED SIGNIFICANCE
      REAL RHO
C        MEASURED DISTANCE BETWEEN CENTER DOCKIN AID LIGHT AND CAMERA
      REAL UC,VC
C        HORIZONTAL AND VERTICAL COMPONENTS OF THE TARGET CENTER-LIGHT
C        IMAGE CENTROID
      REAL VEC1(3)
C        NEGATIVE OF TARGET POSITION IN CAMERA FRAME
C
      COMMON/OPTSYS/DUMMY1(2),FOCLEN,DUMMY2(2)
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* COMPUTE SCALAR MULTIPLIER *)
      RATIO=RHO/SQRT(FOCLEN**2+UC**2+VC**2)
C     (* FIND NEG  OF TARGET POSITION IN CAMERA FRAME *)
      VEC1(1)=-FOCLEN*RATIO
      VEC1(2)=UC*RATIO
      VEC1(3)=-VC*RATIO
C     (* CONVERT TO CAMERA POSITION IN TARGET CENTER-LIGHT FRAME *)
      CALL MMLT(CAMPOS,ACVT,VEC1,3,3,1)
      RETURN
C
      END
```



```
C
      SUBROUTINE QUATRN(CAMPOS,RELPOS,QT,ACV,U,V)
C     (* 442 - COMPUTE TARGET ATTITUDE QUATERNION IN PRIMARY REFERENCE FRAME
C        FROM MEASUREMENT *)
C
C     CALLS
C        ATAN2,SIN,SQR1,COS,MSCL,DIRMAT,MMLT
C     CALLED BY
C        ATITUD
C
C     INPUT
C        CAMPOS,RELPOS,ACV,U,V
C     OUTPUT
C        QT
C
      REAL ACV(3,3)
C        CHASE VEHICLE'S MEASURED DIRECTION COSINE MATRIX
      REAL AT(3,3),TRNAT(3,3)
C        DIRECTION COSINE MATRIX FOR TARGET (WITH RESPECT .J PRIMARY REF
C        FRAME, BEFORE CORRECTION) AND ITS TRANSPOSE
      REAL CAMPOS(3)
C        MEASURED CAMERA POSITION IN REFERENCE FRAME PARALLEL TO PRIMARY
C        REF  FRAME BUT CENTERED AT CENTER DOCKING AID LIGHT
      REAL DOTPRD
C        DOT PRODUCT OF CAMPOS, RELPOS
      REAL PHI
C        EULER ROTATION ANGLE
      REAL PRDMAG
C        MAGNITUDE OF CROSS PRODUCT
      REAL QC(4)
C        CORRECTION QUATERNION
      REAL QT1(4),QT(4)
C        TARGET ATTITUDE QUATERNION (MAYBE WITH ROTATION ABOUT LINE OF SIGHT)
C        AND CORRECTED VERSION OF SAME
      REAL RELPOS(3)
C        CAMERA COORDINATES IN CURRENT DOCKING AID REFERENCE FRAME
      REAL ROTLOS
C        COMPENSATING ROTATION ANGLE ABOUT LINE OF SIGHT
      REAL U(3),V(3)
C        LAMP IMAGE CENTROIDS
      REAL V1(3)
C        INTERMEDIATE RESULTS   CROSS PRODUCT COMPONENTS
      REAL V2(3),V3(3)
C        INTERMEDIATE RESULTS
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
C     (* COMPUTE COMPONENTS OF CROSS PRODUCT TO DEFINE ROTATION AXIS FROM
C         RELPOS TO CAMPOS *)
          V1(1)=-CAMPOS(2)*RELPOS(3)+CAMPOS(3)*RELPOS(2)
          V1(2)=-CAMPOS(3)*RELPOS(1)+CAMPOS(1)*RELPOS(3
          V1(3)=-CAMPOS(1)*RELPOS(2)+CAMPOS(2)*RELPOS(1)
          PRDMAG=SQR1(V1(1)**2+V1(2)**2+V1(3)**2)
          DOTPRD=CAMPOS(1)*RELPOS(1)+CAMPOS(2)*RELPOS(2)+CAMPOS(3)*
     A        RELPOS(3)
          IF(PRDMAG GT 0 ) GO TO 20
C             (* VECTORS ARE PARALLEL OR ANTIPARALLEL  FIND OUT WHICH *)
                QT1(1)=0
                QT1(2)=0
                IF(DOTPRD GT 0 ) GO TO 10
                    QT1(3)=1 0
                    QT1(4)=0 0
                    GO TO 30
   10           CONTINUE
                    QT1(3)=0
                    QT1(4)=1 0
                    GO TO 30
   20     CONTINUE
C             (* COMPUTE EULER ROTATION ANGLE *)
                PHI=ATAN2(PRDMAG,DOTPRD)
C             (* COMPUTE QUATERNION COMPONENTS *)
                CALL MSCL(QT1,V1,3,1,SIN(PHI* 5)/PRDMAG)
                QT1(4)=COS(PHI* 5)
   30 CONTINUE
      IF(U(3) EQ U(1) AND V(3) EQ V(1)) RETURN
C     (* CORRECT FOR ROTATION ABOUT LINE OF SIGHT *)
C         (* 901 - CONVERT QT1 TO DIRECTION COSINE MATRIX *)
            CALL DIRMAT(QT1,AT,TRNAT)
C         (* FIND ROTATION OF LINF IN -Y DIRECTION ON TARGET, AS PREDICTED
C           BY 'AT', AS COMPARED TO LINE FROM LIGHT 1 TO LIGHT 3 *)
            V2(1)=-TRNAT(1,2)
            V2(2)=-TRNAT(2,2)
            V2(3)=-TRNAT(3,2)
            CALL MMLT(V3,ACV,V2,3,3,1)
            ROTLOS=ATAN2(V3(3),V3(2))-ATAN2(V(3)-V(1),U(1)-U(3))
C         (* FORM QUATERNION FOR CORRECTING ROTATION *)
            CALL MSCL(QC,RELPOS,3,1,SIN( 5*ROTLOS)/SQRT(RELPOS(1)**2+
     A          RELPOS(2)**2+RELPOS(3)**2))
            QC(4)=COS( 5*ROTLOS)
C         (* COMPUTE NET ROTATION OF QT1 FOLLOWED BY QC *)
            QT(1)= QC(4)*QT1(1)+QC(3)*QT1(2)-QC(2)*QT1(3)+QC(1)*QT1(4)
            QT(2)=-QC(3)*QT1(1)+QC(3)*QT1(2)+QC(1)*QT1(3)+QC(2)*QT1(4)
            QT(3)= QC(2)*QT1(1)-QC(1)*QT1(2)+QC(4)*QT1(3)+QC(3)*QT1(4)
            QT(4)=-QC(1)*QT1(1)-QC(2)*QT1(2)-QC(3)*QT1(3)+QC(4)*QT1(4)
      RETURN
C
      END
```



```
C
      SUBROUTINE INCORP(CVPOS,P,ESTATE)
C     (* 450 - INCORPORATE MEASUREMENT *)
C
C     CALLS
C        UPDCOV,UPDSTA,COMPG,ESTCOV,KALGAN
C     CALLED BY
C        DOCK
C
C     INPUT
C        ESTATE,P,R
C     OUTPUT
C        P,ESTATE
C
      REAL CVPOS(3)
C        MEASURED CHASE VEHICLE POSITON IN PRIMARY REFERENCE FRAME
      REAL ESTATE(6)
C        ESTIMATE OF STATE VECTOR
      REAL G(3,6)
C        PARTIAL OF MEASUREMENT WITH RESPECT TO STATE
      REAL KGAIN(6,3)
C        KALMAN GAIN MATRIX
      REAL P(6,6)
C        COVARIANCE OF STATE ESTIMATE
      REAL R(3,3)
C        MEASUREMENT COVARIANCE
C
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* 451 - COMPUTE JACOBIAN G *)
         CALL COMPG(ESTATE,G)
C     (* 452 - ESTIMATE MEASUREMENT COVARIANCE *)
         CALL ESTCOV(ESTATE,R)
C     (* 453 - COMPUTE KALMAN GAIN MATRIX *)
         CALL KALGAN(R,P,G,KGAIN)
C     (* 454 - UPDATE STATE *)
         CALL UPDSTA(ESTATE,KGAIN,CVPOS)
C     (* 455 - UPDATE COVARIANCE MATRIX *)
         CALL UPDCOV(P,KGAIN,G)
      RETURN
C
      END
```

ORIGINAL PAGE IS OF POOR QUALITY

```
C
      SUBROUTINE COMPG(ESTATE,G)
C     (* 451 - COMPUTE PARTIAL OF MEASUREMENT WITH RESPECT TO STATE *)
C
C     CALLS
C        <NONE>
C     CALLED BY
C        INCORP
C
C     INPUT
C        ESTATE
C     OUTPUT
C        G
C
      REAL ESTATE(6)
C        STATE ESTIMATE
      REAL G(3,6)
C        JACOBIAN
      INTEGER I,J
C        DO LOOP INDICES
C- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* COMPUTE G *)
         DO 10 I=1,3
            DO 10 J=1,6
               G(I,J)=0
   10    CONTINUE
         DO 20 I=1,3
            G(I,I)=1
   20    CONTINUE
      RETURN
C
      END
```



```
C
      SUBROUTINE ESTCOV(ESTATE,R)
C     (* 452 - ESTIMATE MEASUREMENT COVARIANCE *)
C
C     CALLS
C        <NONE>
C     CALLED BY
C        INCORP
C
C     INPUT
C        ESTATE
C     OUTPUT
C        R
C
      REAL ESTATE(6)
C        ESTIMATE OF STATE VECTOR
      INTEGER I,J
C        DO LOOP INDICES
      REAL R(3,3)
C        MEASUREMENT COVARIANCE
      REAL VARNCE
C        ESTIMATED VARIANCE (PER AXIS)
C- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      DO 10 I=1,3
         DO 10 J=1,3
            R(I,J)=0 0
   10 CONTINUE
      VARNCE= 818E-6*(ESTATE(1)**2+ESTATE(2)**2+ESTATE(3)* 2)**2+ 01
      DO 20 I=1,3
         R(I,I)=VARNCE
   20 CONTINUE
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE KALGAN(R,P,G,KGAIN)
C     (* 453 - COMPUTE KALMAN GAIN MATRIX *)
C
C     CALLS
C        MADD,MINV,MMLT
C     CALLED BY
C        INCORP
C
C     INPUT
C        R,P,G
C     OUTPUT
C        KGAIN
C
      INTEGER ERR
C        ERROR CODE (=0 IF NO ERROR)
      REAL G(3,6)
C        PARTIAL OF MEASUREMENT WITH RESPECT TO STATE
      REAL GT(6,3)
C        TRANSPOSE OF G
      INTEGER I,J
C        DO LOOP INDICES
      REAL KGAIN(6,3)
C        KALMAN GAIN MATRIX
      REAL P(6,6)
C        STATE ESTIMATE COVARIANCE
      REAL R(3,3)
C        MEASUREMENT COVARIANCE
      REAL SCRACH(4,6)
C        SCRATCHPAD FOR ROUTINE MINV
      REAL TEMP1(6,3),TEMP2(3,3),TEMP3(3,3)
C        INTERMEDIATE RESULTS WITH NO PROBLEM-RELATED SIGNIFICANCE
      INTEGER TERMNL
C        FORTRAN UNIT NUMBER FOR TERMINAL
C
      COMMON/SIMUL/TERMNL,IDUMMY
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* COMPUTE KGAIN=P*TRN(G)*INV(R+G*P*TRN(G)) *)
         DO 10 I=1,3
            DO 10 J=1,6
               GT(J,I)=G(I,J)
   10    CONTINUE
         CALL MMLT(TEMP1,P,GT,6,6,3)
         CALL MMLT(TEMP2,G,TEMP1,3,6,3)
         CALL MADD(TEMP3,R,TEMP2,3,3)
         CALL MINV(TEMP2,TEMP3,3,SCRACH,4,6,ERR)
         IF(ERR NE 0) GO TO 20
         CALL MMLT(TEMP1,GT,TEMP2,6,3,3)
         CALL MMLT(KGAIN,P,TEMP1,6,6,3)
         RETURN
C
   20 CONTINUE
C     (* REPORT ERROR *)
         WRITE(TERMNL,901)
         STOP
C
  901 FORMAT('MATRIX INVERSION FAILURE IN SUBROUTINE KALGAN')
C
      END
```



```
C
      SUBROUTINE UPDSTA(ESTATE,KGAIN,CVPOS)
C     (* 454 - UPDATE STATE ESTIMATE *)
C
C     CALLS
C        MMLT
C     CALLED BY
C        INCORP
C
C     INPUT
C        RELPOS,AT,ESTATE
C     OUTPUT
C        ESTATE
C
      REAL CVPOS(3)
C        MEASURED CHASE VEHICLE POSITION IN PRIMARY REFERENCE FRAME
      REAL DELTA(6)
C        ADJUSTMENT TO ESTATE
      REAL ESTATE(6)
C        ESTIMATE OF STATE VECTOR
      INTEGER I
C        DO LOOP INDEX
      REAL KGAIN(6,3)
C        KALMAN GAIN MATRIX
      REAL POSERR(3)
C        MEASURED POSITION MINUS PREDICTED POSITION
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* COMPUTE ESTATE=ESTATE+KGAIN*(CVPOS-PREDICTED POSITION) *)
         DO 10 I=1,3
            POSERR(I)=CVPOS(I)-ESTATE(I)
   10    CONTINUE
         CALL MMLT(DELTA,KGAIN,POSERR,6,3,1)
         DO 20 I=1,6
            ESTATE(I)=ESTATE(I)+DELTA(I)
   20    CONTINUE
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE UPDCOV(P,KGAIN,G)
C     (* 455 - UPDATE COVARIANCE MATRIX FROM (P=(I-K*G)*P)   *)
C
C     CALLS
C        MMLT
C     CALLED BY
C        INCORP
C
C     INPUT
C        P,KGAIN,G
C     OUTPUT
C        P
C
      REAL G(3,6)
C        PARTIAL OF MEASUREMENT WITH RESPECT TO STATE
      INTEGER I,J
C        DO LOOP INDICES
      REAL KGAIN(6,3)
C        KALMAN GAIN MATRIX
      REAL P(6,6)
C        COVARIANCE OF STATE ESTIMATE
      REAL TEMP(6,6),P1(6,6)
C        INTERMEDIATE RESULTS WITH NO PROBLEM-RELATED SIGNIFICANCE
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* COMPUTE I-KGAIN*G *)
      CALL MMLT(TEMP,KGAIN,G,6,3,6)
      DO 10 I=1,6
         DO 10 J=1,6
            TEMP(I,J)=-TEMP(I,J)
   10 CONTINUE
      DO 20 I=1,6
         TEMP(I,I)=TEMP(I,I)+1.0
   20 CONTINUE
C     (* COMPUTE NEW P *)
      CALL MMLT(P1,TEMP,P,6,6,6)
      DO 30 I=1,6
         DO 30 J=1,6
            P(I,J)=P1(I,J)
   30 CONTINUE
      RETURN
C
      END
```



```
C
      SUBROUTINE PROPES(P,ESTATE,F,STEP,ACVT)
C     (* 460 - PROPAGATE STATE ESTIMATE AND COVARIANCE *)
C
C     CALLS
C        MMLT,MSCL
C     CALLED BY
C        DOCK
C
C     INPUT
C        P,ESTATE,ACVT,F,STEP
C     OUTPUT
C        P,ESTATE
C
      REAL ACCEL(3)
C        ESTIMATED ACCELERATION
      REAL ACVT(3,3)
C        TRANSPOSE OF CHASE VEHICLE DIRECTION COSINE MATRIX
      REAL AK2(3,14)
C        MATRIX RELATING FORCES FROM INDIVIDUAL THRUSTERS TO NET FORCE VECTOR
      REAL AMAGSQ
C        SQUARE OF MAGNITUDE OF TOTAL ACCELERATION
      REAL AVAR
C        ESTIMATED ACCELERATION VARIANCE
      REAL DSTATE(6)
C        CHANGE IN STATE
      REAL DP(6,6)
C        CHANGE IN COVARIANCE
      REAL ESTATE(6)
C        ESTIMATED STATE
      REAL F(14)
C        FORCES FROM THRUSTERS AFTER SELECTION
      REAL FBOD(3)
C        FORCES ALONG CHASE VEHICLE BODY AXES
      REAL FORCL(3)
C        FORCES IN PRIMARY REFERENCE FRAME
      INTEGER I,J
C        DO LOOP INDICES
      REAL P(6,6)
C        COVARIANCE OF STATE ESTIMATE
      REAL STEP
C        INTEGRATION TIME STEP
      PARAMETER AVGMAS=3650
C        ESTIMATED CHASE VEHICLE MASS
C
      COMMON/VEHIC/DUMMY1(14),AK2,DUMMY2(57)
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
C     (* COMPUTE FORCE=ACVT*AK2*F *)
         CALL MMLT(FBOD,AK2,F,3,14,1)
         CALL MMLT(FORCE,ACVT,FBOD,3,3,1)
C     (* ESTIMATE ACCELERATION *)
         CALL MSCL(ACCEL,FORCE,3,1,1./AVGMAS)
C     (* ESTIMATE ACCELERATION VARIANCE *)
         AMAGSQ=ACCEL(1)**2+ACCEL(2)**2+ACCEL(3)**2
         AVAR=0.05*AMAGSQ+1.E-7
C     (* FIND POSITION CHANGE *)
         DO 10 I=1,3
            DSTATE(I)=(ESTATE(I+3)+.5*ACCEL(I)*STEP)*STEP
   10    CONTINUE
C     (* FIND VELOCITY CHANGE *)
         DO 20 I=4,6
            DSTATE(I)=STEP*ACCEL(I-3)
   20    CONTINUE
C     (* FIND COVARIANCE CHANGE *)
         DO 30 I=4,6
            DO 30 J=1,6
               DP(I,J)=0.0
   30    CONTINUE
         DP(4,4)=AVAR
         DP(5,5)=AVAR
         DP(6,6)=AVAR
         DO 40 I=1,3
            DO 40 J=1,6
               DP(I,J)=P(I+3,J)
   40    CONTINUE
         DO 50 I=1,6
            DO 50 J=1,3
               DP(I,J)=DP(I,J)+P(I,J+3)
   50    CONTINUE
C     (* COMPUTE NEW STATE AND COVARIANCE *)
         DO 60 I=1,6
            ESTATE(I)=ESTATE(I)+DSTATE(I)
            DO 60 J=1,6
               P(I,J)=P(I,J)+DP(I,J)
   60    CONTINUE
      RETURN
C
      END
```



```
C
      SUBROUTINE SETGOL(ESTATE,P,T,ACV,AT,GXL,GXH,GYL,GYH,GZL,GZH,
     A   DEDLIN,VALID)
C     (* 470 - SET GOAL *)
C
C     CALLS
C        SQRT,AMAX1,MTRN,MMLT,MSUB
C     CALLED BY
C        DOCK
C
C     INPUT
C        ESTATE,P,DOKRAD,T,ACV,AT
C     OUTPUT
C        GXL,GXH,GYL,GYH,GZL,GZH,DEDLIN
C
      REAL ACV(3,3),AT(3,3),TRNAT(3,3)
C        DIRECTION COSINE MATRICES FOR CHASE VEHICLE & TARGET SPACECRAFT
C        AND TRANSPOSE OF TARGET DIRECTION COSINE MATRIX
      REAL D
C        3-SIGMA POSITION UNCERTAINTY
      REAL DEDLIN
C        TIME BY WHICH STRATEGY LOGIC WANTS CHASE VEHICLE IN GOAL 'BOX'
      REAL DOKRAD
C        RADIUS FROM TARGET SPACECRAFT WITHIN WHICH THE SPACECRAFT ARE
C        CONSIDERED DOCKED
      REAL ESTATE(6)
C        ESTIMATE OF STATE VECTOR
      REAL GXL,GXH,GYL,GYH,GZL,GZH
C        X,Y,Z LOWER AND UPPER LIMITS OF GOAL 'BOX'
      REAL HDC(3),HDT(3)
C        DOCKING FIXTURES' POSITIONS IN THEIR RESPECTIVE SPACECRAFT
C        COORDINATE SYSTEMS
      INTEGER I,J
C        DO LOOP INDICES
      REAL P(6,6)
C        COVARIANCE OF STATE ESTIMATE
      REAL SLOP
C        ALLOWED ERROR IN POSITION (PER AXIS)
      REAL T
C        ELAPSED TIME
      REAL V1(3),V2(3),V3(3),R(3),V4(3),V5(3)
C        INTERMEDIATE RESULTS
      LOGICAL VALID
C        TRUE IF LAST MEASUREMENT WAS VALID
C
      COMMON/VEHIC/DUMMY1(78),DOKRAD,DUMMY2(14)
      COMMON/HNGOFF/DUMMY3(6),HDC,HDT
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      IF(SQR1(ESTATE(1)**2+ESTATE(2)**2+ESTATE(3)**2) LT 12    AND
     A   SQR1(ESTATE(4)**2+ESTATE(5)**2+ESTATE(6)**2) LT  15)
     B   GO TO 10
         D=3 *SQR1(P(1,1)+P(2,2)+P(3,3))
         GO TO 20
   10 CONTINUE
         D=0
   20 CONTINUE
C     (* COMPUTE AIM POINT ON TARGET X AXIS *)
         R(1)=-D+HDT(1)
         R(2)=0 +HDT(2)
         R(3)=0 +HDT(3)
         CALL MMLT(V4,AT,ESTATE,3,3,1)
         IF(V4(2)**2+V4(3)**2 GT 225 0   OR    NOT  VALID) R(1)=-D-20
         CALL MSUB(V5,R,HDC,3,1)
C     (* CONVERT COORDINATES AND SUBTRACT CURRENT POSITION *)
         CALL MTRN(TRNAT,AT,3)
         CALL MMLT(V1,TRNAT,V5,3,3,1)
         CALL MSUB(V2,V1,ESTATE,3,1)
         CALL MMLT(V3,ACV,V2,3,3,1)
      SLOP=0 25*AMIN1( 1*ABS(V4(1)), 25)
      GXH=V3(1)
      IF(GXH LT 8 ) GXH=GXH+SLOP
      GYH=V3(2)+SLOP
      GZH=V3(3)+SLOP
      GXL=GXH-2 0*SLOP
      GYL=V3(2)-SLOP
      GZL=V3(3)-SLOP
      DEDLIN=T+ 125*SQRT(V3(1)**2+V3(2)**2+V3(3)**2)
      RETURN
C
      END
```



```
C
      SUBROUTINE THRUST(F,ESTATE,GXL,GXH,GYL,GYH,GZL,GZH,TLIM,ACV,
     A   ATRATE,RPYERR)
C     (* 480 -  SELECT THRUSTERS *)
C
C     CALLS
C        CNTLAW,DEFINF,FIRTHR,SELECT
C     CALLED BY
C        DOCK
C
C     INPUT
C        ESTATE,GXL,    ,GZH,TLIM,ESTATE,ACV,ATRATE,RPYERR
C     OUTPUT
C        F
C
      REAL ACV(3,3)
C        MEASURED CHASE VEHICLE ATTITUDE IN PRIMARY PEF FRAME
      REAL ATRATE(3)
C        MEASURED ATTITUDE RATES OF CHASE VEHICLE
      REAL E(14)
C        THRUSTER COMMAND ENTRIES FROM THRUSTER SELECT LOGIC
      REAL ESTATE(6)
C        STATE ESTIMATE
      REAL F(14)
C        FORCES FROM THRUSTERS AFTER SELECTION
      REAL F1(14)
C        LIST OF MAXIMUM THRUSTER FORCES
      REAL GXL,GXH,GYL,GYH,GZL,GZH
C        LOWER AND UPPER LIMITS OF GOAL 'BOX'
      INTEGER I
C        DO LOOP INDEX
      REAL ROTCMD(3),XLTCMD(3,
C        ROTATIONAL AND TRANSLATIONAL COMMANDS FROM CONTROL LAW
      REAL RPYERR(3)
C        MEASURED ATTITUDE ERROR IN ROLL, PITCH, AND YAW (RADIANS)
      REAL TLIM
C        TIME LIMIT
C
      COMMON/VEHIC/DUMMY(99),F1
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* 481 - USE CONTROL LAW TO DETERMINE NEEDED THRUST *)
         CALL CNTLAW(ROTCMD,XLTCMD,ESTATE,ACV,TLIM,GXL,GXH,GYL,GYH,GZL,
     A      GZH,ATRATE,RPYERR)
C     (* 482 - SELECT THRUSTER SET TO GIVE NEEDED THRUST *)
         CALL SELECT(ROTCMD,XLTCMD,E)
         DO 10 I=1,14
            F(I)=E(I)*F1(I)
   10    CONTINUE
C     (* 483 - FIRE THRUSTERS *)
         CALL FIRTHR(E)
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE CNTLAW(ROTCMD, XLTCMD, ESTATE, ACV, TLIM, GXL, GXH, GYL, G\H,
     A   GZL, GZH, ATRATE, RPYERR)
C     (* 481 - CONTROL LAW *)
C
C     CALLS
C         MMLT, ACCEL
C     CALLED BY
C         THRUST
C
C     INPUT
C         ESTATE, ACV, TLIM, GXL,     , GZH, ATRATE, RPYERR
C     OUTPUT
C         ROTCMD, XLTCMD
C
      REAL ACV(3,3)
C         MEASURED CHASE VEHICLE ATTITUDE IN PRIMARY REFERENCE FRAME
      REAL ATRATE(3)
C         MEASURED CHASE VEHICLE ATTIT'DE RATES
      REAL ESTATE(6)
C         STATE ESTIMATE
      REAL GXL, GXH, GYL, GYH, GZL, GZH
C         LOWER AND UPPER LIMITS OF GOAL 'BOX
      REAL ROTCMD(3), XLTCMD(3)
C         ROTATIONAL AND TRANSLATIONAL COMMANDS FROM CONTROL LAW
      REAL RPYERR(3)
C         MEASURED ATTITUDE ERROR IN ROLL, PITCH, AND YAW
      REAL TLIM
C         TIME LIMIT
      REAL VBOD(3)
C         VELOCITY IN CHASE VEHICLE BODY COORDINATE SYSTEM
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* CONVERT VELOCITY TO BODY COORD  SYS  *)
      CALL MMLT(VBOD, ACV, ESTATE(4), 3, 3, 1)
C     (* 481 1 - FIND IDEAL ACCELERATIONS *)
      XLTCMD(1)=ACCEL(VBOD(1), GXL, GX+ TLIM, 1 233333,  03,  1)
      XLTCMD(2)=ACCEL(VBOD(2), GYL, GYH, TLIM, 1 233333  03,  1)
      XLTCMD(3)=ACCEL(VBOD(3 , GZL, GZH, TLIM, 1 233333,  03,  1)
      ROTCMD(1)=ACCEL(ATRATE(1), (-RPYERR(1)- 02', (-RPYERR(1)+ 02),
     A   1 233333, 1 23333   02,  074)
      ROTCMD(2)=ACCEL(ATRATE(2), -RPYERR(2)- 02, -RPYERR(2)+ 02,
     A   1 233333, 1 233333,  02,  074)
      ROTCMD(3)=ACCEL(ATRATE(3), -RPYERR(3)- 02, -RPYERR(3)+ 02, TLIM,
     A   1 233333,  02  074)
      RETURN
C
      END
```



```
C
      FUNCTION ACCEL(X, XDOT, XMIN, XMAX, TLIM, DT, AMIN, AMAX)
C     (* 481 1 - ESTIMATE ACCELERATION FOR ONE AXIS *)
C
C     CALLS
C         <NONE>
C     CALLED BY
C         CNTLAW
C
C     INPUT
C         X, XDOT, XMIN, XMAX, TLIM, DT
C     OUTPUT
C         <FUNCTION VALUE ONLY>
C
      REAL AMIN, AMAX
C         MIN AND MAX RELATIVE THRUST
      REAL DT
C         TIME STEP BETWEEN DECISIONS
      REAL K1, K2, K3, K4
C         CONSTANTS BASED ON MIN  MAX POSSIBLE ACCELERATION
      REAL TLIM
C         TIME LIMIT - SECONDS UNTIL DEADLINE
      REAL XDOT
C         VELOCITY IN THIS AXIS
      REAL XMIN, XMAX
C         MIN, MAX ACCEPTABLE FINAL X VALUE
```

```
C
C
      K1= 5*AMAX*DT**2
      K2=AMAX*DT
      K3= 5/AMIN
      K4=AMIN* 5
      IF(XMAX GE 0 ) GO TO 50
         IF(XDOT LT 0 ) GO TO 10
            ACCEL=-1
            RETURN
  10     IF(XDOT*DT-K3*XDOT**2 GE XMIN) GO TO 20
            ACCEL=1
            RETURN
  20     IF(XDOT*DT-K1-K3*(K2-XDOT)**2 GE XMIN) GO TO 30
            ACCEL=0
            RETURN
  30     IF(XDOT*TLIM-K4*(TLIM-DT)**2 GT XMAX) GO TO 40
            ACCEL=0
            RETURN
  40     CONTINUE
            ACCEL=-1
            RETURN
  50  IF(XMIN LE 0 ) GO TO 100
         IF(XDOT GT 0 ) GO TO 60
            ACCEL=1
            RETURN
  60     IF(XDOT*DT+K3*XDOT**2 LE XMAX) GO TO 70
            ACCEL=-1
            RETURN
  70     IF(XDOT*DT+K1+K3*(K2+XDOT)**2 LE XMAX) GO TO 80
            ACCEL=0
            RETURN
  80     IF(XDOT*DT+K4*(TLIM-DT)**2 LT XMIN) GO TO 90
            ACCEL=0
            RETURN
  90     CONTINUE
            ACCEL=1
            RETURN
 100  IF(XDOT LE 0 ) GO TO 120
         IF(XDOT*DT+K3*XDOT**2 LE XMAX) GO TO 110
            ACCEL=-1
            RETURN
 110     CONTINUE
            ACCEL=0
            RETURN
 120  IF(XDOT*DT-K3*XDOT**2 GE XMIN) GO TO 130
         ACCEL=1
         RETURN
 130  CONTINUE
         ACCEL=0
         RETURN
C
      END
```



```
C
      SUBROUTINE SELECT(ROTCMD,XLTCMD,E)
C     (* 482 - FROM CONTROL LAW OUTPUTS SELECT WHICH THRUSTERS TO FIRE *)
C
C     CALLS
C        TABLE1,TABLE2,TABLE3
C     CALLED BY
C        THRUST
C
C     INPUT
C        ROTCMD,XLTCMD
C     OUTPUT
C        E
C
      REAL E(14)
C        THRUSTER COMMAND ENTRIES FROM THRUSTER SELECT LOGIC
      REAL ROTCMD(3),XLTCMD(3)
C        ROTATIONAL AND TRANSLATIONAL COMMANDS FROM CONTROL LAW
      INTEGER I,J
C        DO LOOP INDICES
      INTEGER INDEX1,INDEX2,INDEX3
C       INDICES FOR COMMAND TABLES
      INTEGER ROTCOD(3),TRLCOD(3)
C        ROTATION AND TRANSLATION CODES USED TO COMPUTE INDICES
C        FOR TABLES
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* DETERMINE CODE FOR THRUSTER ACTIVATION *)
         DO 40 I=1,3
            IF(XLTCMD(I)) 10,20,30
  10        TRLCOD(I)=1
            GO TO 40
  20        TRLCOD(I)=0
            GO TO 40
  30        TRLCOD(I)=2
  40     CONTINUE
         DO 80 I=1,3
            IF(ROTCMD(I)) 50,60,70
  50        ROTCOD(I)=1
            GO TO 80
  60        ROTCOD(I)=0
            GO TO 80
  70        ROTCOD(I)=2
  80     CONTINUE
C     (* ZERO OUT THRUSTER COMMANDS *)
         DO 90 J=1,14
            E(J)=0 0
  90     CONTINUE
C     (* DETERMINE WHICH TABLE TO CONSULT FOR THRUSTER COMMANDS *)
         INDEX1=TRLCOD(1)+3*ROTCOD(2)+9*ROTCOD(3)
         INDEX2=TRLCOD(2)+3*ROTCOD(1)+9*ROTCOD(3)
         INDEX3=TRLCOD(3)+3*ROTCOD(1)+9*ROTCOD(2)
C     (* 482 1 THRU 482 3 - SELECT THRUSTER SET *)
         IF(INDEX1 EQ 0) GO TO 100
            CALL TABLE1(INDEX1,E)
 100     CONTINUE
         IF(INDEX2 EQ 0) GO TO 110
            CALL TABLE2(INDEX2,E)
 110     CONTINUE
         IF(INDEX3 EQ 0) GO TO 120
            CALL TABLE3(INDEX3,E)
 120     CONTINUE
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE TABLE1(INDEX1,E)
C     (* 4B2 1 -  FIND COMINATION OF TABLE ENTRIES THAT SATISFY THRUSTER
C        SELECTION REQUIREMENTS *)
C
C     CALLS
C        <NONE>
C     CALLED BY
C        SELECT
C
C     INPUT
C        INDEX1
C     OUTPUT
C        E
C
      REAL E(14)
C        THRUSTER COMMAND ENTRIES FROM THRUSTER SELECT LOGIC
      INTEGER INDEX1
C        THRUSTER TABLE INDEX
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* DO SELECTION BASED ON INDEX *)
      GOTO(10,20,30,40,50,60,70,80,90,100,110,120,130,140,150,160,170,
     A   180,190,200,210,220,230,240,250,260), INDEX1
   10 E(1)=1.0
      GO TO 300
   20 E(14)=1.0
      GO TO 300
   30 E(10)=1.0
      E(13)=1.0
      GO TO 300
   40 E(1)=1.0
      E(10)=1.0
      E(13)=1.0
      GO TO 300
   50 E(10)=1.0
      E(13)=1.0
      E(14)=1.0
      GO TO 300
   60 E(11)=1.0
      E(12)=1.0
      GO TO 300
   70 E(1)=1.0
      E(11)=1.0
      E(12)=1.0
      GO TO 300
   80 E(11)=1.0
      E(12)=1.0
      E(14)=1.0
      GO TO 300
   90 E(2)=1.0
      E(4)=1.0
      GO TO 300
  100 E(1)=1.0
      E(2)=1.0
      E(4)=1.0
      GO TO 300
  110 E(2)=1.0
      E(4)=1.0
      E(14)=1.0
      GO TO 300
  120 E(2)=1.0
      E(4)=1.0
      E(10)=1.0
      E(13)=1.0
      GO TO 300
  130 E(1)=1.0
      E(2)=1.0
      E(4)=1.0
      GO TO 300
  140 E(2)=1.0
      E(4)=1.0
      E(10)=1.0
      E(13)=1.0
      E(14)=1.0
      GO TO 300
```



```
  150 E(2)=1.0
      E(4)=1.0
      E(11)=1.0
      E(12)=1.0
      GO TO 300
  160 E(1)=1.0
      E(2)=1.0
      E(4)=1.0
      E(11)=1.0
      E(12)=1.0
      GO TO 300
  170 E(2)=1.0
      E(4)=1.0
      E(11)=1.0
      E(12)=1.0
      E(14)=1.0
      GO TO 300
  180 E(3)=1.0
      E(5)=1.0
      GO TO 300
  190 E(1)=1.0
      E(3)=1.0
      E(5)=1.0
      GO TO 300
  200 E(3)=1.0
      E(5)=1.0
      E(14)=1.0
      GO TO 300
  210 E(3)=1.0
      E(5)=1.0
      E(10)=1.0
      E(13)=1.0
      GO TO 300
  220 E(1)=1.0
      E(3)=1.0
      E(5)=1.0
      E(10)=1.0
      E(13)=1.0
      GO TO 300
  230 E(3)=1.0
      E(5)=1.0
      E(10)=1.0
      E(13)=1.0
      E(14)=1.0
      GO TO 300
  240 E(3)=1.0
      E(5)=1.0
      E(11)=1.0
      E(12)=1.0
      GO TO 300
  250 E(1)=1.0
      E(3)=1.0
      E(5)=1.0
      E(11)=1.0
      E(12)=1.0
      GO TO 300
  260 E(3)=1.0
      E(5)=1.0
      E(11)=1.0
      E(12)=1.0
      E(14)=1.0
  300 RETURN
C
      END
```

ORIGINAL PAGE IS OF POOR QUALITY

```
C
      SUBROUTINE TABLE2(INDEX2,E)
C     (* 482 2 -  FIND COMINATION OF TABLE EN RIES THAT SATISFY THRUSTE
C        SELECTION REQUIREMENTS *)
C
C     CALLS
C        <NONE>
C     CALLED BY
C        SELECT
C
C     INPUT
C        INDEX2
C     OUTPUT
C        E
C
      REAL E(14)
C        THRUSTER COMMAND ENTRIES FROM THRUSTER SELECT LOGIC
      INTEGER INDEX2
C        THRUSTER TABLE INDEX
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* DO SELECTION BASED ON INDEX *)
      GOTO(10,20,30,40,50,60,70,80,90,100,110,120,130,140,150,160,170,
     A     180,190,200,210,220,230,240,250,260),INDEX2
   10 E(4)=1.0
      E(5)=1.0
      GO TO 300
   20 E(2)=1.0
      E(3)=1.0
      GO TO 300
   30 E(7)=1.0
      E(8)=1.0
      GO TO 300
   40 E(4)=1.0
      E(5)=1.0
      E(7)=1.0
      E(8)=1.0
      GO TO 300
   50 E(2)=1.0
      E(3)=1.0
      E(7)=1.0
      E(8)=1.0
      GO TO 300
   60 E(6)=1.0
      E(9)=1.0
      GO TU 300
   70 E(4)=1.0
      E(5)=1.0
      E(6)=1.0
      E(9)=1.0
      GO TO 300
   80 E(2)=1.0
      E(3)=1.0
      E(6)=1.0
      E(9)=1.0
      GO TO 300
   90 E(2)=1.0
      E(4)=1.0
      GO TO 300
  100 E(2)=1.0
      E(4)=1.0
      E(6)=1.0
      E(8)=1.0
      GO TO 300
  110 E(2)=1.0
      E(4)=1.0
      E(7)=1.0
      E(9)=1.0
      GO TO 300
  120 E(2)=1.0
      E(4)=1.0
      E(7)=1.0
      E(8)=1.0
      GO TO 300
  130 E(4)=1.0
      E(8)=1.0
```



```
      GO TO 300
  140 E(2)=1.0
      E(7)=1.0
      GO TO 300
  150 E(2)=1.0
      E(4)=1.0
      E(6)=1.0
      E(9)=1.0
      GO TO 300
  160 E(4)=1.0
      E(6)=1.0
      GO TO 300
  170 E(2)=1.0
      E(9)=1.0
      GO TO 300
  180 E(3)=1.0
      E(5)=1.0
      GO TO 300
  190 E(3)=1.0
      E(5)=1.0
      E(6)=1.0
      E(8)=1.0
      GO TO 300
  200 E(3)=1.0
      E(5)=1.0
      E(7)=1.0
      E(9)=1.0
      GO TO 300
  210 E(3)=1.0
      E(5)=1.0
      E(7)=1.0
      E(8)=1.0
      GO TO 300
  220 E(5)=1.0
      E(8)=1.0
      GO TO 300
  230 E(3)=1.0
      E(7)=1.0
      GO TO 300
  240 E(3)=1.0
      E(5)=1.0
      E(6)=1.0
      E(9)=1.0
      GO TO 300
  250 E(5)=1.0
      E(6)=1.0
      GO TO 300
  26? E(3)=1.0
      E(9)=1.0
  300 RETURN
C
      END
```

```
C
      SUBROUTINE TABLE3(INDEX3,E)
C     (* 4B2 3 -  FIND COMINATION OF TABLE ENTRIES THAT SATISFY THRUSTER
C        SELECTION REQUIREMENTS *)
C
C     CALLS
C        <NONE>
C     CALLED BY
C        SELECT
C
C     INPUT
C        INDEX3
C     OUTPUT
C        E
C
      REAL E(14)
C        COMMAND ENTRIES FROM THRUSTER SELECT LOGIC
      INTEGER INDEX3
C        THRUSTER TABLE INDEX
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* DO SELECTION BASED ON INDEX *)
      GOTO(10,20,30,40,50,60,70,80,90,100,110,120,130,140,150,160,170,
     A   180,190,200,210,220,230,240,250,260), INDEX3
   10 E(10)=1.0
      E(11)=1.0
      GO TO 300
   20 E(12)=1.0
      E(13)=1.0
      GO TO 300
   30 E(7)=1.0
      E(8)=1.0
      GO TO 300
   40 E(7)=1.0
      E(8)=1.0
      E(10)=1.0
      E(11)=1.0
      GO TO 300
   50 E(7)=1.0
      E(8)=1.0
      GO TO 300
   60 E(6)=1.0
      E(9)=1.0
      GO TO 300
   70 E(6)=1.0
      E(9)=1.0
      E(10)=1.0
      E(11)=1.0
      GO TO 300
   80 E(6)=1.0
      E(9)=1.0
      E(12)=1.0
      E(13)=1.0
      GO TO 300
   90 E(10)=1.0
      E(13)=1.0
      GO TO 300
  100 E(10)=1.0
      GO TO 300
  110 E(13)=1.0
      GO TO 300
  120 E(7)=1.0
      E(8)=1.0
      E(10)=1.0
      E(13)=1.0
      GO TO 300
  130 E(7)=1.0
      E(8)=1.0
      E(10)=1.0
      GO TO 300
  140 E(7)=1.0
      E(8)=1.0
      E(13)=1.0
      GO TO 300
  150 E(6)=1.0
      E(9)=1.0
```



```
      E(10)=1.0
      E(13)=1.0
      GO TO 300
  160 E(6)=1.0
      E(9)=1.0
      E(10)=1.0
      GO TO 300
  170 E(6)=1.0
      E(9)=1.0
      E(13)=1.0
      GO TO 300
  180 E(11)=1.0
      E(12)=1.0
      GO TO 300
  190 E(11)=1.0
      GO TO 300
  200 E(12)=1.0
      GO TO 300
  210 E(7)=1.0
      E(8)=1.0
      E(11)=1.0
      E(12)=1.0
      GO TO 300
  220 E(7)=1.0
      E(8)=1.0
      E(11)=1.0
      GO TO 300
  230 E(7)=1.0
      E(8)=1.0
      E(12)=1.0
      GO TO 300
  240 E(6)=1.0
      E(9)=1.0
      E(11)=1.0
      E(12)=1.0
      GO TO 300
  250 E(6)=1.0
      E(9)=1.0
      E(12)=1.0
      GO TO 300
  260 E(6)=1.0
      E(9)=1.0
      E(12)=1.0
  300 RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE FIRTHR(E)
C     (* 483 - PUT THRUSTER COMMANDS IN COMMAND PORTS *)
C
C     CALLS
C        <NONE>
C     CALLED BY
C        THRUST
C
C     INPUT
C        E
C     OUTPUT
C        <NONE>
C
      REAL E(14)
C        THRUSTER COMMAND ENTRIES FROM THRUSTER SELECT LOGIC
C
C     (* TEMPORARY DUMMY SUBROUTINE *)
      RETURN
C
      END
```



```
C
      SUBROUTINE IMU(STATE,ACV,ACVT,ATRATE)
C     (* 490 - SIMULATE IMU MEASUREMENT *)
C
C     CALLS
C        DIRMAT,MADD,MSCL,ANGVEC
C     CALLED BY
C        DOCK
C
C     INPUT
C        QO,STATE,FULDIS,INERCV,MEMPTY
C     OUTPUT
C        ACV,ACVT,ATRATE
C
      REAL ACV(3,3),ACVT(3,3)
C        DIRECTION COSINE MATRIX AND ITS TRANSPOSE FOR 'MEASURED' ATTITUDE
C        OF CHASE VEHICLE WITH RESPECT TO PRIMARY REFERENCE FRAME
      REAL ATRATE(3)
C        MEASURED ATTITUDE RATES ABOUT CHASE VEHICLE AXES
      REAL FULDIS(3,3)
C        TENSOR DESCRIBING FUEL DISTRIBUTION
      REAL INERCV(3,3)
C        TENSOR DESCRIBING INERTIA OF EMPTY CHASE VEHICLE
      REAL INERFL(3,3)
C        FUEL INERTIA TENSOR
      REAL INERTA(3,3)
C        INERTIA OF CHASE VEHICLE (LOADED)
      REAL MEMPTY
C        MASS OF EMPTY CHASE VEHICLE
      REAL QO(4),QP(4)
C        INITIAL CHASE VEHICLE ATTITUDE QUATERNION IN 'TRUTH' REFERENCE FRAME
C        AND CURRENT PRIMARY REFERENCE FRAME, RESPECTIVELY
      REAL STATE(14)
C        TRUE CHASE VEHICLE STATE
      REAL TACV(3,3),TACVT(3,3)
C        CHASE VEHICLE DIRECTION COSINE MATRIX WITH RESPECT TO TRUTH
C        COORDINATE SYSTEM AND INVERSE OF THIS MATRIX
C
      COMMON/REF/QO
      COMMON/MASPRP/FULDIS,INERCV,MEMPTY
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* SUBTRACT INITIAL ATTITUDE FROM CURRENT ATTITUDE TO GET CURRENT
C        ATTITUDE IN PRIMARY REF  FRAME *)
         QP(1)=QO(4)*STATE(10)+QO(3)*STATE(11)-QO(2)*STATE(12)-
     A        QO(1)*STATE(13)
         QP(2)=-QO(3)*STATE(10)+QO(4)*STATE(11)+QO(1)*STATE(12)-
     A        QO(2)*STATE(13)
         QP(3)=QO(2)*STATE(10)-QO(1)*STATE(11)+QO(4)*STATE(12)-
     A        QO(3)*STATE(13)
         QP(4)=QO(1)*STATE(10)+QO(2)*STATE(11)+QO(3)*STATE(12)+
     A        QO(4)*STATE(13)
C     (* 901 - CONVERT QUATERNIONS TO DIRECTION COSINE MATRICES *)
         CALL DIRMAT(QP,ACV,ACVT)
         CALL DIRMAT(STATE(10),TACV,TACVT)
C     (* DETERMINE INERTIA *)
         CALL MSCL(INERFL,FULDIS,3,3,STATE(14)-MEMPTY)
         CALL MADD(INERTA,INERCV,INERFL,3,3)
C     (* 904 - FIND ANGULAR VELOCITY VECTOR *)
         CALL ANGVEC(INERTA,STATE(7),ATRATE,TACV)
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE RPY(U,V,RPYERR)
C     (* 4A0 - CALCULATE ROLL, PITCH, AND YAW ERRORS *)
C
C     CALLS
C        ATAN2
C     CALLED BY
C        DOCK
C
C     INPUT
C        U,V
C     OUTPUT
C        RPYERR
C
      REAL RPYERR(3)
C        ROLL, PITCH AND YAW ERRORS (RADIANS)
      REAL U(3),V(3)
C        LAMP IMAGE CENTROIDS FROM TELEVISION CAMERA
C
      COMMON/OPTSYS/DUMMY1(2),FOCLEN,DUMMY2(2)
C
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* CALCULATE PITCH ERROR *)
         RPYERR(2)=ATAN2(V(2),FOCLEN)
C     (* CALCULATE YAW ERROR *)
         RPYERR(3)=ATAN2(U(2),FOCLEN)
C     (* CALCULATE ROLL ERROR *)
         RPYERR(1)=ATAN2(V(3)-V(1),U(3)-U(1))
      RETURN
C
      END
```



```
C
      SUBROUTINE ESTRPY(ESTATE,ACVT,RPYERR)
C     (* 4B0 - ESTIMATE ATTITUDE ERROR FROM STATE ESTIMATE *)
C
C     CALLS
C        ATAN2,ASIN,MSCL
C     CALLED BY
C        DOCK
C
C     INPUT
C        ESTATE,ACVT
C     OUTPUT
C        RPYERR
C
      REAL ACVT(3,3)
C        TRANSPOSE OF CHASE VEHICLE DIRECTION COSINE MATRIX (WITH RESPECT
C        TO PRIMARY REFERENCE FRAME)
      REAL DOTPRD
C        DOT PRODUCT OF VECTORS DEFINING LINE TO TARGET AND VEHICLE X AXIS
      REAL ESTATE(6)
C        ESTIMATED STATE
      REAL PHI
C        EULER ERROR ANGLE
      REAL PRDMAG
C        MAGNITUDE OF CROSS PRODUCT OF VECTORS DEFINING LINE TO TARGET AND
C        VEHICLE X AXIS
      REAL Q(4)
C        ATTITUDE ERROR QUATERNION
      REAL R1(3),R2(3)
C        INTERMEDIATE RESULTS
      REAL RPYERR(3)
C        ERRORS IN ROLL, PITCH AND YAW, RESPECTIVELY (RADIANS)
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      R1(1)=-ESTATE(2)*ACVT(3,1)+ESTATE(3)*ACVT(2,1)
      R1(2)=-ESTATE(3)*ACVT(1,1)+ESTATE(1)*ACVT(3,1)
      R1(3)=-ESTATE(1)*ACVT(2,1)+ESTATE(2)*ACVT(1,1)
      PRDMAG=SQRT(R1(1)**2+R1(2)**2+R1(3)**2)
      DOTPRD=ESTATE(1)*ACVT(1,1)+ESTATE(2)*ACVT(2,1)+ESTATE(3)*
     A   ACVT(3,1)
      IF(PRDMAG GT 0 ) GO TO 30
         Q(1)=0
         Q(2)=0
         IF(DOTPRD GT 0 ) GO TO 10
            Q(3)=1
            Q(4)=0 0
            GO TO 20
   10    CONTINUE
            Q(3)=0 0
            Q(4)=1 0
   20    CONTINUE
         GO TO 40
   30 CONTINUE
         CALL MSCL(R2,R1,3,1,1 /PRDMAG)
         PHI=ATAN2(PRDMAG,DOTPRD)
         CALL MSCL(Q,R2,3,1,SIN(PHI* 5))
         Q(4)=COS(PHI* 5)
   40 CONTINUE
      RPYERR(1)=0
      RPYERR(2)=-ASIN(2 *(Q(1)*Q(3)-Q(2)*Q(4)))
      IF(ABS(RPYERR(2)) GT 1 57) GO TO 50
         RPYERR(3)=ATAN2(2 *(Q(1)*Q(2)+Q(3)*Q(4)),Q(1)**2-Q(2)**2
     A      -Q(3)**2+Q(4)**2)
         GO TO 60
   50 CONTINUE
         RPYERR(3)=0
   60 CONTINUE
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE DIRMAT(Q,A,TRNA)
C     (* 901 - COMPUTE A DIRECTION COSINE MATRIX FROM A QUATERNION *)
C
C     CALLS
C        MTRN
C     CALLED BY
C        IMU,ATITUD,STPRIM,DOCK,FLASH,QUATRN,FINDCV
C
C     INPUT
C        Q
C     OUTPUT
C        A,TRNA
C
      REAL A(3,3)
C        DIRECTION COSINE MATRIX FORMED FROM QUATERNION Q
      REAL TRNA(3,3)
C        TRANPOSE OF A
      REAL Q(4)
C        QUATERNION
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C
C     (* COMPUTE ELEMENTS OF A *)
         A(1,1)=Q(1)**2-Q(2)**2-Q(3)**2+Q(4)**2
         A(1,2)=2 *(Q(1)*Q(2)+Q(3)*Q(4))
         A(1,3)=2 *(Q(1)*Q(3)-Q(2)*Q(4))
         A(2,1)=2 *(Q(1)*Q(2)-Q(3)*Q(4))
         A(2,2)=-Q(1)**2+Q(2)**2-Q(3)**2+Q(4)**2
         A(2,3)=2 *(Q(2)*Q(3)+Q(1)*Q(4))
         A(3,1)=2 *(Q(1)*Q(3)+Q(2)*Q(4))
         A(3,2)=2 *(Q(2)*Q(3)-Q(1)*Q(4))
         A(3,3)=-Q(1)**2-Q(2)**2+Q(3)**2+Q(4)**2
C     (* COMPUTE TRANSPOSE OF A *)
         CALL MTRN(TRNA,A,3)
      RETURN
C
      END
```



```
C
      SUBROUTINE STPRIM(STATE,F,INERTA,T,DSTATE)
C     (* 902 - DETERMINE TIME DERIVATIVE OF STATE VECTOR *)
C
C     CALLS:
C        ANGVEC,FORCE,LINACL,LPRIME,MAKROT,MPRIME,QPRIME,TORQUE,DIRMAT
C     CALLED BY:
C        COMPK1,COMPK2,COMPK3,COMPK4
C
C     INPUT:
C        F,T,STATE,INERTA
C     OUTPUT:
C        DSTATE
C
      REAL A(3,3)
C        'TRUE' DIRECTION COSINE MATRIX FOR CHASE VEHICLE
      REAL BODVEL(3)
C        ANGULAR VELOCITY ABOUT AXES OF CHASE VEHICLE
      REAL DSTATE(14)
C        TIME DERIVATIVE OF STATE VECTOR
      REAL F(14)
C        FORCES FROM CHASE VEHICLE THRUSTERS
      REAL INERTA(3,3)
C        INERTIA OF CASE VEHICLE (INCL. FUEL)
      REAL N(3)
C        TORQUE ON SPACECRAFT ABOUT ITS CENTER OF MASS
      REAL NTFORC(3)
C        NET FORCE FROM THRUSTER OPERATION
      REAL OMEGA(4,4)
C        ROTATION MATRIX OMEGA, TO BE APPLIED TO QUATERNION MATRIX 'Q'
C        TO FORM TIME DERIVATIVE OF Q (Q=ELEMENTS 10-13 OF STATE)
      REAL STATE(14)
C        CHASE VEHICLE STATE VECTOR
      REAL T
C        ELAPSED TIME
      REAL TRNA(3,3)
C        TRANSPOSE OF 'A'
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* 901 - COMPUTE A,TRNA FROM QUATERNION *)
         CALL DIRMAT(STATE(10),A,TRNA)
C     (* 902.1 - DETERMINE NET FORCE FROM THRUSTERS IN 'TRUTH' COORD.SYS *)
         CALL FORCE(F,NTFORC,TRNA)
C     (* 902.2 - COMPUTE TIME DERIVATIVE OF SPACECRAFT MASS *)
         CALL MPRIME(F,DSTATE(14))
C     (* 902.3 - COMPUTE LINEAR ACCELERATIONS *)
         CALL LINACL(T,STATE(14),NTFORC,STATE(1),DSTATE(4))
C     (* 904 - FIND ANGULAR VELOCITY VECTOR *)
         CALL ANGVEC(INERTA,STATE(7),BODVEL,A)
C     (* 902.4 - CALCULATE TORQUE ON SPACECRAFT FROM THRUSTER OPERATION *)
         CALL TORQUE(F,N,TRNA)
C     (* 902.5 - COMPUTE TIME DERIVATIVE OF ANGULAR MOMENTUM VECTOR *)
         CALL LPRIME(N,BODVEL,STATE(7),DSTATE(7),TRNA)
C     (* 902.6 - FORM ROTATION MATRIX OMEGA *)
         CALL MAKROT(BODVEL,OMEGA,A)
C     (* 902.7 - CALCULATE THE TIME DERIVATIVE OF QUATERNION *)
         CALL QPRIME(STATE(10),OMEGA,DSTATE(10))
C     (* DEFINE VELOCITY ELEMENTS OF DSTATE *)
         DSTATE(1)=STATE(4)
         DSTATE(2)=STATE(5)
         DSTATE(3)=STATE(6)
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE FORCE(F,NTFORC,TRNA)
C     (* 902 1 -   COMPUTE NET FORCE FROM THRUSTER OPERATION USING EQUATION
C            NET FORCE= A TRANSPOSE * AK2 * F *)
C
C     CALLS
C        MMLT
C     CALLED BY
C        STPRIM
C
C     INPUT
C        F,AK2,Q
C     OUTPUT
C        NTFORC
C
      REAL AK2(3,14
C        CONSTANT MATRIX RELATING INDIVIDUAL THRUSTER OUTPUTS
C        TO NET THRUST IN SPACECRAFT REFERENCE FRAME
C        (THIS ACCOMODATES MISALIGNMENT OF THRUSTERS)
      REAL B(3)
C         INTERMEDIATE RESULTS
      REAL F(14)
C        FORCE VECTOR OF CHASE VEHICLE AFTER THRUSTER SELECTION
      REAL NTFORC(3)
C        NET FORCE FROM THRUSTER OPERATION IN TRUTH COORD  SYS
      REAL TRNA(3,3)
C        TRANSPOSE OF DIRECTION COSINE MATRIX A
C
      COMMON/VEHIC/DUMMY1(14),AK2,DUMMY2(57)
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* COMPUTE NET FORCE *)
         CALL MMLT(B,AK2,F,3,14,1)
         CALL MMLT(NTFORC,TRNA,B,3,3,1)
      RETURN
C
      END
```



```
C
      SUBROUTINE MPRIME(F,DMDT)
C     (* 902 2 -   COMPUTE TIME DERIVATIVE OF SPACECRAFT MASS *)
C
C     CALLS
C        MMLT
C     CALLED BY
C        STPRIM
C
C     INPUT
C        AK1,F
C     OUTPUT
C        DMDT
C
      REAL AK1(14)
C        CONSTANT VECTOR DERIVED FROM ENGINE SPECIFIC IMPULSE DATA
      REAL DMDT
C        TIME DERIVATIVE OF MASS
      REAL F(14)
C        FORCES FROM THRUSTERS AFTER SELECTION
C
      COMMON/VEHIC/AK1,DUMMY(99)
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* COMPUTE DMDT = AK1*F *)
         CALL MMLT(DMDT,AK1,F,1,14,1)
C     (* CHANGE SIGN   MASS LOSS IS NEGATIVE *)
         DMDT=-DMDT
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C     SUBROUTINE LINACL(T,M,NTFORC,CURPOS,ACCEL)
C     (* 902.3 -  COMPUTE LINEAR ACCELERATIONS IN X, Y, AND Z DIRECTIONS *)
C     CALLS
C        SIN,COS
C     CALLED BY
C        STPRIM
C     INPUT
C        T M NTFORC CURPOS
C     OUTPUT
C        ACCEL
      REAL ACCEL(3)
C        SPACECRAFT ACCELERATIONS
      REAL AGGX,AGGY,AGGZ
C        ACCELERATIONS DUE TO GRAVITY GRADIANT
      REAL CURPOS(3)
C        CURRENT CHASE VEHICLE POSITION VECTOR
      REAL M
C        MASS OF SPACECRAFT IS M
      REAL NTFORC(3)
C        NET FORCE FROM THRUSTER OPERATION
      REAL SWT,CWT,A,B
C        THESE ARE INTERMEDIATE RESULTS
      REAL T
C        ELAPSED TIME
      REAL WO
C        RELATIVE ANGULAR ACCELERATION IN RADS/SEC
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      (* PRECOMPUTE VALUES USED SEVERAL PLACES *)
         WO=1.1629E-3
         A=2.67671E-6
         B=WO*T
         SWT=SIN(B)
         CWT=COS(B)
C     (* COMPUTE GRAVITY GRADIENT ACCELERATION *)
         AGGX=A*SWT*(CURPOS(1)*SWT+CURPOS(3)*CWT)
         AGGY=0.0
         AGGZ=A*CWT*(CURPOS(1)*SWT+CURPOS(3)*CWT)
C     (* COMPUTE NET ACCELERATION (GRAVITY PLUS THRUSTERS) *)
         ACCEL(1)=AGGX+NTFORC(1)/M
         ACCEL(2)=AGGY+NTFORC(2)/M
         ACCEL(3)=AGGZ+NTFORC(3)/M
      RETURN
C
      END
```



```
C
      SUBROUTINE TORQUE(F,N,TRNA)
C     (* 902.4 -  CALCULATE TORQUE (N) *)
C     CALLS
C        MMLT,MADD,MSCL
C     CALLED BY
C        STPRIM
C     INPUT:
C        AK3,F,TRNA
C     OUTPUT:
C        N
      REAL AK3(3,14)
C        MATRIX RELATING THRUSTER INPUTS TO TORQUES IN 'CV' REF. FRAME
      REAL F(14)
C        FORCES FROM THRUSTERS AFTER SELECTION
      REAL N(3)
C        TORQUE ON SPACECRAFT ABOUT ITS CENTER OF MASS IN TRUTH COORD. SYS.
      REAL N1(3)
C        TORQUE IN C.V. REFERENCE FRAME
      REAL TRNA(3,3)
C        TRANSPOSE OF DIRECTION COSINE MATRIX THAT GIVES C.V  ATTITUDE
C
      COMMON/VEHIC/DUMMY1(56),AK3,DUMMY2(15)
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* CALCULATE N=TRNA*AK3*F *)
         CALL MMLT(N1,AK3,F,3,14,1)
         CALL MMLT(N,TRNA,N1,3,3,1)
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE LPRIME(N,BODVEL,ANGMNT,DLDT,TRNA)
C     (* 902 5 -  COMPUTE TIME DERIVATIVE OF ANGULAR MOMENTUM *)
C
C     CALLS
C        MSUB,MMLT
C     CALLED BY
C        STPRIM
C
C     INPUT
C        N,ANGVEL,ANGMNT
C     OUTPUT
C        DLDT
C
      REAL ANGMNT(3)
C        ANGULAR MOMENTUM VECTOR IN 'TRUTH' COORDINATE SYSTEM
      REAL ANGVEL(3)
C        CHASE VEHICLE ANGULAR VELOCITY IN 'TRUTH' COORDINATE SYSTEM
      REAL BODVEL(3)
C        CHASE VEHICLE ANGULAR VELOCITY IN CV COORDINATE SYSTEM
      REAL B(3)
C        TEMPORARY STORAGE VECTOR
      REAL DLDT(3)
C        TIME DERIVATIVE OF ANGULAR MOMENTUM VECTOR
      REAL N(3)
C        TORQUE ON C V  ABOUT ITS CENTER OF MASS IN 'TRUTH' COORDINATE SYSTEM
      REAL TRNA(3,3)
C        TRANSPOSE OF DIRECTION COSINE MATRIX
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* CONVERT ANGULAR VELOCITY TO 'TRUTH' COORDINATE SYSTEM  *)
         CALL MMLT(ANGVEL,TRNA,BODVEL,3,3,1)
C     (* COMPUTE B = CROSS PRODUCT OF ANGVEL AND ANGMNT *)
         B(1)= ANGVEL(2)*ANGMNT(3)-ANGVEL(3)*ANGMNT(2)
         B(2)= ANGVEL(3)*ANGMNT(1)-ANGVEL(1)*ANGMNT(3)
         B(3)= ANGVEL(1)*ANGMNT(2)-ANGVEL(2)*ANGMNT(1)
C     (* COMPUTE DLDT=N-B *)
         CALL MSUB(DLDT,N,B,3,1)
      RETURN
C
      END
```



```
C
      SUBROUTINE MAKROT(BODVEL,OMEGA,A)
C     (* 902 6 -  FORM ROTATION MATRIX OMEGA FROM ANGULAR VELOCITY VECTOR *)
C
C     CALLS
C        <NONE>
C     CALLED BY
C        STPRIM
C
C     INPUT
C        ANGVEL
C     OUTPUT
C        OMEGA
C
      REAL A(3,3)
C        DIRECTION COSINE MATRIX GIVING TRUE C V  ATTITUDE
      REAL BODVEL(3)
C        ANGULAR VELOCITY IN CV COORDINATE SYSTEM
      INTEGER I
C        DO LOOP INDEX
      REAL OMEGA(4,4)
C        ROTATION MATRIX OMEGA, TO BE APPLIED TO QUATERNION
C        TO FORM TIME DERIVATIVE OF QUATERNION
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      DO 10 I=1,4
         OMEGA(I,I)=0.0
10       CONTINUE
      OMEGA(1,2)=BODVEL(3)
      OMEGA(1,3)=-BODVEL(2)
      OMEGA(1,4)=BODVEL(1)
      OMEGA(2,1)=-BODVEL(3)
      OMEGA(2,3)=BODVEL(1)
      OMEGA(2,4)=BODVEL(2)
      OMEGA(3,1)=BODVEL(2)
      OMEGA(3,2)=-BODVEL(1)
      OMEGA(3,4)=BODVEL(3)
      OMEGA(4,1)=-BODVEL(1)
      OMEGA(4,2)=-BUDVEL(2)
      OMEGA(4,3)=-BODVEL(3)
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
      SUBROUTINE QPRIME(Q,OMEGA,QPRM)
C     (* 902.7 - COMPUTE QPRM = TIME DERIVATIVE OF QUATERNION Q *)
C
C     CALLS
C        MMLT,MSCL
C     CALLED BY
C        STPRIM
C
C     INPUT
C        Q,OMEGA
C     OUTPUT
C        QPRM
C
      REAL OMEGA(4,4)
C        MATRIX FORMED FROM ANGULAR VELOCITY COMPONENTS
      REAL Q(4)
C        ATTITUDE QUATERNIION OF CHASE VEHICLE
      REAL QINT(4)
C        INTERMEDIATE VALUES. NO PROBLEM RELATED SIGNIFICANCE
      REAL QPRM(4)
C        TIME DERIVATIVE OF OF Q
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* COMPUTE TIME DERIVATIVE OF Q = 0.5*OMEGA*Q *)
         CALL MMLT(QINT,OMEGA,Q,4,4,1)
         CALL MSCL(QPRM,QINT,4,1,0.5)
      RETURN
C
      END
```



```
C
      FUNCTION SQR1(X)
C     (* 903 - RETURNS SQUARE ROOT BUT ALLOWS SLIGHTLY NEGATIVE ARGUMENT *)
C
C     CALLS
C        SQRT,AMAX1
C     CALLED BY
C        POSIT,SETGOL,QUATRN
C
C     INPUT
C        X
C     OUTPUT
C        SQR1 (FUNCTION VALUE ONLY)
C
      REAL X
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      SQR1=SQRT(AMAX1(X,0.))
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE ANGVEC(INERTA,ANGMNT,BODVEL,A)
C     (* 904 - COMPUTE TRUE ANGULAR VELOCITY VECTOR ANGVEL
C        IN CV BODY COORDINATE SYSTEM *)
C
C     CALLS
C        MINV,MMLT
C     CALLED BY
C        STPRIK IMU
C
C     INPUT
C        INERTA,ANGMNT,TERMNL,A
C     OUTPUT
C        BODVEL
C
      REAL A(3,3)
C        DIRECTION COSINE MATRIX (CHASE VEHICLE ATTITUDE WRT 'TRUTH' AXES)
      REAL ANGMNT(3)
C        ANGULAR MOMENTUM VECTOR
      REAL BODVEL(3)
C        CHASE VEHICLE ANGULAR VELOCITY IN CV COORDINATE SYSTEM
      REAL INERTA(3,3)
C        INERTIA OF CHASE VEHICLE (LOADED)
      REAL INVIN(3,3)
C        INVERSE OF MOMENT OF INERTIA TENSOR
      REAL TEMP1(3)
C        INTERMEDIATE RESULTS WITH NO PROBLEM-RELATED SIGNIFICANCE
      REAL WORK(4,6)
C        AN INTERMEDIATE WORKSPACE
      INTEGER IERR
C        ERROR FLAG (=0 IF NO ERROR)
      INTEGER TERMNL
C        FORTRAN LOGICAL UNIT NUMBER OF USER TERMINAL
C
      COMMON/SIMUL/TERMNL,IDUMMY
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* FORM INVERSE OF INERTIA TENSOR *)
         CALL MINV(INVIN,INERTA,3,WORK,4,6,IERR)
      IF(IERR NE 0) GO TO 10
C     (* CALCULATE BODVEL=INVIN*A*ANGMNT *)
         CALL MMLT(TEMP1,A,ANGMNT,3,3,1)
         CALL MMLT(BODVEL,INVIN,TEMP1,3,3,1)
      RETURN
C
   10 CONTINUE
C     (* REPORT ERROR CONDITION AND HALT *)
         WRITE(TERMNL,901)
         STOP
C
  901 FORMAT(' MATRIX INVERSION FAILURE IN SUBROUTINE ANGVEC')
C
      END
```



```
C
      SUBROUTINE TRGATT(TRNAT,T)
C     (* 905 - COMPUTE TARGET ATTITUDE AS A FUNCTION OF TIME *)
C
C     CALLS
C        COS,SIN
C     CALLED BY
C        DOCK FLASH
C
C     INPUT
C        T
C     OUTPUT
C        TRNAT
C
      REAL CHGATT(3,3)
C        TRANSPOSE OF DIRECTION COSINE MATRIX REPRESENTING TRUE TARGET
C        ATTITUDE CHANGE
      INTEGER ITUMBL
C        TARGET TUMBLE AXIS--VALUE OF 1, 2, OR 3 REPRESENTS YAW,
C        PITCH, OR ROLL TUMBLE
      REAL PHI
C        ROTATION ANGLE
      REAL T
C        ELAPSED TIME
      REAL TRNAT(3,3)
C        TRANSPOSE OF TARGET'S TRUE ATTITUDE DIRECTION COSINE MATRIX
      REAL TRNATO(3,3)
C        INITIAL VALUE OF TRNAT
      REAL TUMRAT
C        TUMBLE RATE (IN RAD/SEC)
C
      COMMON/ATINFO/TRNATO,ITUMBL,TUMRAT
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      PHI=TUMRAT*T
C     (* SELECT APPROPRIATE SET OF FORMULAS FOR TARGET ATTITUDE *)
         GO TO (10,20,30), ITUMBL
   10    CONTINUE
C        (* COMPUTE ATTITUDE CHANGE RESULTING FROM YAW TUMBLE *)
            CHGATT(1,1)=COS(PHI)
            CHGATT(2,1)=SIN(PHI)
            CHGATT(3,1)=0.0
            CHGATT(1,2)=-SIN(PHI)
            CHGATT(2,2)=COS(PHI)
            CHGATT(3,2)=0.0
            CHGATT(1,3)=0.0
            CHGATT(2,3)=0.0
            CHGATT(3,3)=1.0
            GO TO 40
   20    CONTINUE
C        (* COMPUTE ATTITUDE CHANGE RESULTING FROM PITCH TUMBLE *)
            CHGATT(1,1)=COS(PHI)
            CHGATT(2,1)=0.0
            CHGATT(3,1)=-SIN(PHI)
            CHGATT(1,2)=0.0
            CHGATT(2,2)=1.0
            CHGATT(3,2)=0.0
            CHGATT(1,3)=SIN(PHI)
            CHGATT(2,3)=0.0
            CHGATT(3,3)=COS(PHI)
            GO TO 40
   30    CONTINUE
C        (* COMPUTE ATTITUDE CHANGE RESULTING FROM A ROLL TUMBLE *)
            CHGATT(1,1)=1.0
            CHGATT(2,1)=0.0
            CHGATT(3,1)=0.0
            CHGATT(1,2)=0.0
            CHGATT(2,2)=COS(PHI)
            CHGATT(3,2)=SIN(PHI)
            CHGATT(1,3)=0.0
            CHGATT(2,3)=-SIN(PHI)
            CHGATT(3,3)=COS(PHI)
   40 CONTINUE
C     (* COMPUTE TRUE TARGET ATTITUDE *)
         CALL MMLT(TRNAT,TRNATO,CHGATT,3,3,3)
      RETURN
```

ORIGINAL PAGE IS
OF POOR QUALITY

END

Appendix B
Computer Program to Simulate System That Uses "Rainbow" Beacon and Rangefinder

Appendix B Computer Program to Simulate System That Uses "Rainbow" Beacon and Rangefinder

APPENDIX B--COMPUTER PROGRAM TO SIMULATE SYSTEM THAT USES "RAINBOW" BEACON AND RANGEFINDER

The program listing in this appendix is provided to document the simulation methods used in analyzing the "rainbow" beacon video guidance system. The discussion presented in Appendix A is equally applicable to this program. The text of the terminal session, which is presented in Appendix A, is not repeated here because the user commands and dialog are essentially identical to those in Appendix A.

APPENDIX B -- RAINBOW VERSION OF SIMULATION PROGRAM

PAGE 1

```
C          TRUE STATE
C
C             STATE( 1)  X POSITION
C             STATE( 2)  Y POSITION
C             STATE( 3)  Z POSITION
C             STATE( 4)  X VELOCITY
C             STATE( 5)  Y VELOCITY
C             STATE( 6)  Z VELOCITY
C             STATE( 7)  ANGULAR MOMENTUM ABOUT X
C             STATE( 8)  ANGULAR MOMENTUM ABOUT Y
C             STATE( 9)  ANGULAR MOMENTUM ABOUT Z
C             STATE(10) Q1 \
C             STATE(11) Q2  \
C             STATE(12) Q3   > ATTITUDE W.R.T. 'TRUTH' AXES
C             STATE(13) Q4  /
C             STATE(14)  MASS, INCL FUEL
C
C               (X,Y,Z) = 'TRUTH' AXES
C
C
C          STATE ESTIMATE
C
C             ESTATE(1)  X' POSITION
C             ESTATE(2)  Y' POSITION
C             ESTATE(3)  Z' POSITION
C             ESTATE(4)  X' VELOCITY
C             ESTATE(5)  Y' VELOCITY
C             ESTATE(6)  Z' VELOCITY
C
C               (X',Y',Z') = 'PRIMARY REF. FRAME' AXES
```

ROUTINE	PAGE	ROUTINE NBR	
<MAIN>	3	000	
ACCEL	41	481	1
ANGVEC	66	904	
ATITUD	24	440	
CENTRD	15	410	
CNTLAW	40	481	
COMPG	30	451	
COMPK1	18	421	
COMPK2	19	422	
COMPK3	20	423	
COMPK4	21	424	
DIRMAT	56	901	
DOCK	11	400	
DPPD	70	907	
ESTCOV	31	452	
ESTRPY	53	480	
FINDCV	25	441	
FIRTHR	50	483	
FORCE	58	902	1
GETPCH	54	400	
GETYAW	55	400	
IMU	51	490	
INCORP	29	450	
INIPAR	5	200	
INISIM	10	300	
KALGAN	32	453	
LINACL	60	902	3
LPRIME	62	902	5
MAKROT	63	902	6
MPRIME	59	902	2
OPEN	4	100	
POINT	22	425	
POSIT	23	430	
PROPES	35	460	
PROPTR	17	420	
QPRIME	64	902	7
QUATRN	26	442	
RPY	52	4A0	
SELECT	43	482	
SETGOL	37	470	
SOP1	65	903	
STPRIM	57	902	
TABLE1	44	482	1
TABLE2	46	482	2
TABLE3	48	482	3
THRUST	39	480	
TOQUAT	28	442	1
TORQUE	61	902	4
TRGATT	67	905	
UPDCOV	34	455	
UPDSTA	33	454	
XPRD	69	906	


```
      PROGRAM MSFC DSIM2 - DOCKING SIMULATION PROGRAM, RAINBOW

      CALLS
         CLOS$A, DOCK, EXIT, INIPAR INISIM, OPEN, YSNO$A

      REAL AT(3,3)
         MEASURED DIRECTION COSINE MATRIX OF TARGET SPACECRAFT
      REAL CAMPOS(3,2)
         MEASURED CENTER-CAMERA POSITION IN COORDINATE SYSTEM CENTERED AT
         AT DOCKING AID CENTER AND PARALLEL TO PRIMARY REFERENCE FRAME
         (FIRST COLUMN IS CURRENT MEASUREMENT, SECOND IS PREVIOUS ONE)
      LOGICAL CLOS$A
         LIBRARY FUNCTION TO CLOSE FILES (FUNCTION VALUE INDICATES SUCCESS)
      REAL ESTATE(6)
         ESTIMATE OF STATE VECTOR
      REAL F(14)
         FORCES FROM THRUSTERS AFTER SELECTION
      INTEGER OUT
         FORTRAN UNIT NUMBER OF OUTPUT FILE
      REAL P(6,6)
         COVARIANCE OF STATE ESTIMATE
      REAL RELPOS(3,2)
         RELATIVE CENTER-CAMERA COORDINATES IN CURRENT DOCKING AID FRAME
         CENTERED AT BEACON LIGHT  FIRST COLUMN IS CURRENT MEASUREMENT,
         SECOND IS PREVIOUS MEASUREMENT
      REAL STATE(14)
         CHASE VEHICLE STATE VECTOR
      REAL T
         ELAPSED TIME
      INTEGER TERMNL
         FORTRAN UNIT NUMBER FOR TERMINAL
      LOGICAL DOCKED
         CODE SET TO ' TRUE ' IF THE CHASE VEHICLE IS WITHIN DOCKING RADIUS
      LOGICAL FILERR
         ERROR CODE RETURNED FROM 'OPEN' SUBROUTINE
      LOGICAL YAWCHK
         TRUE IF YAW BEACON LIGHT IS ON, OTHERWISE, PITCH BEACON LIGHT IS ON
      LOGICAL YSNO$A
         FUNCTION TO GET YES/NO ANSWER FROM USER AT TERMINAL

      COMMON/SIMUL/TERMNL,OUT
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      OUT=6
      TERMNL=1
      (* LOOP UNTIL USER WANTS NO MORE SIMULATIONS *)
   10 CONTINUE
         (* 100 - OPEN DATA FILE *)
            CALL OPEN(FILERR,TERMNL,OUT)
         (* 200 - INITIALIZE PROBLEM PARAMETERS *)
            CALL INIPAR(T,STATE,P,ESTATE,F,YAWCHK,RELPOS CAMPOS,AT)
         (* 300 - INITIALIZE SIMULATOR *)
            CALL INISIM(STATE)
         DOCKED= FALSE
         (* LOOP WHILE NOT DOCKED *)
   20    IF (DOCKED) GO TO 30
            (* 400 - SIMULATE OBSERVATION & INTERVAL BETWEEN OBSERVATIONS *
               CALL DOCK(P,ESTATE,STATE,T,DOCKED,F,YAWCHK,AT,RELPOS,
     A            CAMPOS)
            GO TO 20
   30    CONTINUE
         (* CLOSE DATA FILE *)
            IF( NOT CLOS$A(OUT-4)) GO TO 40
   40    CONTINUE
         (* ASK OPERATOR WHETHER HE WANTS ANOTHER SIMULATION *,
            IF(YSNO$A('DO YOU WANT ANOTHER SIMULATION' 30,-1)) GO TO 10
      CALL EXIT

      END
```

```
C
      SUBROUTINE OPEN(FILERR, TERMNL, OUT)
C     (* 100 - OPEN FILES FOR OUTPUT  WARNING  HIGHLY INSTALLATION DEPENDENT *)
C
C     CALLS
C         DELESA, EXSTSA, TSRCS$
C     CALLED BY
C         <MAIN>
C
C     INPUT
C         IN, AT
C     OUTPUT
C         FILERR
C
      INTEGER CHRPOS(2)
C         CHARACTER POSITION/COUNT CODES FOR SYSTEM ROUTINE TSRCS$
      INTEGER CODE
C         ERROR CODE RETURNED BY TSRCS$ (=0 IF NO ERROR)
      INTEGER I
C         DO LOOP INDEX
      INTEGER OUT, TERMNL
C         FORTRAN LOGICAL UNIT NUMBER FOR OUTPUT
      INTEGER PATH(16)
C         ARRAY OF ASCII DEFINING PATH NAME
      INTEGER TERMNL
C         FORTRAN LOGICAL UNIT NUMBER OF USER TERMINAL
      INTEGER TYPE
C         TYPE CODE RETURNED BY TSRCS$ (UNUSED HERE)
      LOGICAL DELESA
C         SYSTEM ROUTINE TO DELETE A FILE (RETURNS TRUE IF SUCCESSFUL)
      LOGICAL EXSTSA
C         SYSTEM ROUTINE TO CHECK EXISTENCE OF FILE (TRUE IF IT EXISTS)
      LOGICAL FILERR
C         ERROR CODE RETURNED TO CALLING ROUTINE
      LOGICAL YSNOSA
C         FUNCTION TO GET YES/NO ANSWER FROM USER AT TERMINAL
C
$INSERT SYSCOM>KEYS F
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      CHRPOS(2)=32
      TYPE=0
   10 CONTINUE
            WRITE(TERMNL, 902)
            READ(TERMNL, 901)(PATH(I), I=1, 16)
         IF( NOT EXSTSA(PATH, 32)) GO TO 20
            IF( NOT YSNOSA('***FILE ALREADY EXISTS   OK TO OVERWRITE',
     A            40, -1)) GO TO 10
      IF( NOT DELESA(PATH, 32)) GO TO 20
   20 CONTINUE
      CHRPOS(1)=0
      CALL TSRCS$(K$WRIT+K$NSAM, PATH, OUT-4, CHRPOS, TYPE, CODE)
      IF(CODE NE 0) GO TO 30
      FILERR= FALSE
      RETURN
C
   30 FILERR= TRUE
      RETURN
C
  901 FORMAT(16A2)
  902 FORMAT('ENTER PATHNAME FOR OUTPUT     ')
C
      END
```

ORIGINAL PAGE IS OF POOR QUALITY


```
C
      SUBROUTINE INIPAR(T, STATE, P, ESTATE, F, YAWCHK, RELPOS, CAMPOS, AT)
C     (* 200 - INITIALIZE PROBLEM PARAMETERS *)
C
C     CALLS
C         SQRT, RANFN
C     CALLED BY
C         <MAIN>
C
C     INPUT
C         <NONE>
C     OUTPUT
C         AK1, AK2, AK3, DOKRAD, ESTATE, FOCLEN, FULDIS, F, F1, INERCV, MEMPTY, OUT, P, QO,
C         TPHIH, TPHIV, STATE, TERMNL, T, HCL, HCR, HDC, HDT, CAMPOS, RELPOS, AT
C
      REAL AK1(14), AK2(3, 14), AK3(3, 14)
C         MATRICES SPECIFYING CHASE VEHICLE:
C            AK1 GIVES FUEL USE PER NEWTON-SECOND OF THRUST FOR EACH THRUSTER
C            AK2 GIVES FORCE DIRECTION FOR EACH THRUSTER
C            AK3 GIVES TORQUE/NEWTON & TORQUE DIRECTION FOR EACH THRUSTER
      REAL AT(3, 3)
C         MEASURED DIRECTION COSINE MATRIX OF TARGET SPACECRAFT
      REAL CAMPOS(3, 2)
C         MEASURED CENTER-CAMERA POSITION IN COORDINATE SYSTEM CENTERED AT
C         DOCKING AID CENTER AND PARALLEL TO PRIMARY REFERENCE FRAME
C         (FIRST COLUMN IS CURRENT MEASUREMENT, SECOND IS PREVIOUS ONE)
      REAL DOKRAD
C         RADIUS FROM TARGET SPACECRAFT WITHIN WHICH THE SPACECRAFT ARE
C         CONSIDERED DOCKED
      REAL ESTATE(6)
C         ESTIMATE OF STATE VECTOR
      REAL FOCLEN
C         LENS FOCAL LENGTH IN METERS
      REAL FULDIS(3, 3)
C         CHASE VEHICLE FUEL-DISTRIBUTION INERTIA TENSOR
      REAL F(14)
C         FORCES FROM THRUSTERS AFTER SELECTION
      REAL F1(14)
C         LIST OF MAXIMUM THRUST MAGNITUDES FROM EACH THRUSTER OF CHASE VEHICLE
      REAL HCL(3), HCC(3), HCR(3), HT(3)
C         LEFT, CENTER, AND RIGHT CAMERA POSITIONS IN CHASE VEHICLE BODY FRAME
C         AND POSITION OF CENTER OF DOCKING AID IN
C         TARGET SPACECRAFT BODY FRAME, RESPECTIVELY.
      REAL HDC(3), HDT(3)
C         DOCKING FIXTURE POSITIONS IN THEIR RESPECTIVE SPACECRAFT COORDI-
C         NATE SYSTEMS
      INTEGER I, J
C         DO LOOP INDICES
      REAL INERCV(3, 3)
C         INERTIA OF EMPTY CHASE VEHICLE
      INTEGER ITUMBL
C         TUMBLE AXIS, VALUE OF 1, 2, OR 3 INDICATES YAW, PITCH, OR ROLL
C         TUMBLE, RESPECTIVELY
      REAL MEMPTY
C         MASS OF CHASE VEHICLE WITHOUT FUEL
      INTEGER OUT
C         FORTRAN UNIT NUMBER FOR OUTPUT FILE
      REAL P(6, 6)
C         COVARIANCE OF STATE ESTIMATE
      REAL PHI, PSI, THETA
C         ANGLES SPECIFYING INITIAL TARGET ATTITUDE FOR A YAW-PITCH-ROLL
C         SEQUENCE
      REAL QO(4)
C         INITIAL ATTITUDE QUATERNION OF CHASE VEHICLE
      REAL RELPOS(3, 2)
C         RELATIVE CENTER-CAMERA COORDINATES IN CURRENT DOCKING AID FRAME
C         CENTERED AT BEACON LIGHT: FIRST COLUMN IS CURRENT MEASUREMENT,
C         SECOND COLUMN IS PREVIOUS MEASUREMENT
      REAL TPHIH, TPHIV
C         TANGENTS OF HALF-FIELD-OF-VIEW ANGLES IN HORIZONTAL AND VERTICAL
C         DIRECTIONS FOR CAMERA SCANNING, RESPECTIVELY
      REAL STATE(14)
C         CHASE VEHICLE STATE VECTOR
      INTEGER TERMNL
C         FORTRAN LOGICAL UNIT NUMBER OF USER TERMINAL
```

```
      REAL T
C         ELAPSED TIME
      REAL TRNATO(3,3)
C         TRUE INITIAL TARGET DIRECTION COSINE MATRIX
      LOGICAL YAWCHK
C         TRUE IF YAW BEACON LIGHT IS ON. OTHERWISE, PITCH BEACON LIGHT IS O.
C
      COMMON/MASPRP/FULDIS,INERCV,MEMPTY
      COMMON/OPTSYS/TPHIH,TPHIV,FOCLEN
      COMMON/VEHIC/AK1,AK2,AK3,DOKRAD,F1
      COMMON/SIMUL/TERMNL,OUT
      COMMON/HNGOFF/HCL,HCR,HT,HDC,HDT,HCC
      COMMON/REF/GO
      COMMON/ATINFO/TRNATO,ITUMBL,TUMRAT
```



```
C
C     (* INPUT INITIAL ATTITUDE ANGLES *)
      WRITE(TERMNL,901)
      READ(TERMNL,*) PHI,PSI,THETA
C     (* CONVERT ANGLES TO RADIANS *)
         PHI=PHI* 01745329
         PSI=PSI* 01745329
         THETA=THETA* 01745329
C     (* COMPUTE TRANSPOSE OF TRUE INITIAL TARGET ATTITUDE *)
         TRNATO(1,1)=COS(PSI)*COS(PHI)
         TRNATO(2,1)=COS(PSI)*SIN(PHI)
         TRNATO(3,1)=-SIN(PSI)
         TRNATO(1,2)=SIN(THETA)*SIN(PSI)*COS(PHI)-COS(THETA)*SIN(PHI)
         TRNATO(2,2)=SIN(PHI)*SIN(THETA)*SIN(PSI)+COS(THETA)*COS(PHI)
         TRNATO(3,2)=SIN(THETA)*COS(PSI)
         TRNATO(1,3)=COS(THETA)*SIN(PSI)*COS(PHI)+SIN(THETA)*SIN(PHI)
         TRNATO(2,3)=SIN(PHI)*COS(THETA)*SIN(PSI)-SIN(THETA)*COS(PHI)
         TRNATO(3,3)=COS(THETA)*COS(PSI)
C     (* INPUT TUMBLE AXIS *)
         WRITE(TERMNL,902)
    3    READ(TERMNL,*) ITUMBL
            IF(ITUMBL EQ 1 OR ITUMBL EQ 2 OR ITUMBL EQ 3) GO TO 5
            WRITE(TERMNL,904)
            GO TO 3
    5 CONTINUE
C     (* INPUT TUMBLE RATE *)
         WRITE(TERMNL,903)
         READ(TERMNL,*) TUMRAT
C     (* CONVERT TUMBLE RATE FROM DEGREES/HOUR TO RADIANS/SECOND
         TUMRAT=TUMRAT*4 84813E-6
      CALL MIDN(AT,3)
      DOKRAD= 4
      FOCLEN= 05
      F1(1)=360
      F1(2)=80
      F1(3)=80
      F1(4)=80
      F1(5)=80
      F1(6)=20
      F1(7)=20
      F1(8)=20
      F1(9)=20
      F1(10)=80
      F1(11)=80
      F1(12)=80
      F1(13)=80
      F1(14)=360
      DO 10 I=1,3
         DO 10 J=1,3
            INERCV(I,J)=0
   10 CONTINUE
      INERCV(1,1)=1055
      INERCV(2,2)=1905
      INERCV(3,3)=1905
      TPHIH= 5
      TPHIV= 5
      T=0 0
      DO 20 I=1,14
         F(I)=0
   20 CONTINUE
      DO 30 I=1,3
         DO 30 J=1,3
            FULDIS(I,J)=0
   30 CONTINUE
      FULDIS(1,1)= 781
      FULDIS(2,2)= 706
      FULDIS(3,3)= 706
      DO 40 I=1,14
         AK1(I)=4 636E-4
   40 CONTINUE
      DO 50 I=1,3
         DO 50 J=1,14
            AK2(I,J)=0
            AK3(I,J)=0
   50 CONTINUE
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
      AK2(1,1)=-1
      AK2(2,2)=1
      AK2(2,3)=1
      AK2(2,4)=-1
      AK2(2,5)=-1
      AK2(2,6)=-1
      AK2(2,7)=1
      AK2(2,8)=-1
      AK2(2,9)=1
      AK2(3,10)=-1
      AK2(3,11)=-1
      AK2(3,12)=1
      AK2(3,13)=1
      AK2(1,14)=1
      AK3(3,2)=-.375
      AK3(3,3)=.375
      AK3(3,4)=-.375
      AK3(3,5)=.375
      AK3(1,6)=1.25
      AK3(1,7)=-1.25
      AK3(1,8)=-1.25
      AK3(1,9)=1.25
      AK3(2,10)=-.375
      AK3(2,11)=.375
      AK3(2,12)=.375
      AK3(2,13)=-.375
      MEMPTY=1800
      DO 60 I=1,6
         DO 60 J=1,6
            P(I,J)=0
   60 CONTINUE
      P(1,1)=400
      P(2,2)=563
      P(3,3)=563
      P(4,4)=.03
      P(5,5)=.03
      P(6,6)=.03
      ESTATE(1)=-300
      ESTATE(2)=0
      ESTATE(3)=0
      ESTATE(4)=.1
      ESTATE(5)=.1
      ESTATE(6)=.1
      DO 65 I=1,3
      RELPOS(I,1)=ESTATE(I)
      RELPOS(I,2)=ESTATE(I)
      CAMPOS(I,1)=ESTATE(I)
   65 CONTINUE
      DO 70 I=1,6
         STATE(I)=RANFN(2,ESTATE(I),SQRT(P(I,I)))
   70 CONTINUE
      STATE(7)=0.0
      STATE(8)=0.0
      STATE(9)=0.0
      STATE(10)=0
      STATE(11)=0
      STATE(12)=0
      STATE(13)=1
      STATE(14)=3700
      QO(1)=STATE(10)
      QO(2)=STATE(11)
      QO(3)=STATE(12)
      QO(4)=STATE(13)
      HCL(1)=.98
      HCL(2)=2.0
      HCL(3)=0.0
      HCC(1)=.98
      HCC(2)=0.0
      HCC(3)=0.0
      HCR(1)=.98
      HCR(2)=-2.0
      HCR(3)=0.0
      HT(1)=-1.25
      HT(2)=2.0
      HT(3)=0.0
```



```
      HDC(1)=3.49
      HDC(2)=0
      HDC(3)=0
      HDT(1)=-2
      HDT(2)=0
      HDT(3)=0
      YAWCHK=.FALSE.
      RETURN
  901 FORMAT(/' ENTER THREE ROTATION ANGLES TO SPECIFY INITIAL TARGET'/
     A    ' ATTITUDE FOR A YAW-PITCH-ROLL SEQUENCE---ANGLES MUST BE'/
     B    ' IN DEGREES')
  902 FORMAT(/' ENTER TUMBLE AXIS-- ENTER 1, 2, OR 3 FOR YAW, PITCH, '/
     A    ' OR ROLL, RESPECTIVELY')
  903 FORMAT(/' ENTER TUMBLE RATE IN DEGREES PER HOUR ')
  904 FORMAT(/' A VALUE OF 1, 2, OR 3 MUST BE ENTERED FOR TUMBLE AXIS')
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE INISIM(STATE)
C     (* 300 - INITIALIZE SIMULATOR *
C
C     CALLS
C         (NONE)
C     CALLED BY
C         (MAIN)
C
C     INPUT
C         (NONE)
C     OUTPUT
C         (NONE)
C
C     (* DUMMY ROUTINE - ACCOMODATES PHYSICAL SIMULATION *)
      RETURN
C
      END
```



```
C
      SUBROUTINE DOCK(P,ESTATE,STATE,T,DOCKED,F,YAWCHK,AT,RELPOS,
     A   CAMPOS)
C     (* 400 - RUN SIMULATION FOR ONE MEASUREMENT INTERVAL *)
C
C     CALLS
C         SETGOL,INCORP,PROPST,SQRT,CENTRD,PROPTR,POSIT,ATITUD,PROPES,THRUST,
C         IMU,RPY,MMLT,MSUB,MADD,GETYAW,GETPCH,ESTRPY,TRGATT,SETGOL,AMAXI
C     CALLED BY
C         (MAIN)
C
C     INPUT
C         P,ESTAT,YAWCHKE,DOKRAD,STATE,T,F,INERCV,CAMPOS,RELPOS,AT
C     OUTPUT
C         STATE,T,F,DOCKED,ESTATE,P
C
      REAL ACV(3,3),ACVT(3,3)
C         DIRECTION COSINE MATRIX DESCRIBING CURRENT CHASE VEHICLE ATTITUDE
C         (RELATIVE TO PRIMARY REFERENCE FRAME) AND TRANSPOSE OF ACV
      REAL AT(3,3)
C         MEASURED DIRECTION COSINE MATRIX OF TARGET SPACECRAFT
      REAL ATRATE(3)
C         MEASURED ATTITUDE RATES ABOUT CHASE VEHICLE AXES
      REAL CAMPOS(3,2)
C         MEASURED CENTER-CAMERA POSITION IN COORDINATE SYSTEM CENTERED AT
C         DOCKING AID CENTER AND PARALLEL TO PRIMARY REFERENCE FRAME
C         (FIRST COLUMN IS CURRENT MEASUREMENT , SECOND IS PREVIOUS ONE)
      REAL CVPOS(3)
C         MEASURED CHASE VEHICLE POSITION IN PRIMARY REFERENCE FRAME
C         FIRST COLUMN IS CURRENT MEASUREMENT, SECOND IS PREVIOUS MEASUREMENT
      REAL DEDLIN
C         TIME BY WHICH STRATEGY LOGIC WANTS CHASE VEHICLE IN GOAL 'BOX'
      REAL DOKRAD
C         RADIUS FROM TARGET AT WHICH CHASE VEHICLE IS CONSIDERED DOCKED
      REAL DT1,DT2
C         COMPUTATION TIME ALLOWANCE BEFORE THRUSTER COMMAND CHANGE AND
C         DELAY TO NEXT OBSERVATION, RESPECTIVELY
      REAL ESTATE(6)
C         ESTIMATE OF STATE VECTOR
      REAL F(14)
C         FORCES FROM CHASE VEHICLE THRUSTERS AFTER SELECTION
      REAL GXL,GXH,GYL,GYH,GZL,GZH
C         X,Y,Z UPPER,LOWER LIMITS OF GOAL 'BOX'
      REAL HDC(3),HDT(3)
C         DOCKING FIXTURES' POSITIONS IN RESPECTIVE SPACECRAFT COORDINATE
C         SYSTEMS
      INTEGER OUT
C         FORTRAN UNIT NUMBER FOR OUTPUT FILE
      REAL P(6,6)
C         COVARIANCE OF STATE ESTIMATE
      REAL PITCH
C         CORRUPTED VERSION OF THENGULAR ERROR FROM -X/Y PLANE MEASURED
C         IN THE PROJECTON OF THE POSITION ONTO THE -X/Z PLANE
      REAL R1(3),R2(3),R3(3),R4(3)
C         INTERMEDIATE RESULTS WITH NO SIMPLE PHYSICAL INTERPRETATION
      REAL RELPOS(3,2)
C         RELATIVE CENTER-CAMERA COORDINATES IN CURRENT DOCKING AID FRAME
C         CENTERED AT BEACON LIGHT  FIRST COLUMN IS CURRENT MEASUREMENT,
C         SECOND IS PREVIOUS MEASUREMENT
      REAL RHO
C         MEASURED RANGE, CAMERA TO TARGET DOCKING AID CENTER
      REAL RPYERR(3)
C         MEASURED ROLL, PITCH, AND YAW ERRORS IN RADIANS
      REAL STATE(14)
C         CHASE VEHICLE STATE VECTOR
      REAL T
C         ELAPSED TIME
      REAL TAOUT(3,3)
C         TRANSPOSE OF TARGET SPACECRAFT DIRECTION COSINE MATRIX
C         (WITH RESPECT TO TRUTH AXES) FOR OUTPUT ONLY
      REAL U(3),V(3)
C         U IS HORIZONTAL DEFLECTION, V IS VERTICAL DEFLECTION, SUBSCRIPT
C         INDICATES CAMERA NUMBER (LEFT, CENTER, OR RIGHT)
      REAL YAW
C         CORRUPTED VERSION OF THE ANGULAR ERROR FROM THE -X/Z PLANE MEASURED
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C         IN THE PROJECTION OF THE POSITION ONTO THE -X/Y PLANE
      LOGICAL DOCKED
C         TRUE IF CHASE VEHICLE IS WITHIN DOCKING RADIUS
      LOGICAL VALID,VALID1
C         TRUE IF MEASUREMENT VALID (LIGHT IN FIELD OF VIEW)
C         (VALID IS A MORE THOROUGH TEST THAN VALID1)
      LOGICAL YAWCHK
C         TRUE IF YAW BEACON LIGHT IS ON. OTHERWISE, PITCH BEACON LIGHT IS ON
C
      COMMON/SIMUL/IDUMMY,OUT
      COMMON/HNGOFF/DUMMY3(9),HDC,HDT,HCC
      COMMON/VEHIC/DUMMY1(98),DOKRAD,DUMMY2(14)
```

ORIGINAL PAGE IS
OF POOR QUALITY


```
C
      VALID= TRUE
C     (* TAKE MEASUREMENT *)
         IF(YAWCHK) GO TO 10
C           (* 4C0 - MEASURE ERROR ANGLE ABOUT TARGET Y AXIS *)
               CALL GETPCH(STATE,PITCH,T,VALID)
            DT1= 4
            DT2= 3
            GO TO 20
10       CONTINUE
C           (* 4D0 - MEASURE ERROR ANGLE ABOUT TARGET Z AXIS *)
               CALL GETYAW(STATE,YAW,T,VALID)
            DT1= 5333333
            DT2=1 2
20       CONTINUE
C        (* 410 - MEASURE CENTROID WITH EACH CAMERA  *)
            CALL CENTRD(STATE,VALID,VALID1,U,V,T)
C     (* 490 - TAKE IMU MEASUREMENT *)
         CALL IMU(STATE,ACV,ACVT,ATRATE)
C     (* 420 - PROPAGATE TRUE STATE TO TIME OF THRUSTER COMMAND CHANGE *)
         CALL PROPTR(DT1,F,STATE,T)
      IF( NOT VALID1) GO TO 30
C        (* 4A0 - CALCULATE ROLL, PITCH AND YAW ERRORS *)
            CALL RPY(U,V,RPYERR,ACV,AT)
         GO TO 40
30    CONTINUE
C        (* 4B0 - ESTIMATE ATTITUDE ERROR FROM STATE *)
            CALL ESTRPY(ESTATE,ACVT,RPYERR)
40    CONTINUE
      IF( NOT VALID) GO TO 50
C        (* IF YAW HAS NOT BEEN CALCULATED, DON'T CALCULATE ATTITUDE *)
            IF( NOT YAWCHK) GO TO 50
C              (* 430 - CALCULATE POSITONS AT TIME OF MEASUREMENT *)
                  CALL POSIT(U,V,RELPOS,RHO,PITCH,YAW,VALID)
C              (* 440 - CALC  TARGET ATTITUDE AND CHASE VEHICLE POSITION AT
C                 MEASUREMENT TIME *)
                  CALL ATITUD(ACV,ACVT,RELPOS,RHO,U,V,AT,CVPOS,CAMPOS)
C              (* 450 - USE MEASUREMENT TO UPDATE STATE ESTIMATE *)
                  CALL INCORP(CVPOS,P,ESTATE)
50    CONTINUE
C     (* WRITE PARAMETERS TO DISK FILE *)
         CALL TRGATT(TAOUT,T)
         WRITE(OUT) T,STATE,TAOUT
C     (* 460 - PROPAGATE STATE ESTIMATE TO TIME OF THRUSTER COMMAND CHANGE *)
         CALL PROPES(P,ESTATE,F,DT1,ACVT)
C     (* 470 - SET GOAL FOR NEXT MEASUREMENT INTERVAL *)
         CALL SETGOL(ESTATE,P,T,ACV,AT,GXL,GXH,GYL,GYH,GZL,GZH,DEDLIN,
     A      VALID)
C     (* 480 - CONTROL THRUSTERS *)
         CALL THRUST(F,ESTATE,GXL,GXH,GYL,GYH,GZL,GZH,AMAX1(
     A      DEDLIN-T,0 ),ACV,ATRATE,RPYERR)
C     (* 420 - PROPAGATE TRUE STATE TO TIME  NEXT MEASUREMENT *)
         CALL PROPTR(DT2,F,STATE,T)
C     (* 460 - PROP  STATE ESTIMATE TO TIME NEXT MEASUREMENT *)
         CALL PROPES(P,ESTATE,F,0 6,ACVT)
C     (* SEE IF CHASE VEHICLE HAS DOCKED YET *)
         CALL MMLT(R1,ACVT,HDC,3,3,1)
         CALL MADD(R2,R1,STATE,3,1)
         CALL MMLT(R3,AT,R2,3,3,1)
         CALL MSUB(R4,HDT,R3,3,1)
         DOCKED=(SQRT(R4(1)**2+R4(2)**2+R4(3)**2) LE DOKRAD)  OR
     A      (T GT 280 )  OR
     B      (STATE(1) LT -380 )  OR  (STATE(1) GT 15 )  OR
     C      (AMAX1(ABS(STATE(2)),ABS(STATE(3))) GT 200 )
C     (* WRITE PARAMETERS TO DISK FILE *)
         CALL TRGATT(TAOUT,T)
         WRITE(OUT) T STATE,TAOUT
C        WRITE(1,901) T STATE(1),STATE(2),STATE(3)
C        WRITE(1,902) STATE(4),STATE(5),STATE(6)
C        WRITE(1,903) STATE(10),STATE(11),STATE(12),STATE(13)
C        WRITE(1,904) STATE(7),STATE(8),STATE(9)
C        WRITE(1,905) STATE(14)
         YAWCHK= NOT YAWCHK
      RETURN
C
```

```
 901 FORMAT(/' -----TIME=',F10 2/' POSITION IS ',3G14 7)
 902 FORMAT(' VELOCITY IS ',3G14 7)
 903 FORMAT(' QUATERNION IS ',4G14 7)
 904 FORMAT(' ANGULAR MOMENTUM IS ',3G14 7)
 905 FORMAT(' MASS IS ',G14 7)
C
      END
```



```
C
      SUBROUTINE CENTRD(STATE, VALID, VALID1, U, V, T)
C     (* 410 - MEASURE CENTRO.D WITH EACH CAMERA *)
C
C     CALLS
C        TRGATT, MMLT, DIRMAT, MSUB, RANFN, MTRN, MSCL
C     CALLED BY
C        DOCK
C
C     INPUT
C        FOCLEN, TPHIH, TPHIV, STATE, VALID
C     OUTPUT
C        U, V, VALID
C
      REAL AC(3,3), TRNAC(3,3)
C        CHASE VEHICLE DIRECTION COSINE MATRIX AND ITS TRANSPOSE
      REAL AT(3,3), TRNAT(3,3)
C        TARGET DIRECTION COSINE MATRIX & ITS TRANSPOSE
      INTEGER CAMERA
C        INDICATES CAMERA NUMBER (LEFT, CENTER, OR RIGHT)
      REAL FOCLEN
C        LENS FOCAL LENGTH IN METERS
      REAL HCL(3), HCC(3), HCR(3)
C        LEFT, CENTER, AND RIGHT CAMERA POSITIONS IN CHASE VEHICLE BODY FRAME
      REAL HT(3)
C        DOCKING AID POSITION IN TARGET SPACECRAFT BODY FRAME
      REAL STATE(14)
C        TRUE CHASE VEHICLE STATE
      REAL T
C        ELAPSED TIME
      REAL TPHIH, TPHIV
C        TANGENTS OF HALF-FIELD-OF-VIEW ANGLES IN HORIZONTAL AND VERTICAL
C        DIRECTIONS FOR CAMERA SCANNING, RESPECTIVELY
      REAL U(3), V(3)
C        U IS HORIZONTAL DEFLECTION, V IS VERTICAL DEFLECTION, SUBSCRIPT
C        INDICATES CAMERA NUMBER (LEFT, CENTER, OR RIGHT)
      LOGICAL VALID, VALID1
C        TRUE IF MEASUREMENT VALID (LIGHTS IN FIELD OF VIEW)
      REAL VLT(3)
C        COORDINATES OF LAMP WITH RESPECT TO CAMERA
      REAL VL(3), V1(3), V2(3), V3(3), V5(3), V6(3), HC(3)
C        INTERMEDIATE RESULTS WITH NO PROBLEM-RELATED SIGNIFICANCE
C
      COMMON/OPTSYS/TPHIH, TPHIV, FOCLEN
      COMMON/HNOOFF/HCL, HCR, HT, DUMMY(6), HCC
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      VALID1=VALID
C     (* FIND LAMP POSITION (=VLT) WITH RESPECT TO CAMERA *)
C        (* 905 - COMPUTE TARGET ATTITUDE *)
            CALL TRGATT(TRNAT,T)
         CALL DIRMAT(STATE(10),AC TRNAC)
         DO 50 CAMERA=1,3
         GO TO (10,20,30), CAMERA
 10        CONTINUE
              CALL MSCL(HC,HCL,3 1,1 0)
              GO TO 40
 20        CONTINUE
              CALL MSCL(HC,HCC,3,1,1 0)
              GO TO 40
 30        CONTINUE
              CALL MSCL(HC,HCR,3,1,1 0)
 40      CONTINUE
C        (* FIND VECTOR FROM CAMERA TO LAMP *)
            CALL MMLT(V1,TRNAT,HT,3,3,1)
            CALL MSUB(V2,V1,STATE,3,1)
            CALL DIRMAT(STATE(10) AC,TRNAC)
            CALL MMLT(V3,AC,V2,3,3,1)
            CALL MSUB(VLT,V3,HC,3,1)
C        (* FIND DEFLECTION *)
            U(CAMERA)=-VLT(2)*FOCLEN/VLT(1)
            V(CAMERA)=VLT(3)*FOCLEN/VLT(1)
C        (* BE SURE DOCKING AID IS IN FIELD OF VIEW OF CAMERA *)
            VALID=VALID AND VLT(1) GT 0
            IF(VLT(1) LE 0 0 AND CAMERA EQ 2) VALID1= FALSE
            IF( NOT VALID) RETURN
C        (* BE SURE DOCKING AID IS ON VISIBLE SIDE OF TARGET SPACECRAFT *)
            CALL MMLT(V5,TRNAC,VLT,3,3,1)
            CALL MTRN(AT,TRNAT,3)
            CALL MMLT(V6,AT,V5,3,3,1)
            VALID=VALID AND V6(1) GT 0
            IF(V6(1) LE 0 0) VALID1= FALSE
C        (* CORRUPT CENTROID COORDINATES WITH NOISE *)
            U(CAMERA)=RANFN(2,U(CAMERA),TPHIH*FOCLEN/600 )
            V(CAMERA)=RANFN(2,V(CAMERA),TPHIV*FOCLEN/600 )
 50   CONTINUE
      RETURN
C
      END
```



```
C
      SUBROUTINE PROPTR(DT,F,STATE,T)
C     (* 420 - PROPAGATE TRUE STATE BY 4TH-ORDER RUNGE-KUTTA INTEGRATION *)
C
C     CALLS
C        COMPK1,COMPK2,COMPK3,COMPK4,POINT
C     CALLED BY
C        DOCK
C
C     INPUT
C        DT,F,STATE,T
C     OUTPUT
C        STATE,T
C
      REAL DT
C        INTEGRATION INTERVAL SIZE
      REAL F(14)
C        FORCES FROM THRUSTERS AFTER SELECTION
      INTEGER I
C        DO LOOP INDEX
      REAL K1(14),K2(14),K3(14),K4(14)
C        VECTORS K1 THRU K4 USED IN RUNGE-KUTTA INTEGRATION
      REAL GM
C        SQUARE ROOT OF SUM OF SQUARES OF QUATERNION ELEMENTS
      REAL STATE(14)
C        CHASE VEHICLE STATE VECTOR
      REAL STEP
C        STEP SIZE FOR INTEGRATION
      REAL T,TO
C        ELAPSED TIME AND TIME AT START OF INTEGRATION INTERVAL
      REAL TLEFT
C        TIME LEFT TO GO OUT OF DT
      PARAMETER STEPMX=0 2
C        MAXIMUM INTEGRATION STEP SIZE
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      TLEFT=DT
      TO=T
   10 IF(TLEFT LE 0) GO TO 40
         STEP=AMIN1(STEPMX,TLEFT)
         TLEFT=TLEFT-STEP
C        (* 421 - COMPUTE K1 *)
            CALL COMPK1(F,K1,STATE,STEP,T)
C        (* 422 - COMPUTE K2 *)
            CALL COMPK2(F,K1,STATE,STEP,T,K2)
C        (* 423 - COMPUTE K3 *)
            CALL COMPK3(F,K2,STATE,STEP,T,K3)
C        (* 424 - COMPUTE K4 *)
            CALL COMPK4(F,K3,STATE,STEP,T,K4)
C        (* COMPUTE NEW STATE *)
            DO 20 I=1,14
               STATE(I)=STATE(I)+(K1(I)+2 *K2(I)+2 *K3(I)+K4(I))/6
   20       CONTINUE
C        (* NORMALIZE QUATERNION *)
            GM=SQRT(STATE(10)**2+STATE(11)**2+STATE(12)**2+
     A         STATE(13)**2)
            DO 30 I=10,13
               STATE(I)=STATE(I)/GM
   30       CONTINUE
C        (* 425 - POINT SIMULATION CAMERA *)
            CALL POINT(STATE)
         T=T+STEP
         GO TO 10
   40 CONTINUE
      T=TO+DT
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE COMPK1(F,K1,STATE,STEP,T)
C     (* 421 - COMPUTE VECTOR K1 FOR RUNGE-KUTTA INTEGRATION *)
C
C     CALLS
C        STPRIM,MSCL,MADD
C     CALLED BY
C        PROPTR
C
C     INPUT
C        F,STATE,STEP,T,FULDIS,INERCV,MEMPTY
C     OUTPUT
C        K1
C
      REAL DSTATE(14)
C        TIME DERIVATIVE OF STATE VECTOR
      REAL F(14)
C        FORCES FROM THRUSTERS AFTER SELECTION
      REAL FULDIS(3,3)
C        CHASE VEHICLE FUEL-DISTRIBUTION INERTIA TENSOR
      INTEGER I
C        DO LOOP INDEX
      REAL INERFL(3,3)
C        INERTIA OF FUEL
      REAL INERTA(3,3)
C        INERTIA OF CHASE VEHICLE (LOADED)
      REAL K1(14)
C        VECTOR K1 USED IN RUNGE-KUTTA INTEGRATION
      REAL STATE(14)
C        CHASE VEHICLE STATE VECTOR
      REAL STEP
C        STEP SIZE FOR INTEGRATION
      REAL T
C        ELAPSED TIME
C
      COMMON/MASPRP/FULDIS,INERCV,MEMPTY
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* COMPUTE INERTIA *)
         CALL MSCL(INERFL,FULDIS,3,3,STATE(14)-MEMPTY)
         CALL MADD(INERTA,INERCV,INERFL,3,3)
C     (* 902 - COMPUTE STATE DERIVATIVE *)
         CALL STPRIM(STATE,F,INERTA,T,DSTATE)
C     (* COMPUTE K1=STEP*(STATE DERIVATIVE)  *)
         DO 10 I=1,14
            K1(I)=STEP*DSTATE(I)
   10    CONTINUE
      RETURN
C
      END
```



```
C
      SUBROUTINE COMPK2(F,K1,STATE,STEP,T,K2)
C     (* 422 - DETERMINE VECTOR K2, FOR RUNGE-KUTTA INTEGRATION *)
C
C     CALLS
C        MADD,MSCL,STPRIM
C     CALLED BY
C        PROPTR
C
C     INPUT
C        F,FULDIS,INERCV,K1,MEMPTY,STATE,STEP,T
C     OUTPUT
C        K2
C
      REAL DSTATE(14)
C        TIME DERIVATIVE OF STATE VECTOR
      REAL F(14)
C        FORCES FROM THRUSTERS AFTER SELECTION
      REAL FULDIS(3,3)
C        CHASE VEHICLE FUEL-DISTRIBUTION INERTIA TENSOR
      INTEGER I
C        DO LOOP INDEX
      REAL INERCV(3,3)
C        INERTIA OF EMPTY CHASE VEHICLE
      REAL INERFL(3,3)
C        INERTIA OF FUEL
      REAL INERTA(3,3)
C        INERTIA OF CHASE VEHICLE (LOADED)
      REAL K1(14),K2(14)
C        VECTORS K1 AND K2 USED IN RUNGE-KUTTA INTEGRATION
      REAL MEMPTY
C        MASS OF CHASE VEHICLE WITHOUT FUEL
      REAL STATE(14)
C        CHASE VEHICLE STATE VECTOR
      REAL STEP
C        STEP SIZE FOR INTEGRATION
      REAL T
C        ELAPSED TIME
      REAL TEMPST(14)
C        TEMPORARY STATE VECTOR
C
      COMMON/MASPRP/FULDIS,INERCV,MEMPTY
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* COMPUTE TEMPORARY STATE *)
         DO 10 I=1,14
            TEMPST(I)=STATE(I)+0.5*K1(I)
   10    CONTINUE
C     (* DETERMINE TEMPORARY INERTIA AS FUNCTION OF TEMPORARY
C        STATE MASS *)
         CALL MSCL(INERFL,FULDIS,3,3,TEMPST(14)-MEMPTY)
         CALL MADD(INERTA,INERCV,INERFL,3,3)
C     (* 902 - COMPUTE DERIVATIVE AT TEMPORARY STATE *)
         CALL STPRIM(TEMPST,F,INERTA,T+.5*STEP,DSTATE)
C     (* COMPUTE K2=STEP*(DERIVATIVE AT TEMPORARY STATE *)
         DO 20 I=1,14
            K2(I)=STEP*DSTATE(I)
   20    CONTINUE
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE COMPK3(F,K2,STATE,STEP,T,K3)
C     (* 423 - DETERMINE VECTOR K3, FOR RUNGE-KUTTA INTEGRATION *)
C
C     CALLS
C        MADD,MBCL,STPRIM
C     CALLED BY
C        PROPTR
C
C     INPUT
C        F,FULDIS,INERCV,K2,MEMPTY,STATE,STEP,T
C     OUTPUT
C        K3
C
      REAL DSTATE(14)
C        TIME DERIVATIVE OF STATE VECTOR
      REAL F(14)
C        FORCES FROM THRUSTERS AFTER SELECTION
      REAL FULDIS(3,3)
C        CHASE VEHICLE FUEL-DISTRIBUTION INERTIA TENSOR
      INTEGER I
C        DO LOOP INDEX
      REAL INERCV(3,3)
C        INERTIA OF EMPTY CHASE VEHICLE
      REAL INERFL(3,3)
C        INERTIA OF FUEL
      REAL INERTA(3,3)
C        INERTIA OF CHASE VEHICLE (LOADED)
      REAL K2(14),K3(14)
C        VECTORS K2 AND K3 USED IN RUNGE-KUTTA INTEGRATION
      REAL MEMPTY
C        MASS OF CHASE VEHICLE WITHOUT FUEL
      REAL STATE(14)
C        CHASE VEHICLE STATE VECTOR
      REAL STEP
C        STEP SIZE FOR INTEGRATION
      REAL T
C        ELAPSED TIME
      REAL TEMPST(14)
C        TEMPORARY STATE VECTOR
C
      COMMON/MASPRP/FULDIS,INERCV,MEMPTY
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* COMPUTE TEMPORARY STATE *)
      DO 10 I=1,14
         TEMPST(I)=STATE(I)+0.5*K2(I)
   10 CONTINUE
C     (* DETERMINE TEMPORARY INERTIA AS A FUNCTION OF TEMPORARY
C              STATE MASS *)
      CALL MBCL(INERFL,FULDIS,3,3,TEMPST(14)-MEMPTY)
      CALL MADD(INERTA,INERCV,INERFL,3,3)
C     (* 902 - COMPUTE DERIVATIVE AT TEMPORARY STATE *)
      CALL STPRIM(TEMPST,F,INERTA,T+.5*STEP,DSTATE)
C     (* COMPUTE K3=STEP*(DERIVATIVE AT TEMPORARY STATE *)
      DO 20 I=1,14
         K3(I)=STEP*DSTATE(I)
   20 CONTINUE
      RETURN
C
      END
```



```
C
      SUBROUTINE COMPK4(F,K3,STATE,STEP,T,K4)
C     (* 424 - DETERMINE VECTOR K4, FOR RUNGE-KUTTA INTEGRATION *)
C
C     CALLS
C        MADD,MBCL,STPRIM
C     CALLED BY
C        PROPTR
C
C     INPUT
C        F,FULDIS,INERCV,K3,MEMPTY,STATE,STEP,T
C     OUTPUT
C        K4
C
      REAL DSTATE(14)
C        TIME DERIVATIVE OF STATE VECTOR
      REAL F(14)
C        FORCES FROM THRUSTERS AFTER SELECTION
      REAL FULDIS(3,3)
C        CHASE VEHICLE FUEL-DISTRIBUTION INERTIA TENSOR
      INTEGER I
C        DO LOOP INDEX
      REAL INERCV(3,3)
C        INERTIA OF EMPTY CHASE VEHICLE
      REAL INERFL(3,3)
C        INERTIA OF FUEL
      REAL INERTA(3,3)
C        INERTIA OF CHASE VEHICLE (LOADED)
      REAL K3(14),K4(14)
C        VECTORS K3 AND K4 USED IN RUNGE-KUTTA INTEGRATION
      REAL MEMPTY
C        MASS OF CHASE VEHICLE WITHOUT FUEL
      REAL STATE(14)
C        CHASE VEHICLE STATE VECTOR
      REAL STEP
C        STEP SIZE FOR INTEGRATION
      REAL T
C        ELAPSED TIME
      REAL TEMPST(14)
C        TEMPORARY STATE VECTOR
C
      COMMON/MASPRP/FULDIS,INERCV,MEMPTY
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* COMPUTE TEMPORARY STATE *)
      DO 10 I=1,14
         TEMPST(I)=STATE(I)+K3(I)
   10 CONTINUE
C     (* DETERMINE TEMPORARY INERTIA AS A FUNCTION OF TEMPORARY
C              STATE MASS *)
      CALL MBCL(INERFL,FULDIS,3,3,TEMPST(14)-MEMPTY)
      CALL MADD(INERTA,INERCV,INERFL,3,3)
C     (* 902 - COMPUTE DERIVATIVE AT TEMPORARY STATE *)
      CALL STPRIM(TEMPST,F,INERTA,T+STEP,DSTATE)
C     (* COMPUTE K4=STEP*(DERIVATIVE AT TEMPORARY STATE *)
      DO 20 I=1,14
         K4(I)=STEP*DSTATE(I)
   20 CONTINUE
      RETURN
C
      END
```

```
C
      SUBROUTINE POINT(STATE)
C     (* 425 - POSITION SIMULATION CAMERA *)
C     (* THIS DUMMY SUBROUTINE IS FOR LATER EXPANSION TO PHYSICAL SIMULATION*)
C
C     CALLS
C        <NONE>
C     CALLED BY
C        PROPTR
C
      RETURN
C
      END
```



```
C
      SUBROUTINE POSIT(U,V,RELPOS,RHO,PITCH,YAW,VALID)
C     (* 430 - CALCULATE POSITIONS AT TIME OF MEASUREMENT *)
C
C      CALLS
C         <NONE>
C      CALLED BY
C         DOCK
C
C     INPUT
C        U,V,PITCH,YAW
C     OUTPUT
C        RELPOS,RHO,VALID
C
      LOGICAL VALID
C        TRUE IF MEASUREMENT VALID
      REAL FOCLEN
C        LENS FOCAL LENGTH IN METERS
      REAL HCL(3),HCR(3)
C        LEFT AND RIGHT CAMERA POSITIONS IN CHASE VEHICLE BODY FRAME
      INTEGER I
C        INDEX OF DO LOOP
      REAL PITCH
C        CORRUPTED VERSION OF ANGULAR ERROR FROM THE -X/Y PLANE MEASURED
C        IN THE PROJECTION OF THE POSITION ONTO THE -X/Z PLANE
      REAL RELPOS(3,2)
C        RELATIVE CENTER-CAMERA COORDINATES IN CURRENT DOCKING AID
C        FRAME CENTERED AT BEACON LIGHT (1ST COLUMN IS CURRENT MEASUREMENT,
C        2ND COLUMN IS PREVIOUS MEASUREMENT)
      REAL RHO
C        MEASURED RANGE,CAMERA TO TARGET DOCKING AID CENTER LIGHT
      REAL TEMP,TEST
C        INTERMEDIATE RESULTS WITH NO PROBLEM-RELATED SIGNIFICANCE
      REAL TPHIH,TPHIV
C        TANGENT OF HALF-FIELD-OF-VIEW ANGLE IN HORIZONTAL AND VERTICAL
C        DIRECTIONS FOR CAMERA SCANNING, RESPECTIVELY
      REAL U(3),V(3)
C        U IS HORIZONTAL DEFLECTION, V IS VERTICAL DEFLECTION. SUBSCRIPT
C        INDICATES CAMERA NUMBER (LEFT, CENTER, OR RIGHT)
      REAL X
C        DUMMY ARGUMENT OF TANGENT FUNCTION
      REAL YAW
C        CORRUPTED VERSION OF ANGULAR ERROR FROM THE -X/Z PLANE MEASURED
C        IN THE PROJECTION OF THE POSITION ONTO THE -X/Y PLANE
C
      COMMON/HNGOFF/HCL,HCR,DUMMY(12)
      COMMON/OPTSYS/TPHIH,TPHIV,FOCLEN
C
C     (* STATEMENT FUNCTION DEFINITION FOR TANGENT *)
        TAN(X)=SIN(X)/COS(X)
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* BE SURE LIGHT IS IN FIELD OF VIEW *)
        TEST=TPHIH*FOCLEN
        VALID=VALID  AND  ABS(U(1)) LE TEST  AND  ABS(U(3)).LE TEST
     A       AND. ABS(V(2)) LE TPHIV*FOCLEN
        IF( NOT VALID) GO TO 20
C     (* CALCULATE POSITIONS AT TIME OF MEASUREMENT *)
        TEMP=U(1)-U(3)
        RHO=FOCLEN*(HCL(2)-HCR(2))*TEMP/AMAX1(TEMP**2-4.0*(TPHIH*
     A       FOCLEN/600.)**2,1 E-15)  *  SQRT((U(2)/FOCLEN)**2+
     B       (V(2)/FOCLEN)**2+1 0)
C     (* SAVE OLD MEASUREMENTS *)
        DO 10 I=1,3
           RELPOS(I,2)=RELPOS(I,1)
   10   CONTINUE
C     (* CALCULATE NEW MEASUREMENTS *)
        RELPOS(1,1)=-RHO/SQRT(1 0+(TAN(PITCH))**2+(TAN(YAW))**2)
        RELPOS(2,1)=RELPOS(1,1)*TAN(YAW)
        RELPOS(3,1)=-RELPOS(1,1)*TAN(PITCH)
   20 RETURN
C
      END
```

```
C
      SUBROUTINE ATITUD(ACV,ACVT,RELPOS,RHO,U,V,AT,CVPOS,CAMPOS)
C     (* 440 - DETERMINE TARGET ATTITUDE AND CHASE VEHICLE POSITION IN PRIMARY
C        REFERENCE FRAME *)
C
C     CALLS
C        FINDCV,DIRMAT,QUATRN,MADD,MML   MSUB,MTRN
C     CALLED BY
C        DOCK
C
C     INPUT
C        RELPOS,RHO,U,V,ACV,ACVT,HC,HT,AT,CAMPOS
C     OUTPUT
C        AT,CVPOS
C
      REAL ACV(3,3),ACVT(3,3)
C        DIRECTION COSINE MATRIX DESCRIBING CURRENT CHASE VEHICLE ATTITUDE
C        RELATIVE TO PRIMARY REFERENCE FRAME, AND ITS TRANSPOSE
      REAL AT(3,3),ATT(3,3)
C        DIRECTION COSINE MATRIX OF TARGET SPACECRAFT (IN PRIMARY REF  FRAME),
C        AND TRANSPOSE OF THIS MATRIX
      REAL CAMPOS(3,2)
C        MEASURED CENTER-CAMERA POSITION IN COORD  SYS  CENTERED AT DOCKING AID
C        CENTER AND PARALLEL TO PRIMARY REFERENCE FRAME (1ST COLUMN
C        IS CURRENT MEASUREMENT, 2ND COLUMN IS PREVIOUS MEASUREMENT
      REAL CVPOS(3)
C        MEASURED CHASE VEHICLE POSITION IN PRIMARY REFERENCE FRAME
      REAL HCC(3),HT(3)
C        POSITION OF CENTER CAMERA IN CHASE VEHICLE BODY FRAME AND DOCKING AID
C        POSITION IN TARGET SPACECRAFT BODY FRAME, RESPECTIVELY
      REAL QT(4)
C        QUATERNION CORRESPONDING TO AT
      REAL RELPOS(3)
C        RELATIVE CHASE VEHICLE COORDINATES IN CURRENT TARGET SPACECRAFT
C        REFERENCE FRAME
      REAL RHO
C        DISTANCE BETWEEN LIGHTS AND CAMERA
      REAL U(3),V(3)
C        HORIZONTAL AND VERTICAL COMPONENTS OF LIGHT CENTROID AS MEASURED
C        BY THE CENTER CAMERA
      REAL V1(3),V2(3),V3(3)
C        INTERMEDIATE RESULTS WITH NO PROBLEM-RELATED SIGNIFICANCE
C
      COMMON/HNDOFF/DUMMY1(6),HT,DUMMY(6),HCC
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* 441 - COMPUTE CAMERA POSITION WITH RESPECT TO DOCKING AID *)
      CALL FINDCV(ACVT,RHO,U(2),V(2),CAMPOS)
C     (* 442 - COMPUTE TARGET ATTITUDE IN PRIMARY REFERENCE FRAME *)
      CALL QUATRN(CAMPOS,RELPOS,QT,AT)
C     (* 901 - COMPUTE TRANSPOSE OF TARGET DIRECTION COSINE MATRIX *)
      CALL MTRN(ATT,AT,3)
C     (* COMPUTE POSITION OF CHASE VEHICLE CENTER OF GRAVITY IN PRIMARY REF
C        FRAME *)
      CALL MMLT(V2,ATT,HT,3,3,1)
      CALL MADD(V1,CAMPOS,V2,3,1)
      CALL MMLT(V3,ACVT,HCC,3,3,1)
      CALL MSUB(CVPOS,V1,V3,3,1)
      RETURN
C
      END
```



```
C
      SUBROUTINE FINDCV(ACVT,RHO,UC,VC,CAMPOS)
C     (* 441 - COMPUTE POSITION OF CAMERA FRAME PARALLEL TO PRIMARY REF  FRAME
C        BUT CENTERED AT DOCKING AID LIGHT *)
C
C     CALLS
C        MMLT,SQRT
C     CALLED BY
C        ATITUD
C
C     INPUT
C        ACVT,RHO,UC,VC,FOCLEN
C     OUTPUT
C        CAMPOS
C
      REAL ACVT(3,3)
C        DIRECTION COSINE MARTIX DESCRIBING CURRENT CHASE VEHICLE ATTITUDE
C        RELATIVE TO PRIMARY REFERENCE FRAME
      REAL CAMPOS(3,2)
C        MEASURED CENTER-CAMERA POSITION IN FRAME PARALLEL TO PRIMARY REF  FRAME
C        BUT CENTERED AT DOCKING AID   (1ST COLUMN IS CURRENT MEASUREMENT,
C        2ND COLUMN IS PREVIOUS MEASUREMENT )
      REAL FOCLEN
C        CAMERA LENS FOCAL LENGTH, IN METERS
      INTEGER I
C        INDEX OF DO LOOP
      REAL RATIO
C        INTERMEDIATE RESULTS, NO PROBLEM RELATED SIGNIFICANCE
      REAL RHO
C        MEASURED DISTANCE BETWEEN DOCKIN AID LIGHT AND CAMERA
      REAL UC,VC
C        HORIZONTAL AND VERTICAL COMPONENTS OF THE TARGET LIGHT
C        IMAGE CENTROID AS MEASURED BY CENTER CAMERA
      REAL VEC1(3)
C        NEGATIVE OF TARGET POSITION IN CAMERA FRAME
C
      COMMON/OPTSYS/DUMMY1(2) FOCLEN
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* COMPUTE SCALAR MULTIPLIER *)
      RATIO=RHO/SQRT(FOCLEN**2+UC**2+VC**2)
C     (* FIND NEG  OF TARGET POSITION IN CAMERA FRAME *)
      VEC1(1)=-FOCLEN*RATIO
      VEC1(2)=UC*RATIO
      VEC1(3)=-VC*RATIO
C     (* SAVE OLD MEASUREMENTS *)
      DO 10 I=1,3
         CAMPOS(I,2)=CAMPOS(I,1)
   10 CONTINUE
C     (* CONVERT TO CAMERA POSITION IN DOCKING AID LIGHT FRAME *)
      CALL MMLT(CAMPOS,ACVT,VEC1,3,3,1)
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE QUATRN(CAMPOS,RELPOS,QT,AT)
C     (* 442 - COMPUTE DIRECTION COSINE MATRIX OF TARGET FROM LAST TWO
C        MEASUREMENTS *)
C
C     CALLS
C        DPRD,XPRD,MSCL,MMLT,TOQUAT,MTRN
C     CALLED BY
C        ATITUD
C
C     INPUTS
C        CAMPOS,RELPOS
C     OUTPUTS
C        QT,AT
C
      REAL AT(3,3),QT(4)
C        MEASURED DIRECTION COSINE MATRIX OF TARGET SPACECRAFT AND
C        CORRESPONDING QUATERNION, RESPECTIVELY
      REAL CAMPOS(3,2)
C         MEASURED CENTER-CAMERA POSITION IN COORD  SYS  CENTERED AT DOCKING AID
C         CENTER AND PARALLEL TO PRIMARY REFERENCE FRAME
C         FIRST COLUMN IS LATEST MEASUREMENT, SECOND COLUMN IS PREVIOUS ONE
      REAL MC(3,3)
C        MATRIX CONTAINING 3 COLUMN VECTORS  FIRST IS UNIT VECTOR IN
C        DIRECTION OF CAMPOS, SECOND IS UNIT VECTOR IN DIRECTION PERPENDICULAR
C        TO  PREVIOUS AND CURRENT VALUES OF CAMPOS (COLUMNS 1 AND 2 OF CAMPOS,
C        RESPECTIVELY), THIRD COLUMN IS UNIT VECTOR IN DIRECTION PERPENDICULAR
C        TO VECTORS IN FIRST TWO COLUMNS
      REAL MT(3,3)
C        MATRIX CONTAINING 3 COLUMN VECTORS  FIRST IS UNIT VECTOR IN
C        DIRECTION OF RELPOS, SECOND IS UNIT VECTOR IN DIRECTION PERPENDICULAR
C        TO PREVIOUS AND CURRENT VALUES OF RELPOS (COLUMNS 1 AND 2 OF
C        RELPOS, RESPECTIVELY), AND THIRD COLUMN IS UNIT VECTOR IN
C        DIRECTION PERPENDICULAR TO VECTORS IN FIRST TWO COLUMNS
      REAL TRNMC(3,3)
C        TRANSPOSE OF MC
      REAL RELPOS(3,2)
C        RELATIVE CENTER-CAMERA COORDINATES IN CURRENT DOCKING AID FRAME
C        CENTERED AT BEACON LIGHT  FIRST COLUMN IS LATEST MEASUREMENT, SECOND
C        COLUMN IS PREVIOUS ONE
      REAL RX(3),CX(3),XP(3),XP1(3)
C        INTERMEDIATE RESULTS WITH NO SIMPLE PHYSICAL INTERPRETATION
```



```
C
C     (* IF OLD, NEW MEASUREMENTS ARE TOO SIMILAR, RETURN *)
        CALL XPRD(XP,CAMPOS,CAMPOS(1,2))
        CALL XPRD(XP1,RELPOS,RELPOS(1,2))
        IF(DPRD(XP,XP,3) LE 1 E-3 OR DPRD(XP1,XP1,3) LE 1 E-3) RETURN
C     (* COMPUTE FIRST COLUMN OF MC MATRIX *)
        CALL MSCL(MC,CAMPOS,3,1,1 /SQRT(DPRD(CAMPOS,CAMPOS,3)))
C     (* COMPUTE SECOND COLUMN OF MC MATRIX *)
        CALL XPRD(CX,CAMPOS,CAMPOS(1,2) )
        CALL MSCL(MC(1,2),CX,3,1,1 /SQRT(DPRD(CX,CX,3)))
C     (* COMPUTE THIRD COLUMN OF MC MATRIX *)
        CALL XPRD(MC(1,3),MC,MC(1,2))
C     (* COMPUTE FIRST COLUMN OF MT MATRIX *)
        CALL MSCL(MT,RELPOS,3,1,1 /SQRT(DPRD(RELPOS,RELPOS,3)))
C     (* COMPUTE SECOND COLUMN OF MT MATRIX *)
        CALL XPRD(RX,RELPOS,RELPOS(1,2))
        CALL MSCL(MT(1,2),RX,3,1,1 /SQRT(DPRD(RX,RX,3)))
C     (* COMPUTE THIRD COLUMN OF MT MATRIX *)
        CALL XPRD(MT(1,3),MT,MT(1,2))
C     (* COMPUTE MEASURED ATTITUDE MATRIX OF TARGET SPACECRAFT *)
        CALL MTRN(TRNMC,MC,3)
        CALL MMLT(AT,MT,TRNMC,3,3,3)
C     (* 442 1 - COMPUTE MEASURED QUATERNION OF TARGET SPACECRAFT *)
        CALL TOQUAT(AT,QT)
      RETURN
C
        END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE TOQUAT(A,Q)
C     (* 442 1 - CONVERT DIRECTION COSINE MATRIX TO QUATERNION *)
C
C     CALLS
C        SQR1
C     CALLED BY
C        QUATRN
C
C     INPUT
C        A
C     OUTPUT
C        Q
C
      REAL A(3,3)
C        DIRECTION COSINE MATRIX
      REAL Q(4)
C        CORRESPONDING QUATERNION
      REAL FACTOR
C        INTERMEDIATE RESULTS
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      Q(4)=0.5*SQR1(1.+A(1,1)+A(2,2)+A(3,3))
      IF(Q(4).LT.0.5) GO TO 10
         FACTOP=0.25/Q(4)
         Q(1)=FACTOR*(A(2,3)-A(3,2))
         Q(2)=FACTOR*(A(3,1)-A(1,3))
         Q(3)=FACTOR*(A(1,2)-A(2,1))
         GO TO 40
   10 Q(1)=0.5*SQR1(1.+A(1,1)-A(2,2)-A(3,3))
      IF(Q(1).LT.0.5) GO TO 20
         FACTOR=0.25/Q(1)
         Q(2)=FACTOR*(A(1,2)+A(2,1))
         Q(3)=FACTOR*(A(1,3)+A(3,1))
         Q(4)=FACTOR*(A(2,3)-A(3,2))
         GO TO 40
   20 Q(2)=0.5*SQR1(1.+A(2,2)-A(1,1)-A(3,3))
      IF(Q(2).LT.0.5) GO TO 30
         FACTUR=0.25/Q(2)
         Q(1)=FACTOR*(A(1,2)+A(2,1))
         Q(3)=FACTOR*(A(2,3)+A(3,2))
         Q(4)=FACTOR*(A(3,1)-A(1,3))
         GO TO 40
   30 CONTINUE
         Q(3)=0.5*SQR1(1.+A(3,3)-A(2,2)-A(1,1))
         FACTOR=0.25/Q(3)
         Q(1)=FACTOR*(A(1,3)+A(3,1))
         Q(2)=FACTOR*(A(2,3)+A(3,2))
         Q(4)=FACTOR*(A(1,2)-A(2,1))
   40 CONTINUE
      IF(Q(4).GE.0.) GO TO 50
         Q(1)=-Q(1)
         Q(2)=-Q(2)
         Q(3)=-Q(3)
         Q(4)=-Q(4)
   50 CONTINUE
      RETURN
C
      END
```



```
C
      SUBROUTINE INCORP(CVPOS,P,ESTATE)
C     (* 450 - INCORPORATE MEASUREMENT *)
C
C     CALLS
C        UPDCOV,UPDSTA,COMPG,ESTCOV,KALGAN
C     CALLED BY
C        DOCK
C
C     INPUT
C        ESTATE,P,R
C     OUTPUT
C        P,ESTATE
C
      REAL CVPOS(3)
C        MEASURED CHASE VEHICLE POSITON IN PRIMARY REFERENCE FRAME
      REAL ESTATE(6)
C        ESTIMATE OF STATE VECTOR
      REAL G(3,6)
C        PARTIAL OF MEASUREMENT WITH RESPECT TO STATE
      REAL KGAIN(6,3)
C        KALMAN GAIN MATRIX
      REAL P(6,6)
C        COVARIANCE OF STATE ESTIMATE
      REAL R(3,3)
C        MEASUREMENT COVARIANCE
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* 451 - COMPUTE JACOBIAN G *)
         CALL COMPG(ESTATE,G)
C     (* 452 - ESTIMATE MEASUREMENT COVARIANCE *)
         CALL ESTCOV(ESTATE,R)
C     (* 453 - COMPUTE KALMAN GAIN MATRIX *)
         CALL KALGAN(R,P,G,KGAIN)
C     (* 454 - UPDATE STATE *)
         CALL UPDSTA(ESTATE,KGAIN,CVPOS)
C     (* 455 - UPDATE COVARIANCE MATRIX *)
         CALL UPDCOV(P,KGAIN,G)
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE COMPG(ESTATE,G)
C     (* 451 - COMPUTE PARTIAL OF MEASUREMENT WITH RESPECT TO STATE *)
C
C     CALLS
C        <NONE>
C     CALLED BY
C        INCORP
C
C     INPUT
C        ESTATE
C     OUTPUT
C        G
C
      REAL ESTATE(6)
C        STATE ESTIMATE
      REAL G(3,6)
C        JACOBIAN
      INTEGER I,J
C        DO LOOP INDICES
C- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* COMPUTE G *)
         DO 10 I=1,3
            DO 10 J=1,6
               G(I,J)=0
 10      CONTINUE
         DO 20 I=1,3
            G(I,I)=1
 20      CONTINUE
      RETURN
C
      END
```



```
C
      SUBROUTINE ESTCOV(ESTATE,P)
C     (* 452 - ESTIMATE MEASUREMENT COVARIANCE *)
C
C     CALLS
C        <NONE>
C     CALLED BY
C        INCORP
C
C     INPUT
C        ESTATE
C     OUTPUT
C        R
C
      REAL ESTATE(6)
C        ESTIMATE OF STATE VECTOR
      INTEGER I,J
C        DO LOOP INDICES
      REAL R(3,3)
C        MEASUREMENT COVARIANCE
      REAL VARNCE
C        ESTIMATED VARIANCE (PER AXIS)
C- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      DO 10 I=1,3
         DO 10 J=1,3
            R(I,J)=0 0
 10   CONTINUE
      VARNCE=2 OOE-8*(ESTATE(1)**2+ESTATE(2)**2+ESTATE(3)**2)**2+ 26
      DO 20 I=1,3
         R(I,I)=VARNCE
 20   CONTINUE
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE KALGAN(R,P,G,KGAIN)
C     (* 453 - COMPUTE KALMAN GAIN MATRIX *)
C
C     CALLS
C        MADD,MINV,MMLT
C     CALLED BY
C        INCORP
C
C     INPUT
C        R,P,G
C     OUTPUT
C        KGAIN
C
      INTEGER ERR
C        ERROR CODE (=0 IF NO ERROR)
      REAL G(3,6)
C        PARTIAL OF MEASUREMENT WITH PESPECT TO STATE
      REAL GT(6,3)
C        TRANSPOSE OF G
      INTEGER I,J
C        DO LOOP INDICES
      REAL KGAIN(6,3)
C        KALMAN GAIN MATRIX
      REAL P(6,6)
C        STATE ESTIMATE COVARIANCE
      REAL R(3,3)
C        MEASUREMENT COVARIANCE
      REAL SCRACH(4,6)
C        SCRATCHPAD FOR ROUTINE MINV
      REAL TEMP1(6,3),TEMP2(3,3),TEMP3(3,3)
C        INTERMEDIATE RESULTS WITH NO PROBLEM-RELATED SIGNIFICANCE
      INTEGER TERMNL
C        FORTRAN UNIT NUMBER FOR TERMINAL
C
      COMMON/SIMUL/TERMNL,IDUMMY
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* COMPUTE KGAIN=P*TRN(G)*INV(R+G*P*TRN(G)) *)
         DO 10 I=1,3
            DO 10 J=1,6
               GT(J,I)=G(I,J)
   10    CONTINUE
         CALL MMLT(TEMP1,P,GT,6,6,3)
         CALL MMLT(TEMP2,G,TEMP1,3,6,3)
         CALL MADD(TEMP3,R,TEMP2,3,3)
         CALL MINV(TEMP2,TEMP3,3,SCRACH,4,6,ERR)
         IF(ERR NE 0) GO TO 20
         CALL MMLT(TEMP1,GT,TEMP2,6,3,3)
         CALL MMLT(KGAIN,P,TEMP1,6,6,3)
         RETURN
C
   20 CONTINUE
C     (* REPORT ERROR *)
         WRITE(TERMNL,901)
         STOP
C
  901 FORMAT('MATRIX INVERSION FAILURE IN SUBROUTINE KALGAN')
C
      END
```



```
C
      SUBROUTINE UPDSTA(ESTATE,KGAIN,CVPOS)
C     (* 454 - UPDATE STATE ESTIMATE *)
C
C     CALLS
C        MMLT
C     CALLED BY
C        INCORP
C
C     INPUT
C        RELPOS,AT,ESTATE
C     OUTPUT
C        ESTATE
C
      REAL CVPOS(3)
C        MEASURED CHASE VEHICLE POSITION IN PRIMARY REFERENCE FRAME
      REAL DELTA(6)
C        ADJUSTMENT TO ESTATE
      REAL ESTATE(6)
C        ESTIMATE OF STATE VECTOR
      INTEGER I
C        DO LOOP INDEX
      REAL KGAIN(6,3)
C        KALMAN GAIN MATRIX
      REAL POSERR(3)
C        MEASURED POSITION MINUS PREDICTED POSITION
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* COMPUTE ESTATE=ESTATE+KGAIN*(CVPOS-PREDICTED POSITION) *)
         DO 10 I=1,3
            POSERR(I)=CVPOS(I)-ESTATE(I)
   10    CONTINUE
         CALL MMLT(DELTA,KGAIN,POSERR,6,3,1)
         DO 20 I=1,6
            ESTATE(I)=ESTATE(I)+DELTA(I)
   20    CONTINUE
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C     SUBROUTINE UPDCOV(P,KGAIN,G)
C     (* 455 - UPDATE COVARIANCE MATRIX FROM (P=(I-K*G)*P)   *)
C
C     CALLS
C        MMLT
C     CALLED BY
C        INCORP
C
C     INPUT
C        P,KGAIN,G
C     OUTPUT
C        P
C
      REAL G(3,6)
C        PARTIAL OF MEASUREMENT WITH RESPECT TO STATE
      INTEGER I,J
C        DO LOOP INDICES
      REAL KGAIN(6,3)
C        KALMAN GAIN MATRIX
      REAL P(6,6)
C        COVARIANCE OF STATE ESTIMATE
      REAL TEMP(6,6),P1(6,6)
C        INTERMEDIATE RESULTS WITH NO PROBLEM-RELATED SIGNIFICANCE
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* COMPUTE I-KGAIN*G *)
         CALL MMLT(TEMP,KGAIN,G,6,3,6)
         DO 10 I=1,6
            DO 10 J=1,6
               TEMP(I,J)=-TEMP(I,J)
   10    CONTINUE
         DO 20 I=1,6
            TEMP(I,I)=TEMP(I,I)+1.0
   20    CONTINUE
C     (* COMPUTE NEW P *)
         CALL MMLT(P1,TEMP,P,6,6,6)
         DO 30 I=1,6
            DO 30 J=1,6
               P(I,J)=P1(I,J)
   30    CONTINUE
      RETURN
C
      END
```



```
C     SUBROUTINE PROPES(P,ESTATE,F,STEP,ACVT)
C     (* 460 - PROPAGATE STATE ESTIMATE AND COVARIANCE *)
C
C     CALLS
C        MMLT,MSCL
C     CALLED BY
C        DOCK
C
C     INPUT
C        P,ESTATE,ACVT,F,STEP
C     OUTPUT
C        P,ESTATE
C
      REAL ACCEL(3)
C        ESTIMATED ACCELERATION
      REAL ACVT(3,3)
C        TRANSPOSE OF CHASE VEHICLE DIRECTION COSINE MATRIX
      REAL AK2(3,14)
C        MATRIX RELATING FORCES FROM INDIVIDUAL THRUSTERS TO NET FORCE VECTOR
      REAL AMAGSQ
C        SQUARE OF MAGNITUDE OF TOTAL ACCELERATION
      REAL AVAR
C        ESTIMATED ACCELERATION VARIANCE
      REAL DSTATE(6)
C        CHANGE IN STATE
      REAL DP(6,6)
C        CHANGE IN COVARIANCE
      REAL ESTATE(6)
C        ESTIMATED STATE
      REAL F(14)
C        FORCES FROM THRUSTERS AFTER SELECTION
      REAL FBOD(3)
C        FORCES ALONG CHASE VEHICLE BODY AXES
      REAL FORCE(3)
C        FORCES IN PRIMARY REFERENCE FRAME
      INTEGER I,J
C        DO LOOP INDICES
      REAL P(6,6)
C        COVARIANCE OF STATE ESTIMATE
      REAL STEP
C        INTEGRATION TIME STEP
      PARAMETER AVGMAS=3650
C        ESTIMATED CHASE VEHICLE MASS
C
      COMMON/VEHIC/DUMMY1(14),AK2,DUMMY2(57)
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
C     (* COMPUTE FORCE=ACVT*AK2*F *)
      CALL MMLT(FBOD,AK2,F,3,14,1)
      CALL MMLT(FORCE,ACVT,FBOD,3,3,1)
C     (* ESTIMATE ACCELERATION *)
      CALL MBCL(ACCEL,FORCE,3,1,1./AVGMAS)
C     (* ESTIMATE ACCELERATION VARIANCE *)
      AMAGSQ=ACCEL(1)**2+ACCEL(2)**2+ACCEL(3)**2
      AVAR=0.05*AMAGSQ+1.E-7
C     (* FIND POSITION CHANGE *)
      DO 10 I=1,3
         DSTATE(I)=(ESTATE(I+3)+.5*ACCEL(I)*STEP)*STEP
10    CONTINUE
C     (* FIND VELOCITY CHANGE *)
      DO 20 I=4,6
         DSTATE(I)=STEP*ACCEL(I-3)
20    CONTINUE
C     (* FIND COVARIANCE CHANGE *)
      DO 30 I=4,6
         DO 30 J=1,6
            DP(I,J)=0.0
30    CONTINUE
      DP(4,4)=AVAR
      DP(5,5)=AVAR
      DP(6,6)=AVAR
      DO 40 I=1,3
         DO 40 J=1,6
            DP(I,J)=P(I+3,J)
40    CONTINUE
      DO 50 I=1,6
         DO 50 J=1,3
            DP(I,J)=DP(I,J)+P(I,J+3)
50    CONTINUE
C     (* COMPUTE NEW STATE AND COVARIANCE *)
      DO 60 I=1,6
         ESTATE(I)=ESTATE(I)+DSTATE(I)
         DO 60 J=1,6
            P(I,J)=P(I,J)+DP(I,J)
60    CONTINUE
      RETURN
C
      END
```



```
C
      SUBROUTINE SETGOL(ESTATE,P,T,ACV,AT,GXL,GXH,GYL,GYH,GZL,GZH,
     A   DEDLIN,VALID)
C     (* 470 - SET GOAL *)
C
C     CALLS
C        SQR1,AMAX1,MTRN,MMLT,MSUB
C     CALLED BY
C        DOCK
C
C     INPUT
C        ESTATE,P,DOKRAD,T,ACV,AT
C     OUTPUT
C        GXL,GXH,GYL,GYH,GZL,GZH,DEDLIN
C
      REAL ACV(3,3),AT(3,3),TRNAT(3,3)
C        DIRECTION COSINE MATRICES FOR CHASE VEHICLE & TARGET SPACECRAFT
C        AND TRANSPOSE OF TARGET DIRECTION COSINE MATRIX
      REAL D
C        3-SIGMA POSITION UNCERTAINTY
      REAL DEDLIN
C        TIME BY WHICH STRATEGY LOGIC WANTS CHASE VEHICLE IN GOAL 'BOX'
      REAL DOKRAD
C        RADIUS FROM TARGET SPACECRAFT WITHIN WHICH THE SPACECRAFT ARE
C        CONSIDERED DOCKED
      REAL ESTATE(6)
C        ESTIMATE OF STATE VECTOR
      REAL GXL,GXH,GYL,GYH,GZL,GZH
C        X,Y,Z LOWER AND UPPER LIMITS OF GOAL 'BOX'
      REAL HDC(3),HDT(3)
C        DOCKING FIXTURES' POSITIONS IN THEIR RESPECTIVE SPACECRAFT
C        COORDINATE SYSTEMS
      INTEGER I,J
C        DO LOOP INDICES
      REAL P(6,6)
C        COVARIANCE OF STATE ESTIMATE
      REAL SLOP
C        ALLOWED ERROR IN POSITION (PER AXIS)
      REAL T
C        ELAPSED TIME
      REAL V1(3),V2(3),V3(3),R(3),V4(3),V5(3)
C        INTERMEDIATE RESULTS
      LOGICAL VALID
C        TRUE IF LAST MEASUREMENT WAS VALID
C
      COMMON/VEHIC/DUMMY1(98),DOKRAD,DUMMY2(14)
      COMMON/HNGOFF/DUMMY3(9),HDC,HDT,HCC
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      IF(SQRT(ESTATE(1)**2+ESTATE(2)**2+ESTATE(3)**2) LT 30   AND
     A   SQRT(ESTATE(4)**2+ESTATE(5)**2+ESTATE(6)**2) LT  5)
     B   GO TO 10
         D=3 *SQRT(P(1,1)+P(2,2)+P(3,3))
         GO TO 20
   10 CONTINUE
         D=0
   20 CONTINUE
C     (* COMPUTE AIM POINT ON TARGET X AXIS *)
         R(1)=-D+HDT(1)
         R(2)=0 +HDT(2)
         R(3)=0 +HDT(3)
         CALL MMLT(V4,AT,ESTATE,3,3,1)
         IF(V4(2)**2+V4(3)**2 GT 225 0  OR   NOT  VALID) R(1)=-D-20
         CALL MSUB(V5,R,HDC,3,1)
C     (* CONVERT COORDINATES AND SUBTRACT CURRENT POSITION *)
         CALL MTRN(TRNAT,AT,3)
         CALL MMLT(V1,TRNAT,V5,3,3,1)
         CALL MSUB(V2,V1,ESTATE,3,1)
         CALL MMLT(V3,ACV,V2,3,3,1)
      SLOP=0 25*AMIN1( 1*ABS(V4(1)), 25)
      GXH=V3(1)
      IF(GXH LT 8 ) GXH=GXH+SLOP
      GYH=V3(2)+SLOP
      GZH=V3(3)+SLOP
      GXL=GXH-2 0*SLOP
      GYL=V3(2)-SLOP
      GZL=V3(3)-SLOP
      DEDLIN=T+ 125*SQRT(V3(1)**2+V3(2)**2+V3(3)**2)
      RETURN
C
      END
```



```
C
      SUBROUTINE THRUST(F,ESTATE,GXL,GXH,GYL,GYH,GZL,GZH,TLIM,ACV,
     A   ATRATE,RPYERR)
C     (* 480 -  SELECT THRUSTERS *)
C
C     CALLS
C        CNTLAW,FIRTHR,SELECT
C     CALLED BY
C        DOCK
C
C     INPUT
C        ESTATE,GXL,    ,GZH,TLIM,ESTATE,ACV,ATRATE,RPYERR
C     OUTPUT
C        F
C
      REAL ACV(3,3)
C        MEASURED CHASE VEHICLE ATTITUDE IN PRIMARY REF FRAME
      REAL ATRATE(3)
C        MEASURED ATTITUDE RATES OF CHASE VEHICLE
      REAL E(14)
C        THRUSTER COMMAND ENTRIES FROM THRUSTER SELECT LOGIC
      REAL ESTATE(6)
C        STATE ESTIMATE
      REAL F(14)
C        FORCES FROM THRUSTERS AFTER SELECTION
      REAL F1(14)
C        LIST OF MAXIMUM THRUSTER FORCES
      REAL GXL,GXH,GYL,GYH,GZL,GZH
C        LOWER AND UPPER LIMITS OF GOAL 'BOX'
      INTEGER I
C        DO LOOP INDEX
      REAL ROTCMD(3),XLTCMD(3)
C        ROTATIONAL AND TRANSLATIONAL COMMANDS FROM CONTROL LAW
      REAL RPYERR(3)
C        MEASURED ATTITUDE ERROR IN ROLL, PITCH, AND YAW (RADIANS)
      REAL TLIM
C        TIME LIMIT
C
      COMMON/VEHIC/DUMMY(99),F1
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* 481 - USE CONTROL LAW TO DETERMINE NEEDED THRUST *)
         CALL CNTLAW(ROTCMD,XLTCMD,ESTATE,ACV,TLIM,GXL,GXH,GYL,GYH,GZL,
     A      GZH,ATRATE,RPYERR)
C     (* 482 - SELECT THRUSTER SET TO GIVE NEEDED THRUST *)
         CALL SELECT(ROTCMD,XLTCMD,E)
         DO 10 I=1,14
            F(I)=E(I)*F1(I)
   10    CONTINUE
C     (* 483 - FIRE THRUSTERS *)
         CALL FIRTHR(E)
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE CNTLAW(ROTCMD, XLTCMD, ESTATE, ACV, TLIM, GXL, GXH, GYL, GYH,
     A   GZL, GZH, ATRATE, RPYERR)
C     (* 481 - CONTROL LAW *)
C
C     CALLS
C        MMLT, ACCEL
C     CALLED BY
C        THRUST
C
C     INPUT
C        ESTATE, ACV, TLIM, GXL,     , GZH, ATRATE, RPYERR
C     OUTPUT
C        ROTCMD, XLTCMD
C
      REAL ACV(3,3)
C        MEASURED CHASE VEHICLE ATTITUDE IN PRIMARY REFERENCE FRAME
      REAL ATRATE(3)
C        MEASURED CHASE VEHICLE ATTITUDE RATES
      REAL ESTATE(6)
C        STATE ESTIMATE
      REAL GXL, GXH, GYL, GYH, GZL, GZH
C        LOWER AND UPPER LIMITS OF GOAL 'BOX'
      REAL ROTCMD(3), XLTCMD(3)
C        ROTATIONAL AND TRANSLATIONAL COMMANDS FROM CONTROL LAW
      REAL RPYERR(3)
C        MEASURED ATTITUDE ERROR IN ROLL, PITCH, AND YAW
      REAL TLIM
C        TIME LIMIT
      REAL VBOD(3)
C        VELOCITY IN CHASE VEHICLE BODY COORDINATE SYSTEM
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* CONVERT VELOCITY TO BODY COORD  SYS  *)
         CALL MMLT(VBOD, ACV, ESTATE(4), 3, 3, 1)
C     (* 481 1 - FIND IDEAL ACCELERATIONS *)
         XLTCMD(1)=ACCEL(VBOD(1), GXL, GXH, TLIM, 1 733333, 03, 1)
         XLTCMD(2)=ACCEL(VBOD(2), GYL, GYH, TLIM, 1 733333, 03, 1)
         XLTCMD(3)=ACCEL(VBOD(3), GZL, GZH, TLIM, 1 733333, 03, 1)
         ROTCMD(1)=ACCEL(ATRATE(1), (-RPYERR(1)- 02), (-RPYERR(1)+ 02),
     A      1 733333, 1 733333, 02, 074)
         ROTCMD(2)=ACCEL(ATRATE(2), -RPYERR(2)- 02, -RPYERR(2)+ 02,
     A      1 733333, 1 733333, 02, 074)
         ROTCMD(3)=ACCEL(ATRATE(3), -RPYERR(3)- 02, -RPYERR(3)+ 02, TLIM,
     A      1 733333, 02, 074)
      RETURN
C
      END
```



```
C
      FUNCTION ACCEL(XDOT, XMIN, XMAX, TLIM, DT, AMIN, AMAX)
C     (* 461 1 - ESTIMATE ACCELERATION FOR ONE AXIS *)
C
C     CALLS
C        <NONE>
C     CALLED BY
C        CNTLAW
C
C     INPUT
C        X, XDOT, XMIN, XMAX, TLIM, DT
C     OUTPUT
C        <FUNCTION VALUE ONLY>
C
      REAL AMIN, AMAX
C        MIN AND MAX RELATIVE THRUST
      REAL DT
C        TIME STEP BETWEEN DECISIONS
      REAL K1, K2, K3, K4
C        CONSTANTS BASED ON MIN, MAX POSSIBLE ACCELERATION
      REAL TLIM
C        TIME LIMIT - SECONDS UNTIL DEADLINE
      REAL XDOT
C        VELOCITY IN THIS AXIS
      REAL XMIN, XMAX
C        MIN, MAX ACCEPTABLE FINAL X VALUE
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
C
      K1= 5*AMAX*DT**2
      K2=AMAX*DT
      K3= 5/AMIN
      K4=AMIN* 5
      IF(XMAX GE O ) GO TO 50
        IF(XDOT LT O ) GO TO 10
          ACCEL=-1
          RETURN
 10     IF(XDOT*DT-K3*XDOT**2 GE XMIN) GO TO 20
          ACCEL=1
          RETURN
 20     IF(XDOT*DT-K1-K3*(K2-XDOT)**2 GE XMIN) GO TO 30
          ACCEL=0
          RETURN
 30     IF(XDOT*TLIM-K4*(TLIM-DT)**2 GT XMAX) GO TO 40
          ACCEL=0
          RETURN
 40     CONTINUE
          ACCEL=-1
          RETURN
 50   IF(XMIN LE O ) GO TO 100
        IF(XDOT GT O ) GO TO 60
          ACCEL=1
          RETURN
 60     IF(XDOT*DT+K3*XDOT**2 LE XMAX) GO TO 70
          ACCEL=-1
          RETURN
 70     IF(XDOT*DT+K1+K3*(K2+XDOT)**2 LE XMAX) GO TO 80
          ACCEL=0
          RETURN
 80     IF(XDOT*DT+K4*(TLIM-DT)**2 LT XMIN) GO TO 90
          ACCEL=0
          RETURN
 90     CONTINUE
          ACCEL=1
          RETURN
100   IF(XDOT LE O ) GO TO 120
        IF(XDOT*DT+K3*XDOT**2 LE XMAX) GO TO 110
          ACCEL=-1
          RETURN
110     CONTINUE
          ACCEL=0
          RETURN
120   IF(XDOT*DT-K3*XDOT**2 GE XMIN) GO TO 130
        ACCEL=1
        RETURN
130   CONTINUE
        ACCEL=0
        RETURN
C
      END
```



```
C
      SUBROUTINE SELECT(ROTCMD, XLTCMD, E)
C     (* 4B2 - FROM CONTROL LAW OUTPUTS SELECT WHICH THRUSTERS TO FIRE *)
C
C     CALLS
C        TABLE1, TABLE2, TABLE3
C     CALLED BY
C        THRUST
C
C     INPUT
C        ROTCMD, XLTCMD
C     OUTPUT
C        E
C
      REAL E(14)
C        THRUSTER COMMAND ENTRIES FROM THRUSTER SELECT LOGIC
      REAL ROTCMD(3), XLTCMD(3)
C        ROTATIONAL AND TRANSLATIONAL COMMANDS FROM CONTROL LAW
      INTEGER I,J
C        DO LOOP INDICES
      INTEGER INDEX1, INDEX2, INDEX3
C       INDICES FOR COMMAND TABLES
      INTEGER ROTCOD(3), TRLCOD(3)
C        ROTATION AND TRANSLATION CODES USED TO COMPUTE INDICES
C        FOR TABLES
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* DETERMINE CODE FOR THRUSTER ACTIVATION *)
        DO 40 I=1,3
           IF(XLTCMD(I)) 10,20,30
 10        TRLCOD(I)=1
           GO TO 40
 20        TRLCOD(I)=0
           GO TO 40
 30        TRLCOD(I)=2
 40     CONTINUE
        DO 80 I=1,3
           IF(ROTCMD(I)) 50,60,70
 50        ROTCOD(I)=1
           GO TO 80
 60        ROTCOD(I)=0
           GO TO 80
 70        ROTCOD(I)=2
 80     CONTINUE
C     (* ZERO OUT THRUSTER COMMANDS *)
        DO 90 J=1,14
           E(J)=0 0
 90     CONTINUE
C     (* DETERMINE WHICH TABLE TO CONSULT FOR THRUSTER COMMANDS *)
        INDEX1=TRLCOD(1)+3*ROTCOD(2)+9*ROTCOD(3)
        INDEX2=TRLCOD(2)+3*ROTCOD(1)+9*ROTCOD(3)
        INDEX3=TRLCOD(3)+3*ROTCOD(1)+9*ROTCOD(2)
C     (* 4B2 1 THRU 4B2 3 - SELECT THRUSTER SET *)
        IF(INDEX1 EQ 0) GO TO 100
           CALL TABLE1(INDEX1,E)
100     CONTINUE
        IF(INDEX2 EQ 0) GO TO 110
           CALL TABLE2(INDEX2,E)
110     CONTINUE
        IF(INDEX3 EQ 0) GO TO 120
           CALL TABLE3(INDEX3,E)
120     CONTINUE
      RETURN
C
      END
```

ORIGINAL PAGE IS OF POOR QUALITY

```
C
      SUBROUTINE TABLE1(INDEX1,E)
C     (* 432.1 -  FIND COMINATION OF TABLE ENTRIES THAT SATISFY THRUSTER
C        SELECTION REQUIREMENTS *)
C
C     CALLS
C        <NONE>
C     CALLED BY
C        SELECT
C
C     INPUT
C        INDEX1
C     OUTPUT
C        E
C
      REAL E(14)
C        THRUSTER COMMAND ENTRIES FROM THRUSTER SELECT LOGIC
      INTEGER INDEX1
C        THRUSTER TABLE INDEX
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* DO SELECTION BASED ON INDEX *)
      GOTO(10,20,30,40,50,60,70,80,90,100,110,120,130,140,150,160,170,
     A   180,190,200,210,220,230,240,250,260),INDEX1
   10 E(1)=1.0
      GO TO 300
   20 E(14)=1.0
      GO TO 300
   30 E(10)=1.0
      E(13)=1.0
      GO TO 300
   40 E(1)=1.0
      E(10)=1.0
      E(13)=1.0
      GO TO 300
   50 E(10)=1.0
      E(13)=1.0
      E(14)=1.0
      GO TO 300
   60 E(11)=1.0
      E(12)=1.0
      GO TU 300
   70 E(1)=1.0
      E(11)=1.0
      E(12)=1.0
      GO TO 300
   80 E(11)=1.0
      E(12)=1.0
      E(14)=1.0
      GO TO 300
   90 E(2)=1.0
      E(4)=1.0
      GO TO 300
  100 E(1)=1.0
      E(2)=1.0
      E(4)=1.0
      GO TO 300
  110 E(2)=1.0
      E(4)=1.0
      E(14)=1.0
      GO TO 300
  120 E(2)=1.0
      E(4)=1.0
      E(10)=1.0
      E(13)=1.0
      GO TO 300
  130 E(1)=1.0
      E(2)=1.0
      E(4)=1.0
      GO TO 300
  140 E(2)=1.0
      E(4)=1.0
      E(10)=1.0
      E(13)=1.0
      E(14)=1.0
      GO TO 300
```

APPENDIX B -- RAINBOW VERSION OF SIMULATION PROGRAM
PAGE 45

```
  150 E(2)=1.0
      E(4)=1.0
      E(11)=1.0
      E(12)=1.0
      GO TO 300
  160 E(1)=1.0
      E(2)=1.0
      E(4)=1.0
      E(11)=1.0
      E(12)=1.0
      GO TO 300
  170 E(2)=1.0
      E(4)=1.0
      E(11)=1.0
      E(12)=1.0
      E(14)=1.0
      GO TO 300
  180 E(3)=1.0
      E(5)=1.0
      GO TO 300
  190 E(1)=1.0
      E(3)=1.0
      E(5)=1.0
      GO TO 300
  200 E(3)=1.0
      E(5)=1.0
      E(14)=1.0
      GO TO 300
  210 E(3)=1.0
      E(5)=1.0
      E(10)=1.0
      E(13)=1.0
      GO TO 300
  220 E(1)=1.0
      E(3)=1.0
      E(5)=1.0
      E(10)=1.0
      E(13)=1.0
      GO TO 300
  230 E(3)=1.0
      E(5)=1.0
      E(10)=1.0
      E(13)=1.0
      E(14)=1.0
      GO TO 300
  240 E(3)=1.0
      E(5)=1.0
      E(11)=1.0
      E(12)=1.0
      GO TO 300
  250 E(1)=1.0
      E(3)=1.0
      E(5)=1.0
      E(11)=1.0
      E(12)=1.0
      GO TO 300
  260 E(3)=1.0
      E(5)=1.0
      E(11)=1.0
      E(12)=1.0
      E(14)=1.0
  300 RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE TABLE2(INDEX2,E)
C     (* 482 2 -  FIND COMINATION OF TABLE ENTRIES THAT SATISFY THRUSTER
C        SELECTION REQUIREMENTS *)
C
C     CALLS
C        (NONE)
C     CALLED BY
C        SELECT
C
C     INPUT
C        INDEX2
C     OUTPUT
C        E
C
      REAL E(14)
C        THRUSTER COMMAND ENTRIES FROM THRUSTER SELECT LOGIC
      INTEGER INDEX2
C        THRUSTER TABLE INDEX
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* DO SELECTION BASED ON INDEX *)
      GOTO(10,20,30,40,50,60,70,80,90,100,110,120,130,140,150,160,170,
     A     180,190,200,210,220,230,240,250,260),INDEX2
   10 E(4)=1.0
      E(5)=1.0
      GO TO 300
   20 E(2)=1.0
      E(3)=1.0
      GO TO 300
   30 E(7)=1.0
      E(8)=1.0
      GO TO 300
   40 E(4)=1.0
      E(5)=1.0
      E(7)=1.0
      E(8)=1.0
      GO TO 300
   50 E(2)=1.0
      E(3)=1.0
      E(7)=1.0
      E(8)=1.0
      GO TO 300
   60 E(6)=1.0
      E(9)=1.0
      GO TO 300
   70 E(4)=1.0
      E(5)=1.0
      E(6)=1.0
      E(9)=1.0
      GO TO 300
   80 E(2)=1.0
      E(3)=1.0
      E(6)=1.0
      E(9)=1.0
      GO TO 300
   90 E(2)=1.0
      E(4)=1.0
      GO TO 300
  100 E(2)=1.0
      E(4)=1.0
      E(6)=1.0
      E(8)=1.0
      GO TO 300
  110 E(2)=1.0
      E(4)=1.0
      E(7)=1.0
      E(9)=1.0
      GO TO 300
  120 E(2)=1.0
      E(4)=1.0
      E(7)=1.0
      E(8)=1.0
      GO TO 300
  130 E([illegible])=1.0
      E([illegible])=1.0
```



```
      GO TO 300
  140 E(2)=1.0
      E(7)=1.0
      GO TO 300
  150 E(2)=1.0
      E(4)=1.0
      E(6)=1.0
      E(9)=1.0
      GO TO 300
  160 E(4)=1.0
      E(6)=1.0
      GO TO 300
  170 E(2)=1.0
      E(9)=1.0
      GO TO 300
  180 E(3)=1.0
      E(5)=1.0
      GO TO 300
  190 E(3)=1.0
      E(5)=1.0
      E(6)=1.0
      E(8)=1.0
      GO TO 300
  200 E(3)=1.0
      E(5)=1.0
      E(7)=1.0
      E(9)=1.0
      GO TO 300
  210 E(3)=1.0
      E(5)=1.0
      E(7)=1.0
      E(8)=1.0
      GO TO 300
  220 E(5)=1.0
      E(8)=1.0
      GO TO 300
  230 E(3)=1.0
      E(7)=1.0
      GO TO 300
  240 E(3)=1.0
      E(5)=1.0
      E(6)=1.0
      E(9)=1.0
      GO TO 300
  250 E(5)=1.0
      E(6)=1.0
      GO TO 300
  260 E(3)=1.0
      E(9)=1.0
  300 RETURN
C
      END
```

```
C
      SUBROUTINE TABLE3(INDEX3,E)
C     (* 4B2 3 -  FIND COMINATION OF TABLE ENTRIES THAT SATISFY THRUSTER
C        SELECTION REQUIREMENTS *)
C
C     CALLS
C        (NONE)
C     CALLED BY
C        SELECT
C
C     INPUT
C        INDEX3
C     OUTPUT
C        E
C
      REAL E(14)
C        COMMAND ENTRIES FROM THRUSTER SELECT LOGIC
      INTEGER INDEX3
C        THRUSTER TABLE INDEX
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* DO SELECTION BASED ON INDEX *)
      GOTO(10,20,30,40,50,60,70,80,90,100,110,120,130,140,150,160,170,
     A   180,190,200,210,220,230,240,250,260), INDEX3
   10 E(10)=1.0
      E(11)=1.0
      GO TO 300
   20 E(12)=1.0
      E(13)=1.0
      GO TO 300
   30 E(7)=1.0
      E(8)=1.0
      GO TO 300
   40 E(7)=1.0
      E(8)=1.0
      E(10)=1.0
      E(11)=1.0
      GO TO 300
   50 E(7)=1.0
      E(8)=1.0
      GO TO 300
   60 E(6)=1.0
      E(9)=1.0
      GO TO 300
   70 E(6)=1.0
      E(9)=1.0
      E(10)=1.0
      E(11)=1.0
      GO TO 300
   80 E(6)=1.0
      E(9)=1.0
      E(12)=1.0
      E(13)=1.0
      GO TO 300
   90 E(10)=1.0
      E(13)=1.0
      GO TO 300
  100 E(10)=1.0
      GO TO 300
  110 E(13)=1.0
      GO TO 300
  120 E(7)=1.0
      E(8)=1.0
      E(10)=1
      E(13)=1.0
      GO TO 300
  130 E(7)=1.0
      E(8)=1.0
      E(10)=1.0
      GO TO 300
  140 E(7)=1.0
      E(8)=1.0
      E(13)=1.0
      GO TO 300
  150 E(6)=1.0
      E(9)=1.0
```



```
      E(10)=1.0
      E(13)=1.0
      GO TO 300
  160 E(6)=1.0
      E(9)=1.0
      E(10)=1.0
      GO TO 300
  170 E(6)=1.0
      E(9)=1.0
      E(13)=1.0
      GO TO 300
  180 E(11)=1.0
      E(12)=1.0
      GO TO 300
  190 E(11)=1.0
      GO TO 300
  200 E(12)=1.0
      GO TO 300
  210 E(7)=1.0
      E(8)=1.0
      E(11)=1.0
      E(12)=1.0
      GO TO 300
  220 E(7)=1.0
      E(8)=1.0
      E(11)=1.0
      GO TO 300
  230 E(7)=1.0
      E(8)=1.0
      E(12)=1.0
      GO TO 300
  240 E(6)=1.0
      E(9)=1.0
      E(11)=1.0
      E(12)=1.0
      GO TO 300
  250 E(6)=1.0
      E(9)=1.0
      E(12)=1.0
      GO TO 300
  260 E(6)=1.0
      E(9)=1.0
      E(12)=1.0
  300 RETURN
C
      END
```

```
C
      SUBROUTINE FIRTHR(E)
C     (* 433 - PUT THRUSTER COMMANDS IN COMMAND PORTS *)
C
C     CALLS
C        <NONE>
C     CALLED BY
C        THRUST
C
C     INPUT
C        E
C     OUTPUT
C        <NONE>
C
      REAL E(14)
C        THRUSTER COMMAND ENTRIES FROM THRUSTER SELECT LOGIC
C
C     (* TEMPORARY DUMMY SUBROUTINE *)
      RETURN
C
      END
```



```
C
      SUBROUTINE IMU(STATE, ACV, ACVT, ATRATE)
C     (* 490 - SIMULATE IMU MEASUREMENT *)
C
C     CALLS
C        DIRMAT, MADD, MSCL, ANGVEC
C     CALLED BY
C        DOCK
C
C     INPUT
C        QO, STATE, FULDIS, INERCV, MEMPTY
C     OUTPUT
C        ACV, ACVT, ATRATE
C
      REAL ACV(3,3), ACVT(3,3)
C        DIRECTION COSINE MATRIX AND ITS TRANSPOSE FOR 'MEASURED' ATTITUDE
C        OF CHASE VEHICLE WITH RESPECT TO PRIMARY REFERENCE FRAME
      REAL ATRATE(3)
C        MEASURED ATTITUDE RATES ABOUT CHASE VEHICLE AXES
      REAL FULDIS(3,3)
C        TENSOR DESCRIBING FUEL DISTRIBUTION
      REAL INERCV(3,3)
C        TENSOR DESCRIBING INERTIA OF EMPTY CHASE VEHICLE
      REAL INERFL(3,3)
C        FUEL INERTIA TENSOR
      REAL INERTA(3,3)
C        INERTIA OF CHASE VEHICLE (LOADED)
      REAL MEMPTY
C        MASS OF EMPTY CHASE VEHICLE
      REAL QO(4), QP(4)
C        INITIAL CHASE VEHICLE ATTITUDE QUATERNION IN 'TRUTH' REFERENCE FRAME
C        AND CURRENT PRIMARY REFERENCE FRAME, RESPECTIVELY
      REAL STATE(14)
C        TRUE CHASE VEHICLE STATE
      REAL TACV(3,3), TACVT(3,3)
C        CHASE VEHICLE DIRECTION COSINE MATRIX WITH RESPECT TO TRUTH
C        COORDINATE SYSTEM AND INVERSE OF THIS MATRIX
C
      COMMON/REF/QO
      COMMON/MASPRP/FULDIS, INERCV, MEMPTY
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* SUBTRACT INITIAL ATTITUDE FROM CURRENT ATTITUDE TO GET CURRENT
C        ATTITUDE IN PRIMARY REF  FRAME *)
         QP(1)=QO(4)*STATE(10)+QO(3)*STATE(11)-QO(2)*STATE(12)-
     A      QO(1)*STATE(13)
         QP(2)=-QO(3)*STATE(10)+QO(4)*STATE(11)+QO(1)*STATE(12)-
     A      QO(2)*STATE(13)
         QP(3)=QO(2)*STATE(10)-QO(1)*STATE(11)+QO(4)*STATE(12)-
     A      QO(3)*STATE(13)
         QP(4)=QO(1)*STATE(10)+QO(2)*STATE(11)+QO(3)*STATE(12)+
     A      QO(4)*STATE(13)
C     (* 901 - CONVERT QUATERNIONS TO DIRECTION COSINE MATRICES *)
         CALL DIRMAT(QP, ACV, ACVT)
         CALL DIRMAT(STATE(10), TACV, TACVT)
C     (* DETERMINE INERTIA *)
         CALL MSCL(INERFL, FULDIS, 3, 3, STATE(14)-MEMPTY)
         CALL MADD(INERTA, INERCV, INERFL, 3, 3)
C     (* 904 - FIND ANGULAR VELOCITY VECTOR *)
         CALL ANGVEC(INERTA, STATE(7), ATRATE, TACV)
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE RPY(U,V,RPYERR,ACV,AT)
C     (* 4A0 - CALCULATE ROLL, PITCH, AND YAW ERRORS *)
C
C     CALLS
C        ATAN2,MTRN,MMLT
C     CALLED BY
C        DOCK
C
C     INPUT
C        U,V,ACV,AT
C     OUTPUT
C        RPYERR
C
      REAL ACV(3,3),AT(3,3),AREL(3,3)
C        DIRECTION COSINE MATRICES OF THE CHASE VEHICLE AND TARGET WITH
C        RESPECT TO THE PRIMARY REF  FRAME AND OF THE CHASE VEHICLE WITH
C        RESPECT TO THE TARGET    AREL = ACV * TRN( AT )
      REAL RPYERR(3)
C        ROLL, PITCH AND YAW ERRORS (RADIANS)
      REAL TRNAT(3,3)
C        TRANSPOSE OF AT
      REAL U(3),V(3)
C        BEACON IMAGE CENTROIDS FROM THE THREE TELEVISION CAMERAS
C
      COMMON/OPTSYS/DUMMY1(2),FOCLEN
C
C- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C
      (* CALCULATE PITCH ERROR *)
         RPYERR(2)=ATAN2(V(2),FOCLEN)
C     (* CALCULATE YAW ERROR *)
         RPYERR(3)=ATAN2(U(2),FOCLEN)
C     (* CALCULATE ROLL ERROR *)
         CALL MTRN(TRNAT,AT,3)
         CALL MMLT(AREL,ACV,TRNAT,3,3,3)
         IF(ABS(AREL(2,3)**2+AREL(3,3)**2) LT  01) GO TO 10
            RPYERR(1)=ATAN2(AREL(2,3),AREL(3,3))
            GO TO 20
   10    CONTINUE
            RPYERR(1)=0
   20    CONTINUE
      RETURN
C
      END
```



```
C
      SUBROUTINE ESTRPY(ESTATE,ACVT,RPYERR)
C     (* 4B0 - ESTIMATE ATTITUDE ERROR FROM STATE ESTIMATE *)
C
C     CALLS
C        ATAN2,ASIN,MSCL
C     CALLED BY
C        DOCK
C
C     INPUT
C        ESTATE,ACVT
C     OUTPUT
C        RPYERR
C
      REAL ACVT(3,3)
C        TRANSPOSE OF CHASE VEHICLE DIRECTION COSINE MATRIX (WITH RESPECT
C        TO PRIMARY REFERENCE FRAME)
      REAL DOTPRD
C        DOT PRODUCT OF VECTORS DEFINING LINE TO TARGET AND VEHICLE X AXIS
      REAL ESTATE(6)
C        ESTIMATED STATE
      REAL PHI
C        EULER ERROR ANGLE
      REAL PRDMAG
C        MAGNITUDE OF CROSS PRODUCT OF VECTORS DEFINING LINE TO TARGET AND
C        VEHICLE X AXIS
      REAL Q(4)
C        ATTITUDE ERROR QUATERNION
      REAL R1(3),R2(3)
C        INTERMEDIATE RESULTS
      REAL RPYERR(3)
C        ERRORS IN ROLL, PITCH AND YAW, RESPECTIVELY (RADIANS)
C
C- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      R1(1)=ESTATE(2)*ACVT(3,1)-ESTATE(3)*ACVT(2,1)
      R1(2)=ESTATE(3)*ACVT(1,1)-ESTATE(1)*ACVT(3,1)
      R1(3)=ESTATE(1)*ACVT(2,1)-ESTATE(2)*ACVT(1,1)
      PRDMAG=SQRT(R1(1)**2+R1(2)**2+R1(3)**2)
      DOTPRD=-ESTATE(1)*ACVT(1,1)-ESTATE(2)*ACVT(2,1)-ESTATE(3)*
     A   ACVT(3,1)
      IF(PRDMAG GT 0 ) GO TO 30
         Q(1)=0
         Q(2)=0
         IF(DOTPRD GT 0 ) GO TO 10
            Q(3)=1.
            Q(4)=0 0
            GO TO 20
   10    CONTINUE
            Q(3)=0 0
            Q(4)=1.0
   20    CONTINUE
         GO TO 40
   30 CONTINUE
         CALL MSCL(R2,R1,3,1,1 /PRDMAG)
         PHI=ATAN2(PRDMAG,DOTPRD)
         CALL MSCL(Q,R2,3,1,SIN(PHI* 5))
         Q(4)=COS(PHI* 5)
   40 CONTINUE
      RPYERR(1)=0
      RPYERR(2)=-ASIN(2.*(Q(1)*Q(3)-Q(2)*Q(4)))
      IF(ABS(RPYERR(2)) GT 1 57) GO TO 50
         RPYERR(3)=ATAN2(2.*(Q(1)*Q(2)+Q(3)*Q(4)),Q(1)**2-Q(2)**2
     A      -Q(3)**2+Q(4)**2)
         GO TO 60
   50 CONTINUE
         RPYERR(3)=0.
   60 CONTINUE
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE GETPCH(STATE,PITCH,T,VALID)
C     (* 4C0 - COMPUTE TARGET PITCH ANGLE *)
C
C     CALLS
C        DIRMAT,MMLT,MADD,TRGATT,MTRN,MSUB
C      CALLED BY
C        DOCK
C
C     INPUT
C        STATE,T
C     OUTPUT
C        PITCH,VALID
C
      LOGICAL VALID
C        TRUE IF MEASUREMENT VALID
      REAL AC(3,3),TRNAC(3,3)
C        MEASURED DIRECTION COSINE MATRIX OF CHASE VEHICLE, AND TRANSPOSE
C        OF AC, RESPECTIVELY
      REAL AT(3,3), TRNAT(3,3)
C        MEASURED DIRECTION COSINE MATRIX OF TARGET SPACECRAFT AND TRANSPOSE
C        OF AT, RESPECTIVELY
      REAL HCC(3)
C        CENTER CAMERA POSITION IN CHASE VEHICLE COORDINATE SYSTEM
      REAL HT(3)
C        POSITION OF CENTER OF DOCKING AID IN TARGET SPACECRAFT BODY FRAME
      REAL PITCH1,PITCH
C        ANGULAR ERROR FROM -X/Y PLANE MEASURED IN THE PROJECTION
C        OF THE POSITION ONTO THE -X/Z PLANE AND
C          CORRUPTED VERSION OF PITCH1, RESPECTIVELY
      REAL PROD(3),SUM(3)
C        INTERMEDIATE RESULTS WITH NO PROBLEM-RELATED SIGNIFICANCE
      REAL PV(3)
C        POSITION OF THE CAMERA IN THE DOCKING AID REFERENCE FRAME (DOCKING
C        AID REFERENCE FRAME IS PARALLEL TO TARGET SPAACECRAFT REFERENCE FRAME)
      REAL STATE(14)
C        CHASE VEHICLE STATE VECTOR
      REAL T
C        ELAPSED TIME
      REAL YAW1,YAW
C        ANGULAR ERROR FROM THE -X/Z PLANE MEASURED IN THE PROJECTION OF THE
C        POSITION ONTO THE -X/Y PLANE AND CORRUPTED VERSION OF YAW1,RESPECTIVELY
C
       COMMON/HNGOFF/DUMMY(6),HT,DUMMY1(6),HCC
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C      (* 901 - COMPUTE A DIRECTION COSINE MATRIX FROM A QUATERNION *)
         CALL DIRMAT(STATE(10),AC,TRNAC)
C      (* COMPUTE HCC IN PRIMARY REFERENCE FRAME *)
         CALL MMLT(PROD,TRNAC,HCC,3,3,1)
       CALL MADD(SUM,STATE,PROD,3,1)
C      (* 905- COMPUTE TARGET ATTITUDE AS A FUNCTION OF TIME *)
         CALL TRGATT(TRNAT,T)
C      (* COMPUTE TARGET'S TRUE ATTITUDE DIRECTION COSINE MATRIX *)
         CALL MTRN(AT,TRNAT,3)
C      (* CONVERT SUM TO TARGET REFERENCE FRAME *)
         CALL MMLT(PROD,AT,SUM,3,3,1)
C      (* COMPUTE NET VECTOR FROM DOCKING AID TO CAMERA IN TARGET FRAME *)
         CALL MSUB(PV,PROD,HT,3,1)
C      (* TEST VALIDITY OF X COMPONENT OF CAMERA POSITION  *)
         IF (PV(1) GE 0.0) GO TO 10
         PITCH1=ATAN2(PV(3),-PV(1))
         YAW1=ATAN2(-PV(2),-PV(1))
         PITCH=FLOAT(MINO(MAXO(INT(100 *RANFN(2,PITCH1, 03*YAW1)+ 5),
     A           -100),100))* 01
         RETURN
C
   10  CONTINUE
C      (* FLAG FACT THAT CAMERA CANNOT SEE TARGET *)
          VALID= FALSE
       RETURN
C
          END
```



```
C
      SUBROUTINE GETYAW(STATE,YAW,T,VALID)
C     (* 4D0 - COMPUTE TARGET YAW ANGLE *)
C
C     CALLS
C        DIRMAT,MMLT,MADD,TRGATT,MTRN,MSUB
C      CALLED BY
C        DOCK
C
C     INPUT
C        STATE,T
C     OUTPUT
C        YAW,VALID
C
      LOGICAL VALID
C        TRUE IF MEASUREMENT VALID
      REAL AC(3,3),TRNAC(3,3)
C        MEASURED DIRECTION COSINE MATRIX OF CHASE VEHICLE, AND TRANSPOSE
C        OF AC, RESPECTIVELY
      REAL AT(3,3), TRNAT(3,3)
C        MEASURED DIRECTION COSINE MATRIX OF TARGET SPACECRAFT AND TRANSPOSE
C        OF AT, RESPECTIVELY
      REAL HCC(3)
C        CENTER CAMERA POSITION IN CHASE VEHICLE COORDINATE SYSTEM
      REAL HT(3)
C        POSITION OF CENTER OF DOCKING AID IN TARGET SPACECRAFT BODY FRAME
      REAL PITCH1
C        ANGULAR ERROR FROM -X/Y PLANE MEASURED IN THE PROJECTION OF THE
C        POSITION ONTO THE -X/Z PLANE
      REAL PROD(3),SUM(3)
C        INTERMEDIATE RESULTS WITH NO PROBLEM-RELATED SIGNIFICANCE
      REAL PV(3)
C        POSITION OF THE CAMERA IN THE DOCKING AID REFERENCE FRAME (DOCKING
C        AID REFERENCE FRAME IS PARALLEL TO TARGET SPACECRAFT REFERENCE FRAME)
      REAL STATE(14)
C        CHASE VEHICLE STATE VECTOR
      REAL T
C        ELAPSED TIME
      REAL YAW1,YAW
C        ANGULAR ERROR FROM THE -X/Z FLANE MEASURED IN TE PROJECTION OF THE
C        POSITION ONTO THE -X/Y PLANE AND CORRUPTED VERSION OF YAW RESPECITVELY
C
       COMMON/HNGOFF/DUMMY(6),HT,DUMMY1(6),HCC
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C      (* 901 - COMPUTE THE DIRECTION COSINE MATRIX OF THE CHASE VEHICLE *)
         CALL DIRMAT(STATE(10),AC,TRNAC)
C      (* COMPUTE HCC IN PRIMARY REFERENCE FRAME *)
         CALL MMLT(PROD,TRNAC,HCC,3,3,1)
       CALL MADD(SUM,STATE,PROD,3,1)
C      (* 905- COMPUTE TARGET ATTITUDE AS A FUNCTION OF TIME *)
         CALL TRGATT(TRNAT,T)
C      (* COMPUTE TARGET'S TRUE ATTITUDE DIRECTION COSINE MATRIX *)
         CALL MTRN(AT,TRNAT,3)
C      (* CONVERT SUM TO TARGET REFERENCE FRAME *)
         CALL MMLT(PROD,AT,SUM,3,3,1)
C      (* COMPUTE NET VECTOR FROM DOCKING AID TO CAMERA IN TARGET FRAME *)
         CALL MSUB(PV,PROD,HT,3,1)
C      (* TEST VALIDITY OF X COMPONENT OF CAMERA POSITION *)
         IF (PV(1) GE 0.0) GO TO 10
         PITCH1=ATAN2(PV(3),-PV(1))
         YAW1=ATAN2(-PV(2),-PV(1))
         YAW=FLOAT(MINO(MAXO(INT(100 *RANFN(2,YAW1, 03*PITCH1)+ 5),
     A          -100),100))* 01
       RETURN
C
   10  CONTINUE
C      (* FLAG FACT THAT CAMERA CANNOT SEE TARGET *)
          VALID= FALSE
       RETURN
C
          END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE DIRMAT(Q,A,TRNA)
C     (* 901 - COMPUTE A DIRECTION COSINE MATRIX FROM A QUATERNION *)
C
C     CALLS
C        MTRN
C     CALLED BY
C        IMU, STPRIM, GETPCH, GETYAW, CENTRD
C
C     INPUT
C        Q
C     OUTPUT
C        A, TRNA
C
      REAL A(3,3)
C        DIRECTION COSINE MATRIX FORMED FROM QUATERNION Q
      REAL TRNA(3,3)
C        TRANPOSE OF A
      REAL Q(4)
C        QUATERNION
C- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* COMPUTE ELEMENTS OF A *)
         A(1,1)=Q(1)**2-Q(2)**2-Q(3)**2+Q(4)**2
         A(1,2)=2 *(Q(1)*Q(2)+Q(3)*Q(4))
         A(1,3)=2 *(Q(1)*Q(3)-Q(2)*Q(4))
         A(2,1)=2 *(Q(1)*Q(2)-Q(3)*Q(4))
         A(2,2)=-Q(1)**2+Q(2)**2-Q(3)**2+Q(4)**2
         A(2,3)=2 *(Q(2)*Q(3)+Q(1)*Q(4))
         A(3,1)=2 *(Q(1)*Q(3)+Q(2)*Q(4))
         A(3,2)=2 *(Q(2)*Q(3)-Q(1)*Q(4))
         A(3,3)=-Q(1)**2-Q(2)**2+Q(3)**2+Q(4)**2
C     (* COMPUTE TRANSPOSE OF A *)
         CALL MTRN(TRNA,A,3)
      RETURN
C
      END
```



```
C
      SUBROUTINE STPRIM(STATE,F,INERTA,T,DSTATE)
C     (* 902 - DETERMINE TIME DERIVATIVE OF STATE VECTOR *)
C
C     CALLS:
C        ANGVEC, FORCE, LINACL, LPRIME, MAKROT, MPRIME, QPRIME, TORQUE, DIRMAT
C     CALLED BY:
C        COMPK1, COMPK2, COMPK3, COMPK4
C
C     INPUT:
C        F, T, STATE, INERTA
C     OUTPUT:
C        DSTATE
C
      REAL A(3,3)
C        'TRUE' DIRECTION COSINE MATRIX FOR CHASE VEHICLE
      REAL BODVEL(3)
C        ANGULAR VELOCITY ABOUT AXES OF CHASE VEHICLE
      REAL DSTATE(14)
C        TIME DERIVATIVE OF STATE VECTOR
      REAL F(14)
C        FORCES FROM CHASE VEHICLE THRUSTERS
      REAL INERTA(3,3)
C        INERTIA OF CASE VEHICLE (INCL. FUEL)
      REAL N(3)
C        TORQUE ON SPACECRAFT ABOUT ITS CENTER OF MASS
      REAL NTFORC(3)
C        NET FORCE FROM THRUSTER OPERATION
      REAL OMEGA(4,4)
C        ROTATION MATRIX OMEGA, TO BE APPLIED TO QUATERNION MATRIX 'Q'
C        TO FORM TIME DERIVATIVE OF Q (Q=ELEMENTS 10-13 OF STATE)
      REAL STATE(14)
C        CHASE VEHICLE STATE VECTOR
      REAL T
C        ELAPSED TIME
      REAL TRNA(3,3)
C        TRANSPOSE OF 'A'
C
C- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* 901 - COMPUTE A,TRNA FROM QUATERNION  *)
         CALL DIRMAT(STATE(10),A,TRNA)
C     (* 902.1 - DETERMINE NET FORCE FROM THRUSTERS IN 'TRUTH' COORD.SYS. *)
         CALL FORCE(F,NTFORC,TRNA)
C     (* 902.2 - COMPUTE TIME DERIVATIVE OF SPACECRAFT MASS *)
         CALL MPRIME(F,DSTATE(14))
C     (* 902.3 - COMPUTE LINEAR ACCELERATIONS *)
         CALL LINACL(T,STATE(14),NTFORC,STATE(1),DSTATE(4))
C     (* 904 - FIND ANGULAR VELOCITY VECTOR *)
         CALL ANGVEC(INERTA,STATE(7),BODVEL,A)
C     (* 902.4 - CALCULATE TORQUE ON SPACECRAFT FROM THRUSTER OPERATION *)
         CALL TORQUE(F,N,TRNA)
C     (* 902.5 - COMPUTE TIME DERIVATIVE OF ANGULAR MOMENTUM VECTOR *)
         CALL LPRIME(N,BODVEL,STATE(7),DSTATE(7),TRNA)
C     (* 902.6 - FORM ROTATION MATRIX OMEGA *)
         CALL MAKROT(BODVEL,OMEGA,A)
C     (* 902.7 - CALCULATE THE TIME DERIVATIVE OF QUATERNION *)
         CALL QPRIME(STATE(10),OMEGA,DSTATE(10))
C     (* DEFINE VELOCITY ELEMENTS OF DSTATE *)
         DSTATE(1)=STATE(4)
         DSTATE(2)=STATE(5)
         DSTATE(3)=STATE(6)
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE FORCE(F,NTFORC,TRNA)
C     (* 902 1 -   COMPUTE NET FORCE FROM THRUSTER OPERATION USING EQUATION
C              NET FORCE= A TRANSPOSE * AK2 * F *)
C
C     CALLS
C        MMLT
C     CALLED BY
C        STPRIM
C
C     INPUT
C        F,AK2,Q
C     OUTPUT
C        NTFORC
C
      REAL AK2(3,14)
C        CONSTANT MATRIX RELATING INDIVIDUAL THRUSTER OUTPUTS
C        TO NET THRUST IN SPACECRAFT REFERENCE FRAME
C        (THIS ACCOMODATES MISALIGNMENT OF THRUSTERS)
      REAL B(3)
C         INTERMEDIATE RESULTS
      REAL F(14)
C        FORCE VECTOR OF CHASE VEHICLE AFTER THRUSTER SELECTION
      REAL NTFORC(3)
C        NET FORCE FROM THRUSTER OPERATION IN TRUTH COORD  SYS
      REAL TRNA(3,3)
C        TRANSPOSE OF DIRECTION COSINE MATRIX A
C
      COMMON/VEHIC/DUMMY1(14),AK2,DUMMY2(57)
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* COMPUTE NET FORCE *)
         CALL MMLT(B,AK2,F,3,14,1)
         CALL MMLT(NTFORC,TRNA,B,3,3,1)
      RETURN
C
      END
```



```
C
      SUBROUTINE MPRIME(F,DMDT)
C     (* 902 2 -   COMPUTE TIME DERIVATIVE OF SPACECRAFT MASS *)
C
C     CALLS
C        MMLT
C     CALLED BY
C        STPRIM
C
C     INPUT
C        AK1,F
C     OUTPUT
C        DMDT
C
      REAL AK1(14)
C        CONSTANT VECTOR DERIVED FROM ENGINE SPECIFIC IMPULSE DATA
      REAL DMDT
C        TIME DERIVATIVE OF MASS
      REAL F(14)
C        FORCES FROM THRUSTERS AFTER SELECTION
C
      COMMON/VEHIC/AK1,DUMMY(99)
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* COMPUTE DMDT = AK1*F *)
         CALL MMLT(DMDT,AK1,F,1,14,1)
C     (* CHANGE SIGN   MASS LOSS IS NEGATIVE *)
         DMDT=-DMDT
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
C     SUBROUTINE -INACL(T,M,NTFORC,CURPOS,ACCEL)
C     (* 902 3 -  COMPUTE LINEAR ACCELERATIONS IN X, Y, AND Z DIRECTIONS *)
C     CALLS
C        SIN,COS
C     CALLED BY
C        STPRIM
C
C     INPUT
C        T,M,NTFORC,CURPOS
C     OUTPUT
C        ACCEL
C
      REAL ACCEL(3)
C        SPACECRAFT ACCELERATIONS
      REAL AGGX,AGGY,AGGZ
C        ACCELERATIONS DUE TO GRAVITY GRADIANT
      REAL CURPOS(3)
C        CURRENT CHASE VEHICLE POSITION VECTOR
      REAL M
C        MASS OF SPACECRAFT IS M
      REAL NTFORC(3)
C        NET FORCE FROM THRUSTER OPERATION
      REAL SWT,CWT,A,B
C        THESE ARE INTERMEDIATE RESULTS
      REAL T
C        ELAPSED TIME
      REAL WO
C        RELATIVE ANGULAR ACCELERATION IN RADS/SEC
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* PRECOMPUTE VALUES USED SEVERAL PLACES *)
        WO=1.1629E-3
        A=2.67671E-6
        B=WO*T
        SWT=SIN(B)
        CWT=COS(B)
C     (* COMPUTE GRAVITY GRADIENT ACCELERATION *)
        AGGX=A*SWT*(CURPOS(1)*SWT+CURPOS(3)*CWT)
        AGGY=0.0
        AGGZ=A*CWT*(CURPOS(1)*SWT+CURPOS(3)*CWT)
C     (* COMPUTE NET ACCELERATION (GRAVITY PLUS THRUSTERS) *)
        ACCEL(1)=AGGX+NTFORC(1)/M
        ACCEL(2)=AGGY+NTFORC(2)/M
        ACCEL(3)=AGGZ+NTFORC(3)/M
      RETURN
C
      END
```



```
C
C     SUBROUTINE TORQUE(F,N,TRNA)
C     (* 902.4 -  CALCULATE TORQUE (N) *)
C     CALLS:
C        MMLT,MADD,MSCL
C     CALLED BY:
C        STPRIM
C
C     INPUT:
C        AK3,F,TRNA
C     OUTPUT:
C        N
C
      REAL AK3(3,14)
C        MATRIX RELATING THRUSTER INPUTS TO TORQUES IN 'CV' REF. FRAME
      REAL F(14)
C        FORCES FROM THRUSTERS AFTER SELECTION
      REAL N(3)
C        TORQUE ON SPACECRAFT ABOUT ITS CENTER OF MASS IN TRUTH COORD. SYS.
      REAL N1(3)
C        TORQUE IN C.V. REFERENCE FRAME
      REAL TRNA(3,3)
C        TRANSPOSE OF DIRECTION COSINE MATRIX THAT GIVES C.V. ATTITUDE
      COMMON/VEHIC/DUMMY1(36),AK3,DUMMY2(15)
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* CALCULATE N=TRNA*AK3*F *)
        CALL MMLT(N1,AK3,F,3,14,1)
        CALL MMLT(N,TRNA,N1,3,3,1)
      RETURN
C
      END
```

```
C
      SUBROUTINE LPRIME(N,BODVEL,ANGMNT,DLDT,TRNA)
C     (* 902 5 -  COMPUTE TIME DERIVATIVE OF ANGULAR MOMENTUM *)
C
C     CALLS
C        MSUB,MMLT
C     CALLED BY
C        STPRIM
C
C     INPUT
C        N,ANGVEL,ANGMNT
C     OUTPUT
C        DLDT
C
      REAL ANGMNT(3)
C        ANGULAR MOMENTUM VECTOR IN 'TRUTH' COORDINATE SYSTEM
      REAL ANGVEL(3)
C        CHASE VEHICLE ANGULAR VELOCITY IN 'TRUTH' COORDINATE SYSTEM
      REAL BODVEL(3)
C        CHASE VEHICLE ANGULAR VELOCITY IN CV COORDINATE SYSTEM
      REAL B(3)
C        TEMPORARY STORAGE VECTOR
      REAL DLDT(3)
C        TIME DERIVATIVE OF ANGULAR MOMENTUM VECTOR
      REAL N(3)
C        TORQUE ON C V  ABOUT ITS CENTER OF MASS IN  TRUTH' COORDINATE SYSTEM
      REAL TRNA(3,3)
C        TRANSPOSE OF DIRECTION COSINE MATRIX
C
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* CONVERT ANGULAR VELOCITY TO 'TRUTH' COORDINATE SYSTEM  *)
         CALL MMLT(ANGVEL,TRNA,BODVEL,3,3,1)
C     (* COMPUTE B = CROSS PRODUCT OF ANGVEL AND ANGMNT *)
         B(1)= ANGVEL(2)*ANGMNT(3)-ANGVEL(3)*ANGMNT(2)
         B(2)= ANGVEL(3)*ANGMNT(1)-ANGVEL(1)*ANGMNT(3)
         B(3)= ANGVEL(1)*ANGMNT(2)-ANGVEL(2)*ANGMNT(1)
C     (* COMPUTE DLDT=N-B *)
         CALL MSUB(DLDT,N,B,3,1)
      RETURN
C
      END
```



```
C
      SUBROUTINE MAKROT(BODVEL,OMEGA,A)
C     (* 902 6 -  FORM ROTATION MATRIX OMEGA FROM ANGULAR VELOCITY VECTOR *)
C
C     CALLS
C        <NONE>
C     CALLED BY
C        STPRIM
C
C     INPUT
C        ANGVEL
C     OUTPUT
C        OMEGA
C
      REAL A(3,3)
C        DIRECTION COSINE MATRIX GIVING TRUE C V  ATTITUDE
      REAL BODVEL(3)
C        ANGULAR VELOCITY IN CV COORDINATE SYSTEM
      INTEGER I
C        DO LOOP INDEX
      REAL OMEGA(4,4)
C        ROTATION MATRIX OMEGA, TO BE APPLIED TO QUATERNION
C        TO FORM TIME DERIVATIVE OF QUATERNION
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      DO 10 I=1,4
         OMEGA(I,I)=0 0
   10    CONTINUE
      OMEGA(1,2)=BODVEL(3)
      OMEGA(1,3)=-BODVEL(2)
      OMEGA(1,4)=BODVEL(1)
      OMEGA(2,1)=-BODVEL(3)
      OMEGA(2,3)=BODVEL(1)
      OMEGA(2,4)=BODVEL(2)
      OMEGA(3,1)=BODVEL(2)
      OMEGA(3,2)=-BODVEL(1)
      OMEGA(3,4)=BODVEL(3)
      OMEGA(4,1)=-BODVEL(1)
      OMEGA(4,2)=-BODVEL(2)
      OMEGA(4,3)=-BODVEL(3)
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C     SUBROUTINE QPRIME(Q,OMEGA,QPRM)
C     (* 902 7 -  COMPUTE QPRM = TIME DERIVATIVE OF QUATERNION Q *)
C
C     CALLS
C        MMLT,MSCL
C     CALLED BY
C        STPRIM
C
C     INPUT
C        Q,OMEGA
C     OUTPUT
C        QPRM
C
      REAL OMEGA(4,4)
C        MATRIX FORMED FROM ANGULAR VELOCITY COMPONENTS
      REAL Q(4)
C        ATTITUDE QUATERNIION OF CHASE VEHICLE
      REAL QINT(4)
C        INTERMEDIATE VALUES, NO PROBLEM RELATED SIGNIFICANCE
      REAL QPRM(4)
C        TIME DERIVATIVE OF OF Q
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* COMPUTE TIME DERIVATIVE OF Q = 0 5*OMEGA*Q *)
         CALL MMLT(QINT,OMEGA,Q,4,4,1)
         CALL MSCL(QPRM,QINT,4,1,0 5)
      RETURN
C
      END
```



```
C
      FUNCTION SQR1(X)
C     (* 903 - RETURNS SQUARE ROOT BUT ALLOWS SLIGHTLY NEGATIVE ARGUMENT *)
C
C     CALLS
C        SQRT,AMAX1
C     CALLED BY
C        TOQUAT,SETGOL
C
C     INPUT
C        X
C     OUTPUT
C        SQR1 (FUNCTION VALUE ONLY)
C
      REAL X
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      SQR1=SQRT(AMAX1(X,0 ))
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE ANGVEC(INERTA,ANGMNT,BODVEL,A)
C     (* 904 -  COMPUTE TRUE ANGULAR VELOCITY VECTOR ANGVEL
C        IN CV BODY COORDINATE SYSTEM  *)
C
C     CALLS
C        MINV,MMLT
C     CALLED BY
C        STPRIM,IMU
C
C     INPUT
C        INERTA,ANGMNT,TERMNL,A
C     OUTPUT
C        BODVEL
C
      REAL A(3,3)
C        DIRECTION COSINE MATRIX (CHASE VEHICLE ATTITUDE WRT 'TRUTH' AXES)
      REAL ANGMNT(3)
C        ANGULAR MOMENTUM VECTOR
      REAL BODVEL(3)
C        CHASE VEHICLE ANGULAR VELOCITY IN CV COORDINATE SYSTEM
      REAL INERTA(3,3)
C        INERTIA OF CHASE VEHICLE (LOADED)
      REAL INVIN(3,3)
C        INVERSE OF MOMENT OF INERTIA TENSOR
      REAL TEMP1(3)
C        INTERMEDIATE RESULTS WITH NO PROBLEM-RELATED SIGNIFICANCE
      REAL WORK(4,6)
C        AN INTERMEDIATE WORKSPACE
      INTEGER IERR
C        ERROR FLAG (=0 IF NO ERROR)
      INTEGER TERMNL
C        FORTRAN LOGICAL UNIT NUMBER OF USER TERMINAL
C
      COMMON/SIMUL/TERMNL,IDUMMY
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* FORM INVERSE OF INERTIA TENSOR *)
         CALL MINV(INVIN,INERTA,3,WORK,4,6,IERR)
      IF(IERR NE 0) GO TO 10
C     (* CALCULATE BODVEL=INVIN*A*ANGMNT *)
         CALL MMLT(TEMP1,A,ANGMNT,3,3,1)
         CALL MMLT(BODVEL,INVIN,TEMP1,3,3,1)
      RETURN
C
   10 CONTINUE
C     (* REPORT ERROR CONDITION AND HALT *)
         WRITE(TERMNL,901)
         STOP
C
  901 FORMAT(' MATRIX INVERSION FAILURE IN SUBROUTINE ANGVEC')
C
      END
```



```
C
      SUBROUTINE TRGATT(TRNAT,T)
C     (* 905 - COMPUTE TARGET ATTITUDE AS A FUNCTION OF TIME *)
C
C     CALLS
C        COS,SIN,MMLT
C     CALLED BY
C        CENTRD,GETYAW,GETPCH,DOCK
C
C     INPUT
C        T
C     OUTPUT
C        TRNAT
C
      REAL CHGATT(3,3)
C        TRANSPOSE OF DIRECTION COSINE MATRIX REPRESENTING TRUE TARGET
C        ATTITUDE CHANGE
      INTEGER ITUMBL
C        TARGET TUMBLE AXIS--VALUE OF 1, 2, OR 3 REPRESENTS YAW,
C        PITCH, OR ROLL TUMBLE
      REAL PHI
C        ROTATION ANGLE
      REAL T
C        ELAPSED TIME
      REAL TRNAT(3,3)
C        TRANSPOSE OF TARGET'S TRUE-ATTITUDE DIRECTION COSINE MATRIX
      REAL TRNATO(3,3)
C        INITIAL VALUE OF TRNAT
      REAL TUMRAT
C        TUMBLE RATE (IN RAD/SEC)
C
      COMMON/ATINFO/TRNATO,ITUMBL,TUMRAT
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      PHI=TUMRAT*T
C     (* SELECT APPROPRIATE SET OF FORMULAS FOR TARGET ATTITUDE *)
         GO TO (10,20,30), ITUMBL
   10    CONTINUE
C        (* COMPUTE ATTITUDE CHANGE RESULTING FROM YAW TUMBLE *)
            CHGATT(1,1)=COS(PHI)
            CHGATT(2,1)=SIN(PHI)
            CHGATT(3,1)=0 0
            CHGATT(1,2)=-SIN(PHI)
            CHGATT(2,2)=COS(PHI)
            CHGATT(3,2)=0 0
            CHGATT(1,3)=0 0
            CHGATT(2,3)=0 0
            CHGATT(3,3)=1 0
            GO TO 40
   20    CONTINUE
C        (* COMPUTE ATTITUDE CHANGE RESULTING FROM PITCH TUMBLE *)
            CHGATT(1,1)=COS(PHI)
            CHGATT(2,1)=0 0
            CHGATT(3,1)=-SIN(PHI)
            CHGATT(1,2)=0 0
            CHGATT(2,2)=1 0
            CHGATT(3,2)=0 0
            CHGATT(1,3)=SIN(PHI)
            CHGATT(2,3)=0 0
            CHGATT(3,3)=COS(PHI)
            GO TO 40
   30    CONTINUE
C        (* COMPUTE ATTITUDE CHANGE RESULTING FROM A ROLL TUMBLE *)
            CHGATT(1,1)=1 0
            CHGATT(2,1)=0 0
            CHGATT(3,1)=0 0
            CHGATT(1,2)=0 0
            CHGATT(2,2)=COS(PHI)
            CHGATT(3,2)=SIN(PHI)
            CHGATT(1,3)=0 0
            CHGATT(2,3)=-SIN(PHI)
            CHGATT(3,3)=COS(PHI)
   40 CONTINUE
C     (* COMPUTE TPUE TARGET ATTITUDE *)
         CALL MMLT(TRNAT,TRNATO,CHGATT,3,3,3)
      RETURN
```

```
C
      END
```



```
C
      SUBROUTINE XPRD(XP,X,Y)
C     (* 906 - COMPUTE THE CROSS-PRODUCT OF TWO VECTORS *)
C
C     CALLS
C        <NONE>
C     CALLED BY
C        QUATRN
C
C     INPUTS
C        X,Y
C     OUTPUTS
C        XP
C
      REAL XP(3)
C        (* CROSS PRODUCT OF X CROSSED WITH Y *)
      REAL X(3),Y(3)
C        (* VECTORS FOR WHICH CROSS PRODUCT IS TO BE DETERMINED *)
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* COMPUTE CROSS PRODUCT *)
         XP(1)=X(2)*Y(3)-X(3)*Y(2)
         XP(2)=X(3)*Y(1)-X(1)*Y(3)
         XP(3)=X(1)*Y(2)-X(2)*Y(1)
      RETURN
C
      END
```

```
      FUNCTION DPRD(X,Y,IDIM)
      (* 907 - COMPUTE THE DOT PRODUCT OF TWO VECTORS *)
C
C     CALLS:
C        <NONE>
C     CALLED BY:
C        QUATRN
C
C     INPUTS:
C        X,Y,IDIM
C     OUTPUTS:
C        DPRD
C
      REAL DPRD
C        (* DOT PRODUCT OF VECTORS X AND Y *)
      REAL X(3),Y(3)
C        (* VECTORS FOR WHICH DOT PRODUCT IS TO BE DETERMINED *)
      INTEGER IDIM,I
C        (* DIMENSION OF VECTORS X AND Y, AND DO LOOP INDEX, RESPECTIVELY *)
      REAL SUM
C        (* INTERMEDIATE RESULT WITH NO SIMPLE INTERPRETATION *)
C- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* COMPUTE DOT PRODUCT *)
      SUM=0.0
      DO 10 I=1,IDIM
        SUM=SUM+X(I)*Y(I)
   10   CONTINUE
      DPRD=SUM
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

Appendix C
Computer Program to Simulate System That Uses Ring of Lights

Appendix C Computer Program to Simulate System That Uses Ring of Lights

APPENDIX C--COMPUTER PROGRAM TO SIMULATE SYSTEM THAT USES RING OF LIGHTS

The program listing in this appendix is provided to document the simulation methods used in analyzing the "ring-of-lights" video guidance system. The discussion presented in Appendix A is equally applicable to this program. The text of the terminal session, which is presented in Appendix A, is not repeated here because the user commands and dialog are essentially identical to those in Appendix A.

```
C.            TRUE STATE
C
C                STATE( 1)  X POSITION
C                STATE( 2)  Y POSITION
C                STATE( 3)  Z POSITION
C                STATE( 4)  X VELOCITY
C                STATE( 5)  Y VELOCITY
C                STATE( 6)  Z VELOCITY
C                STATE( 7)  ANGULAR MOMENTUM ABOUT X
C                STATE( 8)  ANGULAR MOMENTUM ABOUT Y
C                STATE( 9)  ANGULAR MOMENTUM ABOUT Z
C                STATE(10) Q1 \
C                STATE(11) Q2  \
C                STATE(12) Q3   > ATTITUDE W.R.T. 'TRUTH' AXES
C                STATE(13) Q4  /
C                STATE(14)  MASS, INCL FUEL
C
C                  (X,Y,Z) = 'TRUTH' AXES
C
C
C             STATE ESTIMATE
C
C                ESTATE(1)  X' POSITION
C                ESTATE(2)  Y' POSITION
C                ESTATE(3)  Z' POSITION
C                ESTATE(4)  X' VELOCITY
C                ESTATE(5)  Y' VELOCITY
C                ESTATE(6)  Z' VELOCITY
C
C                  (X',Y',Z') = 'PRIMARY REF. FRAME' AXES
```

APPENDIX C -- RING-OF-LIGHTS VERSION OF SIMULATION PROGRAM


```
C
C          ROUTINE   PAGE ROUTINE-NBR
C
C          <MAIN>      1     000
C          ACCEL      42     481  1
C          ANGVEC     65     904
C          ATITUD     26     440
C          CNTLAW     41     481
C          COMPG      31     451
C          COMPK1     18     421
C          COMPK2     19     422
C          COMPK3     20     423
C          COMPK4     21     424
C          DIRMAT     55     901
C          DOCK       10     400
C          ELIP       14     411
C          EPAR       16     411  2
C          ESTRPD     54     480
C          ESTCOV     32     452
C          FINDCV     27     441
C          FIRTHR     51     483
C          FORCE      57     902  1
C          IMU        52     490
C          INCORP     30     450
C          INIPAR      5     200
C          INISIM      9     300
C          KALGAN     33     453
C          LINACL     59     902  3
C          LOCATE     12     410
C          LPRIME     61     902  5
C          MAKROT     62     902  6
C          MCALC      15     411  1
C          MPRIME     53     902  2
C          OPEN        4     100
C          POINT      22     425
C          PROPLS     36     460
C          PROPIR     17     420
C          QPRIME     63     902  7
C          QUATRN     28     442
C          RPY        53     4A0
C          SELEC1     23     430
C          SELECT     44     482
C          SETGOL     38     470
C          SQR1       64     903
C          STPRIM     56     902
C          TABLE1     45     482  1
C          TABLE2     47     482  2
C          TABLE3     49     482  3
C          THRUST     40     480
C          TOQUAT     25     431  1
C          TORQUE     60     902  4
C          TRGATT     66     905
C          TSTATT     24     431
C          UPDCOV     35     455
C          UPDSTA     34     454
C
```

APPENDIX C -- RING-OF-LIGHTS VERSION OF SIMULATION PROGRAM


```
C
C     PROGRAM MSFC DSIM3 - DOCKING SIMULATION PROGRAM, RING-OF-LIGHTS
C
C     CALLS
C        CLOS$A, DOCK, EXIT, INIPAR, INISIM, OPEN, YSNO$A
C
C
      REAL AOLD(3,3,2)
C        INITIAL VALUES OF TARGET DIRECTION COSINE MATRIX ESTIMATE
      LOGICAL CLOS$A
C        LIBRARY FUNCTION TO CLOSE FILES (FUNCTION VALUE INDICATES SUCCESS)
      REAL ESTATE(6,2)
C        ESTIMATE OF STATE VECTOR
      REAL F(14)
C        FORCES FROM THRUSTERS AFTER SELECTION
      INTEGER OUT
C        FORTRAN UNIT NUMBER OF OUTPUT FILE
      REAL P(6,6,2)
C        COVARIANCE OF STATE ESTIMATE
      REAL STATE(14)
C        CHASE VEHICLE STATE VECTOR
      REAL T
C        ELAPSED TIME
      INTEGER TERMNL
C        FORTRAN UNIT NUMBER FOR TERMINAL
      LOGICAL DOCKED
C        CODE SET TO '.TRUE.' IF THE CHASE VEHICLE IS WITHIN DOCKING RADIUS
      LOGICAL FILERR
C        ERROR CODE RETURNED FROM 'OPEN' SUBROUTINE
      LOGICAL YSNO$A
C        FUNCTION TO GET YES/NO ANSWER FROM USER AT TERMINAL
C
      COMMON/SIMUL/TERMNL,OUT
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      OUT=6
      TERMNL=1
C     (* LOOP UNTIL USER WANTS NO MORE SIMULATIONS *)
   10 CONTINUE
C        (* 100 - OPEN DATA FILE *)
            CALL OPEN(FILERR,TERMNL,OUT)
C        (* 200 - INITIALIZE PROBLEM PARAMETERS *)
            CALL INIPAR(T,STATE,P,ESTATE,F,AOLD)
C        (* 300 - INITIALIZE SIMULATOR *)
            CALL INISIM(STATE)
         DOCKED=.FALSE.
C        (* LOOP WHILE NOT DOCKED *)
   20    IF (DOCKED) GO TO 30
C           (* 400 - SIMULATE OBSERVATION & INTERVAL BETWEEN OBSERVATIONS *)
               CALL DOCK(P,ESTATE,STATE,T,DOCKED,F,AOLD)
            GO TO 20
   30    CONTINUE
C        (* CLOSE DATA FILE *)
            IF(.NOT.CLOS$A(OUT-4)) GO TO 40
   40    CONTINUE
C        (* ASK OPERATOR WHETHER HE WANTS ANOTHER SIMULATION *)
            IF(YSNO$A('DO YOU WANT ANOTHER SIMULATION',30,-1)) GO TO 10
      CALL EXIT
C
      END
```

ORIGINAL PAGE IS OF POOR QUALITY

```
C
      SUBROUTINE OPEN(FILERR, TERMNL, OUT)
C     (* 100 - OPEN FILES FOR OUTPUT  WARNING  HIGHLY INSTALLATION DEPENDENT *)
C
C     CALLS
C        DELE$A, EXST$A, TSRC$$
C     CALLED BY
C        <MAIN>
C
C     INPUT
C        IN, OUT
C     OUTPUT
C        FILERR
C
      INTEGER CHRPOS(2)
C        CHARACTER POSITION/COUNT CODES FOR SYSTEM ROUTINE TSRC$$
      INTEGER CODE
C        ERROR CODE RETURNED BY TSRC$$ (=0 IF NO ERROR)
      INTEGER I
C        DO LOOP INDEX
      INTEGER OUT, TERMNL
C        FORTRAN LOGICAL UNIT NUMBER FOR OUTPUT
      INTEGER PATH(16)
C        ARRAY OF ASCII DEFINING PATH NAME
      INTEGER TERMNL
C        FORTRAN LOGICAL UNIT NUMBER OF USER TERMINAL
      INTEGER TYPE
C        TYPE CODE RETURNED BY TSRC$$ (UNUSED HERE)
      LOGICAL DELE$A
C        SYSTEM ROUTINE TO DELETE A FILE (RETURNS TRUE IF SUCCESSFUL)
      LOGICAL EXST$A
C        SYSTEM ROUTINE TO CHECK EXISTENCE OF FILE (TRUE IF IT EXISTS)
      LOGICAL FILERR
C        ERROR CODE RETURNED TO CALLING ROUTINE
      LOGICAL YSNO$A
C        FUNCTION TO GET YES/NO ANSWER FROM USER AT TERMINAL
C
$INSERT SYSCOM>KEYS F
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      CHRPOS(2)=32
      TYPE=0
   10 CONTINUE
         WRITE(TERMNL,902)
         READ(TERMNL,901)(PATH(I),I=1,16)
      IF( NOT EXST$A(PATH,32)) GO TO 20
         IF( NOT YSNO$A('***FILE ALREADY EXISTS   OK TO OVERWRITE',
     A      40,-1)) GO TO 10
      IF( NOT DELE$A(PATH,32)) GO TO 20
   20 CONTINUE
      CHRPOS(1)=0
      CALL TSRC$$(K$WRIT+K$NSAM,PATH,OUT-4,CHRPOS,TYPE,CODE)
      IF(CODE NE 0) GO TO 30
      FILERR= FALSE
      RETURN
C
   30 FILERR= TRUE
      RETURN
C
  901 FORMAT(16A2)
  902 FORMAT('ENTER PATHNAME FOR OUTPUT    ')
C
      END
```


ORIGINAL PAGE IS
OF POOR QUALITY


```
C
      SUBROUTINE INIPAR(T,STATE,P,ESTATE,F,AOLD)
C     (* 200 - INITIALIZE PROBLEM PARAMETERS *)
C
C     CALLS
C        SQRT, RANFN
C     CALLED BY
C        <MAIN>
C
C     INPUT
C        <NONE>
C     OUTPUT
C        AK1, AK2, AK3, DOKRAD, ESTATE, FOCLEN, FULDIS, F1, INERCV, MEMPTY, P, TPHIH,
C        TPHIV, STATE, TERMNL, T, F, HC, HT, D, K, AOLD, GO
C
      REAL AK1(,4), AK2(3,14), AK3(3,14)
C        MATRICES SPECIFYING CHASE VEHICLE
C           AK1 GIVES FUEL USE PER NEWTON-SECOND OF THRUST FOR EACH THRUSTER
C           AK2 GIVES FORCE DIRECTION FOR EACH THRUSTER
C           AK3 GIVES TORQUE/NEWTON & TORQUE DIRECTION FOR EACH THRUSTER
      REAL AOLD(3,3,2)
C        INITIAL ESTIMATES OF TARGET ATTITUDE
      REAL D
C        DIAMATER OF CIRCLE OF LIGHTS ON DOCKING AID
      REAL DOKRAD
C        RADIUS FROM TARGET SPACECRAFT WITHIN WHICH THE SPACECRAFT ARE
C        CONSIDERED DOCKED
      REAL ESTATE(6,2)
C        ESTIMATE OF STATE VECTOR
      REAL FOCLEN
C        LENS FOCAL LENGTH IN METERS
      REAL FULDIS(3,3)
C        CHASE VEHICLE FUEL-DISTRIBUTION INERTIA TENSOR
      REAL F(14)
C        FORCES FROM THRUSTERS AFTER SELECTION
      REAL F1(14)
C        LIST OF MAXIMUM THRUST MAGNITUDES FROM EACH THRUSTER OF CHASE VEHICLE
      REAL HC(3), HT(3)
C        CAMERA POSITION IN CHASE VEHICLE BODY FRAME AND DOCKING AID POSITION
C        IN TARGET SPACECRAFT BODY FRAME, RESPECTIVELY
      REAL HDC(3), HDT(3)
C        DOCKING FIXTURE' POSITIONS IN THEIR RESPECTIVE SPACECRAFT COORDI-
C        NATE SYSTEMS
      INTEGER I, J
C        DO LOOP INDICES
      REAL INERCV(3,3)
C        INERTIA OF EMPTY CHASE VEHICLE
      INTEGER ITUMBL
C        TUMBLE AXIS, VALUE OF 1, 2, OR 3 INDICATES YAW, PITCH, OR ROLL
C        TUMBLE, RESPECTIVELY
      REAL K
C        CONSTANT RELATING MOMENTS OF LIGHT PATTERN TO ELLIPSE SEMIMAJOR AXIS
C        LENGTH
      REAL MEMPTY
C        MASS OF CHASE VEHICLE WITHOUT FUEL
      INTEGER OUT
C        FORTRAN UNIT NUMBER FOR OUTPUT FILE
      REAL P(6,6,2)
C        COVARIANCE OF STATE ESTIMATE
      REAL PHI, PSI, THETA
C        ANGLES SPECIFYING INITIAL TARGET ATTITUDE FOR A YAW-PITCH-ROLL
C        SEQUENCE
      REAL GO(4)
C        INITIAL ATTITUDE QUATERNION OF CHASE VEHICLE
      REAL TPHIH, TPHIV
C        TANGENTS OF HALF-FIELD-OF-VIEW ANGLES IN HORIZONTAL AND VERTICAL
C        DIRECTIONS FOR CAMERA SCANNING, RESPECTIVELY
      REAL STATE(14)
C        CHASE VEHICLE STATE VECTOR
      INTEGER TERMNL
C        FORTRAN LOGICAL UNIT NUMBER OF USER TERMINAL
      REAL T
C        ELAPSED TIME
      REAL TRNATO(3,3)
C        TRUE INITIAL TARGET DIRECTION COSINE MATRIX
```

```
C
      COMMON/MASPRP/FULDIS,INERCV,MEMPTY
      COMMON/ELIPSE/D,K
      COMMON/OPTSYS/TPHIH,TPHIV,FOCLEN
      COMMON/VEHIC/AK1,AK2,AK3,DOKRAD,F1
      COMMON/SIMUL/TERMNL,DU1
      COMMON/HNGOFF/HC,HT,HDC,HDT
      COMMON/REF/GO
      COMMON/ATINFO/TRNATO,ITUMBL,TUMRAT
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* INPUT INITIAL ATTITUDE ANGLES *)
      WRITE(TERMNL,901)
      READ(TERMNL,*) PHI,PSI,THETA
C     (* CONVERT ANGLES TO RADIANS *)
         PHI=PHI* 01745329
         PSI=PSI* 01745329
         THETA=THETA* 01745329
C     (* COMPUTE TRANSPOSE OF TRUE INITIAL TARGET ATTITUDE *)
         TRNATO(1,1)=COS(PSI)*COS(PHI)
         TRNATO(2,1)=COS(PSI)*SIN(PHI)
         TRNATO(3,1)=-SIN(PSI)
         TRNATO(1,2)=SIN(THETA)*SIN(PSI)*COS(PHI)-COS(THETA)*SIN(PHI)
         TRNATO(2,2)=SIN(PHI)*SIN(THETA)*SIN(PSI)+COS(THETA)*COS(PHI)
         TRNATO(3,2)=SIN(THETA)*COS(PSI)
         TRNATO(1,3)=COS(THETA)*SIN(PSI)*COS(PHI)+SIN(THETA)*SIN(PHI)
         TRNATO(2,3)=SIN(PHI)*COS(THETA)*SIN(PSI)-SIN(THETA)*COS(PHI)
         TRNATO(3,3)=COS(THETA)*COS(PSI)
C     (* INPUT TUMBLE AXIS *)
         WRITE(TERMNL,902)
    3    READ(TERMNL,*) ITUMBL
            IF(ITUMBL EQ 1 OR ITUMBL EQ 2 OR ITUMBL EQ 3) GO TO 5
            WRITE(TERMNL,904)
            GO TO 3
    5 CONTINUE
C     (* INPUT TUMBLE RATE *)
         WRITE(TERMNL,903)
         READ(TERMNL,*) TUMRAT
C     (* CONVERT TUMBLE RATE FROM DEGREES/HOUR TO RADIANS/SECOND
         TUMRAT=TUMRAT*4 84813E-6
      D=2 0
      K=SQRT(2 0)
      DOKRAD= 4
      FOCLEN= 05
      F1(1)=360
      F1(2)=80
      F1(3)=80
      F1(4)=80
      F1(5)=80
      F1(6)=20
      F1(7)=20
      F1(8)=20
      F1(9)=20
      F1(10)=80
      F1(11)=80
      F1(12)=80
      F1(13)=80
      F1(14)=360
      DO 10 I=1,3
         DO 10 J=1,3
            INERCV(I,J)=0
   10 CONTINUE
      INERCV(1,1)=1055
      INERCV(2,2)=1905
      INERCV(3,3)=1905
      TPHIH= 5
      TPHIV= 5
      T=0 0
      DO 20 I=1,14
         F(I)=0
   20 CONTINUE
      DO 30 I=1,3
         DO 30 J=1,3
            FULDIS(I,J)=0
   30 CONTINUE
      FULDIS(1,1)= 781
```



```
      FULDIS(2,2)= 706
      FULDIS(3,3)= 706
      DO 40 I=1,14
         AK1(I)=4 636E-4
   40 CONTINUE
      DO 50 I=1,3
         DO 50 J=1,14
            AK2(I,J)=0
            AK3(I,J)=0
   50 CONTINUE
      AK2(1,1)=-1
      AK2(2,2)=1
      AK2(2,3)=1
      AK2(2,4)=-1
      AK2(2,5)=-1
      AK2(2,6)=-1
      AK2(2,7)=1
      AK2(2,8)=-1
      AK2(2,9)=1
      AK2(3,10)=-1
      AK2(3,11)=-1
      AK2(3,12)=1
      AK2(3,13)=1
      AK2(1,14)=1
      AK3(3,2)=- 375
      AK3(3,3)= 375
      AK3(3,4)=- 375
      AK3(3,5)= 375
      AK3(1,6)=1 25
      AK3(1,7)=-1 25
      AK3(1,8)=-1 25
      AK3(1,9)=1 25
      AK3(2,10)=- 375
      AK3(2,11)= 375
      AK3(2,12)= 375
      AK3(2,13)=- 375
      MEMPTY=1800
      DO 60 I=1,6
         DO 60 J=1,6
            P(I,J,1)=0
            P(I,J,2)=0
   60 CONTINUE
      P(1,1,1)=400
      P(1,1,2)=400
      P(2,2,1)=563
      P(2,2,2)=563
      P(3,3,1)=563
      P(3,3,2)=563
      P(4,4,1)= 03
      P(4,4,2)= 03
      P(5,5,1)= 03
      P(5,5,2)= 03
      P(6,6,1)= 03
      P(6,6,2)= 03
      ESTATE(1,1)=-300
      ESTATE(2,1)=0
      ESTATE(3,1)=0
      ESTATE(4,1)= 1
      ESTATE(5,1)= 1
      ESTATE(6,1)= 1
      ESTATE(1,2)=-300
      ESTATE(2,2)=0
      ESTATE(3,2)=0
      ESTATE(4,2)= 1
      ESTATE(5,2)= 1
      ESTATE(6,2)= 1
      DO 70 I=1,6
         STATE(I)=RANFN(2,ESTATE(I,1),SQRT(P(I,I,1)))
   70 CONTINUE
      STATE(7)=0 0
      STATE(8)=0 0
      STATE(9)=0 0
      STATE(10)=0
      STATE(11)=0
      STATE(12)=0
```

```
      STATE(13)=1
      STATE(14)=3700
      GO(1)=STATE(10)
      GO(2)=STATE(11)
      GO(3)=STATE(12)
      GO(4)=STATE(13)
      HC(1)=3.49
      HC(2)=0.0
      HC(3)=0.0
      HT(1)=-2.0
      HT(2)=0.0
      HT(3)=0.0
      HDC(1)=3.49
      HDC(2)=0.0
      HDC(3)=0.0
      HDT(1)=-2.0
      HDT(2)=0.0
      HDT(3)=0.0
      DO 80 I=1,3
         DO 80 J=1,3
            AOLD(I,J,1)=0
            AOLD(I,J,2)=0
   80 CONTINUE
      DO 90 I=1,3
         DO 90 J=1,2
            AOLD(I,I,J)=1.
   90 CONTINUE
      RETURN
  901 FORMAT(// ENTER THREE ROTATION ANGLES TO SPECIFY INITIAL TARGET'/
     A      ' ATTITUDE FOR A YAW-PITCH-ROLL SEQUENCE---ANGLES MUST BE'/
     B      ' IN DEGREES')
  902 FORMAT(/' ENTER TUMBLE AXIS-- ENTER 1, 2, OR 3 FOR YAW, PITCH, '/
     A      ' OR ROLL, RESPECTIVELY')
  903 FORMAT(/' ENTER TUMBLE RATE IN DEGREES PER HOUR ')
  904 FORMAT(/' A VALUE OF 1, 2, OR 3 MUST BE ENTERED FOR TUMBLE AXIS')
C
      END
```



```
C
      SUBROUTINE INISIM(STATE)
C     (* 300 - INITIALIZE SIMULATOR *)
C
C     CALLS
C         <NONE>
C     CALLED BY
C         <MAIN>
C
C     INPUT
C         <NONE>
C     OUTPUT
C         <NONE>
C
C     (* DUMMY ROUTINE - ACCOMODATES PHYSICAL SIMULATION *)
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE DOCK(P,ESTATE,STATE,T,DOCKED,F,AOLD)
C     (* 400 - RUN SIMULATION FOR ONE MEASUREMENT INTERVAL *)
C
C     CALLS
C        SETCM , IMCORP, SELEC1, SORT, LOCATE, PROPTR, ATITUD, PROPES, THRUST,
C        IMU, RPY , MMLT, MSUB, MADD, MTRN
C     CALLED BY
C        (MAIN)
C
C     INPUT
C        P, ESTATE, DOKRAD, STATE, T, F, INERCV
C     OUTPUT
C        STATE, T, F, DOCKED, ESTATE, P
C
      REAL ACV(3,3), ACVT(3,3)
C        DIRECTION COSINE MATRIX DESCRIBING CURRENT CHASE VEHICLE ATTITUDE
C        (RELATIVE TO PRIMARY REFERENCE FRAME) AND TRANSPOSE OF ACV
      REAL AOLD(3,3,2), ANEW(3,3,2)
C        MEASURED DIRECTION COSINE MATRICES OF TARGET SPACECRAFT (TWO
C        INTERPRETATIONS) FROM LAST AND CURRENT MEASUREMENT
      REAL ATRATE(3)
C        MEASRED ATTITUDE RATES ABOUT CHASE VEHICLE AXES
      INTEGER CONTRL, TRACK
C        SUBSCRIPTS TO SELECT WHICH VERSIONS OF ANEW, P, AND ESTATE TO USE
C        FOR CONTROL AND WHICH ARE TO BE KEPT TRACK OF
      REAL CVPOS(3,2)
C        MEASURED CHASE VEHICLE POSITION IN PRIMARY REFERENCE FRAME
      REAL DEDLIN
C        TIME BY WHICH STRATEGY LOGIC WANTS CHASE VEHICLE IN GOAL 'BOX'
      REAL DOKRAD
C        RADIUS FROM TARGET AT WHICH CHASE VEHICLE IS CONSIDERED DOCKED
      REAL ESTATE(6,2)
C        ESTIMATE OF STATE VECTOR
      REAL F(14)
C        FORCES FROM CHASE VEHICLE THRUSTERS AFTER SELECTION
      REAL GXL, GXH, GYL, GYH, GZL, GZH
C        X, Y, Z UPPER, LOWER LIMITS OF GOAL 'BOX'
      REAL HDC(3), HDT(3)
C        DOCKING FIXTURES' POSITIONS IN RESPECTIVE SPACECRAFT COORDINATE
C        SYSTEMS
      INTEGER OUT
C        FORTRAN UNIT NUMBER FOR OUTPUT FILE
      REAL P(6,6,2)
C        COVARIANCE OF STATE ESTIMATE
      REAL R1(3), R2(3), R3(3), R4(3)
C        INTERMEDIATE RESULTS WITH NO SIMPLE PHYSICAL INTERPRETATION
      REAL RELPOS(3)
C        RELATIVE CAMERA COORDINATES IN CURRENT DOCKING AID FRAME CENTERED
C        AT CENTER LIGHT
      REAL RHO
C        MEASURED RANGE, CAMERA TO TARGET DOCKING AID CENTER LIGHT
      REAL RPYERR(3)
C        MEASURED ROLL, PITCH, AND YAW ERRORS IN RADIANS
      REAL STATE(14)
C        CHASE VEHICLE STATE VECTOR
      REAL T
C        ELAPSED TIME
      REAL THETA
C        ANGLE BETWEEN CAMERA HORIZONTAL AND ELLIPSE MAJOR AXIS
      REAL AT(3,3), TAOUT(3,3)
C        TARGET SPACECRAFT DIRECTION COSINE MATRIX (WITH RESPECT TO TRUTH
C        COORDINATE SYSTEM), AND ITS TRANSPOSE
      REAL XC, YC
C        COORDINATES OF CENTER OF ELLIPSE, AT FOCAL PLANE, IN METERS
      LOGICAL DOCKED
C        TRUE IF CHASE VEHICLE IS WITHIN DOCKING RADIUS
      LOGICAL VALID
C        TRUE IF MEASUREMENT VALID (LIGHT IN FIELD OF VIEW)
```



```
C
      COMMON/SIMUL/IDUMMY, OUT
      COMMON/HNGOFF/DUMMY3(6), HDC, HDT
      COMMON/VEHIC/DUMMY1(98), DOKRAD, DUMMY2(14)
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      VALID= TRUE
C     (* 410 - TAKE MEASUREMENT *)
         CALL LOCATE(T, STATE, VALID, RELPOS, RHO, XC, YC, THETA)
C     (* 490 - TAKE IMU MEASUREMENT *)
         CALL IMU(STATE, ACV, ACVT, ATRATE)
C     (* 420 - PROPAGATE TRUE STATE TO TIME OF THRUSTER COMMAND CHANGE *)
         CALL PROPTR( 5333333, F, STATE, T)
      IF( NOT VALID) GO TO 10
C        (* 4A0 - CALCULATE ROLL, PITCH AND YAW ERRORS *)
            CALL RPY(XC, YC, RPYERR)
C        (* 440 - INTERPRET IMAGE IN TERMS OF TARGET ATTITUDE AND CHASE
C           VEHICLE LOCATION *)
            CALL ATITUD(ACV, ACVT, RELPOS, RHO, THETA, ANEW, CVPOS)
C        (* 430 - SELECT ATTITUDE INTERPRETATION *)
            CALL SELEC1(AOLD, ANEW, P, ESTATE, CONTRL, TRACK)
C        (* 450 - USE MEASUREMENT TO UPDATE STATE ESTIMATE *)
            CALL INCORP(CVPOS(1,CONTRL), P(1,1,CONTRL), ESTATE(1,CONTRL))
            CALL INCORP(CVPOS(1,TRACK), P(1,1,TRACK), ESTATE(1,TRACK))
      GO TO 20
   10 CONTINUE
C        (* 480 - ESTIMATE ATTITUDE ERROR FROM STATE *)
            CALL ESTRPY(ESTATE(1,CONTRL), ACVT, RPYERR)
   20 CONTINUE
C     (* WRITE PARAMETERS TO DISK FILE *)
         CALL TRGATT(TAOUT, T)
         WRITE(OUT) T, STATE, TAOUT
C     (* 460 - PROPAGATE STATE ESTIMATES TO TIME OF THRUSTER COMMAND CHANGE *)
         CALL PROPES(P(1,1,CONTRL), ESTATE(1,CONTRL), F, 5333333, ACVT)
         CALL PROPES(P(1,1,TRACK), ESTATE(1,TRACK), F, 5333333, ACVT)
C     (* 470 - SET GOAL FOR NEXT MEASUREMENT INTERVAL *)
         CALL SETGOL(ESTATE(1,CONTRL), P(1,1,CONTRL), T, ACV, ANEW(1,1,
     A      CONTRL), GXL, GXH, GYL, GYH, GZL, GZH, DEDLIN, VALID)
C     (* 480 - CONTROL THRUSTERS *)
         CALL THRUST(F, ESTATE(1,CONTRL), GXL, GXH, GYL, GYH, GZL, GZH, AMAX1(
     A      DEDLIN-T, 0 ), ACV, ATRATE, RPYERR)
C     (* 420 - PROPAGATE TRUE STATE TO TIME OF NEXT MEASUREMENT *)
         CALL PROPTR(0 7, F, STATE, T)
C     (* 460 - PROP  STATE ESTIMATES TO TIME OF NEXT MEASUREMENT *)
         CALL PROPES(P(1,1,CONTRL), ESTATE(1,CONTRL), F, 0 7, ACVT)
         CALL PROPES(P(1,1,TRACK), ESTATE(1,TRACK), F, 0 7, ACVT)
C     (* SEE IF CHASE VEHICLE HAS DOCKED YET *)
         CALL MMLT(R1, ACVT, HDC, 3, 3, 1)
         CALL MADD(R2, R1, STATE, 3, 1)
         CALL TRGATT(TAOUT, T)
         CALL MTRN(AT, TAOUT, 3)
         CALL MMLT(R3, AT, R2, 3, 3, 1)
         CALL MSUB(R4, HDT, R3, 3, 1)
         DOCKED=(SQRT(R4(1)**2+R4(2)**2+R4(3)**2) LE DOKRAD)  OR
     A      (T GT 280 )  OR
     B      (STATE(1) LT -380 )  OR  (STATE(1) GT 15 )  OR
     C      (AMAX1(ABS(STATE(2)), ABS(STATE(3))) GT 200 )
C     (* WRITE PARAMETERS TO DISK FILE *)
         WRITE(OUT) T, STATE, TAOUT
C        WRITE(1,901) T, STATE(1), STATE(2), STATE(3)
C        WRITE(1,902) STATE(4), STATE(5), STATE(6)
C        WRITE(1,903) STATE(10), STATE(11), STATE(12), STATE(13)
C        WRITE(1,904) STATE(7), STATE(8), STATE(9)
C        WRITE(1,905) STATE(14)
      RETURN
C
  901 FORMAT(/' -----TIME=',F10.2/' POSITION IS ',3G14 7)
  902 FORMAT(' VELOCITY IS ',3G14 7)
  903 FORMAT(' QUATERNION IS ',4G.4 7)
  904 FORMAT(' ANGULAR MOMENTUM IS ',3G14 7)
  905 FORMAT(' MASS IS ',G14 7)
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE LOCATE(T, STATE, VALID, RELPOS, RHO, XC, YC, THETA)
C     (* 410 - TAKE MEASUREMENT *)
C
C     CALLS
C        TRGATT, MMLT, DIRMAT, MSUB, RANFN, MTRN, ELIP
C     CALLED BY
C        DOCK
C
C     INPUT
C        FOCLEN, TPHIH, TPHIV, STATE, VALID, T, D, HC, HT
C     OUTPUT
C        RELPOS, RHO, XC, YC, THETA VALID
C
      REAL AC(3,3), TRNAC(3,3)
C        CHASE VEHICLE DIRECTION COSINE MATRIX AND ITS TRANSPOSE
      REAL D
C        DIAMETER OF RING OF LIGHTS
      REAL AT(3,3), TRNAT(3,3)
C        TARGET DIRECTION COSINE MATRIX & ITS TRANSPOSE
      REAL HC(3), HT(3)
C        CAMERA POSITION IN CHASE VEHICLE BODY FRAME AND DOCKING AID POSITION
C        IN TARGET SPACECRAFT BODY FRAME, RESPECTIVELY
      REAL FOCLEN
C        LENS FOCAL LENGTH IN METERS
      INTEGER LAMP
C        LAMP NUMBER (1-8)
      REAL STATE(14)
C        TRUE CHASE VEHICLE STATE
      REAL T
C        ELAPSED TIME
      REAL THETA
C        ANGLE FROM CAMERA HORIZONTAL TO ELLIPSE MAJOR AXIS
      REAL TPHIH, TPHIV
C        TANGENTS OF HALF-FIELD-OF-VIEW ANGLES IN HORIZONTAL AND VERTICAL
C        DIRECTIONS FOR CAMERA SCANNING, RESPECTIVELY
      REAL X(8), Y(8)
C        COORDINATES OF CENTROID OF LAMP IMAGES, IN METERS
      REAL XC, YC
C        COORDINATES OF CENTER OF ELLIPSE, AT FOCAL PLANE, IN METERS
      LOGICAL VALID
C        TRUE IF MEASUREMENT VALID (LIGHTS IN FIELD OF VIEW)
      REAL VLT(3)
C        COORDINATES OF LAMP WITH RESPECT TO CAMERA
      REAL VL(3), V1(3), V2(3), V3(3), V5(3), V6(3), ANG, R
C        INTERMEDIATE RESULTS WITH NO PROBLEM-RELATED SIGNIFICANCE
C
      COMMON/OPTSYS/TPHIH, TPHIV, FOCLEN
      COMMON/HNCOFF/HC, HT, DUMMY(6)
      COMMON/ELIPSE/D, DUMMY1
```



```
C
C     (* 905 - COMPUTE TARGET AND CHASE VEHICLE ATTITUDES *)
         CALL TRGATT(TRNAT, T)
         CALL DIRMAT(STATE(10), AC, TRNAC)
      VL(1)=HT(1)
      R=D/2
      DO 10 LAMP=1, 8
C        (* FIND LAMP POSITION IN CAMERA COORDINATE SYSTEM *)
            ANG=0.7853982*FLOAT(LAMP-1)
            VL(2)=HT(2)+R*SIN(ANG)
            VL(3)=HT(3)+R*COS(ANG)
            CALL MMLT(V1, TRNAT, VL, 3, 3, 1)
            CALL MSUB(V2, V1, STATE, 3, 1)
            CALL MMLT(V3, AC, V2, 3, 3, 1)
            CALL MSUB(VLT, V3, HC, 3, 1)
C        (* FIND DEFLECTION *)
            X(LAMP)=-VLT(2)*FOCLEN/VLT(1)
            Y(LAMP)=VLT(3)*FOCLEN/VLT(1)
C        (* BE SURE DOCKING AID IS IN FIELD OF VIEW OF CAMERA *)
            VALID=VALID.AND. VLT(1) GT 0
            IF( NOT VALID) RETURN
C        (* BE SURE DOCKING AID IS ON VISIBLE SIDE OF TARGET SPACECRAFT *)
            CALL MMLT(V5, TRNAC, VLT, 3, 3, 1)
            CALL MTRN(AT, TRNAT, 3)
            CALL MMLT(V6, AT, V5, 3, 3, 1)
            VALID=VALID AND V6(1) GT 0
C        (* CORRUPT CENTROID COORDINATES WITH NOISE *)
            X(LAMP)=RANFN(2, X(LAMP), TPHIH*FOCLEN/600 )
            Y(LAMP)=RANFN(2, Y(LAMP), TPHIV*FOCLEN/600 )
   10 CONTINUE
      IF( NOT VALID) RETURN
C     (* 411 - INTERPRET THE CORRUPTED IMAGE *)
         CALL ELIP(X, Y, RELPOS, RHO, XC, YC, THETA)
      RETUR''
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE ELIP(X,Y,RELPOS,RHO,XC,YC,THETA)
C     (* 411 - INTERPRET IMAGE OF ELLIPSE *)
C
C     CALLS
C        MCALC,EPAR,SQR1
C
C     CALLED BY
C        LOCATE
C
C     INPUT
C        X,Y
C     OUTPUT
C        RELPOS,RHO,XC,YC,THETA
C
      REAL D
C        CIRCLE DIAMETER
      REAL RATIO,MAJOR
C        RATIO OF ELLIPSE MINOR AND MAJOR AXES AND SEMIMAJOR AXIS LENGTH (AT
C        FOCAL PLANE)
      REAL RELPOS(3),RHO
C        CAMERA POSITION IN DOCKING AID REFERENCE FRAME AND ITS MAGNITUDE
      REAL THETA
C        ANGLE BETWEEN ELLIPSE MAJOR AXIS AND CAMERA HORIZONTAL
      REAL X(8),Y(8)
C        HORIZONTAL AND VERTICAL DEFLECTIONS OF IMAGES OF 8 LIGHTS
      REAL XC,YC
C        COORDINATES OF CENTER OF ELLIPSE, AT FOCAL PLANE, IN METERS
      REAL XXP,YYP,XYP
C        MOMENTS USED IN CALCULATING ELLIPSE PARAMETERS
C
      COMMON/OPTSYS/DUMMY(2),FOCLEN
      COMMON/ELIPSE/D,DUMMY1
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* 411 1 - COMPUTE MOMENTS *)
         CALL MCALC(X,Y,XXP,YYP,XYP,XC,YC)
C     (* 411 2 - COMPUTL ELLIPSE PARAMETERS *)
         CALL EPAR(XXP,YYP,XYP,THETA,RATIO,MAJOR)
C     (* COMPUTE RHO, RELPOS *)
         RHO=0.5*FOCLEN*D/MAJOR
         RELPOS(1)=-RHO*RATIO
         RELPOS(2)=0
         RELPOS(3)=RHO*SQR1(1.-RATIO**2)
      RETURN
C
      END
```



```
C
      SUBROUTINE MCALC(X,Y,XXP,YYP,XYP,XC,YC)
C     (* 411 1 - COMPUTE MOMENTS FROM LIGHT PATTERN *)
C
C     CALLS
C        <NONE>
C     CALLED BY
C        ELIP
C
C      INPUT
C        X,Y
C     OUTPUT
C        XC,YC,XXP,XYP,YYP
C
      INTEGER I
C        DO LOOP INDEX
      REAL XC,YC
C        COORDINATES OF CENTER OF ELLIPSE
      REAL X(8),Y(8)
C        COORDINATES OF THE 8 LIGHT IMAGES
      REAL XX,XY,YY
C        MOMENTS COMPUTED FROM LIGHT PATTERN
      REAL XXP,XYP,YYP
C        MOMENTS AFTER CORRECTION FOR OFFSET FROM CENTER OF FIELD OF VIEW
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      XC=0
      YC=0
      XX=0
      YY=0
      XY=0
      DO 10 I=1,8
         XC=XC+X(I)
         YC=YC+Y(I)
         XX=XX+X(I)**2
         YY=YY+Y(I)**2
         XY=XY+X(I)*Y(I)
   10 CONTINUE
      XC=XC/8
      YC=YC/8
      XX=XX/8
      YY=YY/8
      XY=XY/8
      XXP=XX-XC**2
      YYP=YY-YC**2
      XYP=XY-XC*YC
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

APPENDIX C -- RING-OF-LIGHTS VERSION OF SIMULATION PROGRAM
PAGE 16

```
C
      SUBROUTINE EPAR(XXP,YYP,XYP,THETA,RATIO,MAJOR)
C     (* 411 2 - COMPUTE ELLIPSE PARAMETERS *)
C
C     CALLS
C        SQR1
C     CALLED BY
C        ELIP
C
C     INPUT
C        XXP,XYP,YYP,K
C     OUTPUT
C        THETA,RATIO
C
      REAL K
C        CONSTANT RELATING MOMENTS TO ELLIPSE SIZE
      REAL MAJOR,MINOR,RATIO
C        SEMIMAJOR AND SEMIMINOR AXES OF ELLIPSE, AT FOCAL PLANE, IN METERS
C        AND THE RATIO MINOR/MAJOR
      REAL THETA
C        ANGLE FROM CAMERA HORIZONTAL TO ELLIPSE MAJOR AXIS
      REAL XXP,XYP,YYP
C        MOMENTS COMPUTED FROM LIGHT PATTERN
      REAL LAMDA1,LAMDA2,DISC
C        INTERMEDIATE RESULTS
C
      COMMON/ELIPSE/DUMMY,K
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      DISC=SQR1((XXP-YYP)**2+4.*(XYP**2))
      LAMDA1=0.5*(XXP+YYP+DISC)
      LAMDA2=0.5*(XXP+YYP-DISC)
      IF(LAMDA1.LT.LAMDA2) GO TO 10
         MAJOR=K*SQR1(LAMDA1)
         MINOR=K*SQR1(LAMDA2)
         GO TO 20
   10 CONTINUE
         MAJOR=K*SQR1(LAMDA2)
         MINOR=K*SQR1(LAMDA1)
   20 CONTINUE
      RATIO=MINOR/MAJOR
      THETA=0.5*ATAN2(2.*XYP,XXP-YYP)
      RETURN
C
      END
```

APPENDIX C -- RING-OF-LIGHTS VERSION OF SIMULATION PROGRAM
PAGE 17

```
C
      SUBROUTINE PROPTR(DT,F,STATE,T)
C     (* 420 - PROPAGATE TRUE STATE BY 4TH-ORDER RUNGE-KUTTA INTEGRATION *)
C
C     CALLS
C        COMPK1,COMPK2,COMPK3,COMPK4,POINT
C     CALLED BY
C        DOCK
C
C     INPUT
C        DT,F,STATE,T
C     OUTPUT
C        STATE,T
C
      REAL DT
C        INTEGRATION INTERVAL SIZE
      REAL F(14)
C        FORCES FROM THRUSTERS AFTER SELECTION
      INTEGER I
C        DO LOOP INDEX
      REAL K1(14),K2(14),K3(14),K4(14)
C        VECTORS K1 THRU K4 USED IN RUNGE-KUTTA INTEGRATION
      REAL QM
C        SQUARE ROOT OF SUM OF SQUARES OF QUATERNION ELEMENTS
      REAL STATE(14)
C        CHASE VEHICLE STATE VECTOR
      REAL STEP
C        STEP SIZE FOR INTEGRATION
      REAL T,TO
C        ELAPSED TIME AND TIME AT START OF INTEGRATION INTERVAL
      REAL TLEFT
C        TIME LEFT TO GO OUT OF DT
      PARAMETER STEPMX=0.2
C        MAXIMUM INTEGRATION STEP SIZE
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      TLEFT=DT
      TO=T
   10 IF(TLEFT.LE.0) GO TO 40
         STEP=AMIN1(STEPMX,TLEFT)
         TLEFT=TLEFT-STEP
C        (* 421 - COMPUTE K1 *)
            CALL COMPK1(F,K1,STATE,STEP,T)
C        (* 422 - COMPUTE K2 *)
            CALL COMPK2(F,K1,STATE,STEP,T,K2)
C        (* 423 - COMPUTE K3 *)
            CALL COMPK3(F,K2,STATE,STEP,T,K3)
C        (* 424 - COMPUTE K4 *)
            CALL COMPK4(F,K3,STATE,STEP,T,K4)
C        (* COMPUTE NEW STATE *)
            DO 20 I=1,14
               STATE(I)=STATE(I)+(K1(I)+2.*K2(I)+2.*K3(I)+K4(I))/6
   20       CONTINUE
C        (* NORMALIZE QUATERNION *)
            QM=SQRT(STATE(10)**2+STATE(11)**2+STATE(12)**2+
     A         STATE(13)**2)
            DO 30 I=10,13
               STATE(I)=STATE(I)/QM
   30       CONTINUE
C        (* 425 - POINT SIMULATION CAMERA *)
            CALL POINT(STATE)
         T=T+STEP
         GO TO 10
   40 CONTINUE
      T=TO+DT
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE COMPK1(F,K1,STATE,STEP,T)
C     (* 421 - COMPUTE VECTOR K1 FOR RUNGE-KUTTA INTEGRATION *)
C
C     CALLS
C        STPRIM,MSCL,MADD
C     CALLED BY
C        PROPTR
C
C     INPUT
C        F,STATE,STEP,T,FULDIS,INERCV,MEMPTY
C     OUTPUT
C        K1
C
      REAL DSTATE(14)
C        TIME DERIVATIVE OF STATE VECTOR
      REAL F(14)
C        FORCES FROM THRUSTERS AFTER SELECTION
      REAL FULDIS(3,3)
C        CHASE VEHICLE FUEL-DISTRIBUTION INERTIA TENSOR
      INTEGER I
C        DO LOOP INDEX
      REAL INERFL(3,3)
C        INERTIA OF FUEL
      REAL INERTA(3,3)
C        INERTIA OF CHASE VEHICLE (LOADED)
      REAL K1(14)
C        VECTOR K1 USED IN RUNGE-KUTTA INTEGRATION
      REAL STATE(14)
C        CHASE VEHICLE STATE VECTOR
      REAL STEP
C        STEP SIZE FOR INTEGRATION
      REAL T
C        ELAPSED TIME
C
      COMMON/MASPRP/FULDIS,INERCV,MEMPTY
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* COMPUTE INERTIA *)
         CALL MSCL(INERFL,FULDIS,3,3,STATE(14)-MEMPTY)
         CALL MADD(INERTA,INERCV,INERFL,3,3)
C     (* 902 - COMPUTE STATE DERIVATIVE *)
         CALL STPRIM(STATE,F,INERTA,T,DSTATE)
C     (* COMPUTE K1=STEP*(STATE DERIVATIVE)  *)
         DO 10 I=1,14
            K1(I)=STEP*DSTATE(I)
   10    CONTINUE
      RETURN
C
      END
```



```
C
      SUBROUTINE COMPK2(F,K1,STATE,STEP,T,K2)
C     (* 422 - DETERMINE VECTOR K2, FOR RUNGE-KUTTA INTEGRATION *)
C
C     CALLS
C        MADD,MSCL,STPRIM
C     CALLED BY
C        PROPTR
C
C     INPUT
C        F,FULDIS,INERCV,K1,MEMPTY,STATE,STEP,T
C     OUTPUT
C        K2
C
      REAL DSTATE(14)
C        TIME DERIVATIVE OF STATE VECTOR
      REAL F(14)
C        FORCES FROM THRUSTERS AFTER SELECTION
      REAL FULDIS(3,3)
C        CHASE VEHICLE FUEL-DISTRIBUTION INERTIA TENSOR
      INTEGER I
C        DO LOOP INDEX
      REAL INERCV(3,3)
C        INERTIA OF EMPTY CHASE VEHICLE
      REAL INERFL(3,3)
C        INERTIA OF FUEL
      REAL INERTA(3,3)
C        INERTIA OF CHASE VEHICLE (LOADED)
      REAL K1(14),K2(14)
C        VECTORS K1 AND K2 USED IN RUNGE-KUTTA INTEGRATION
      REAL MEMPTY
C        MASS OF CHASE VEHICLE WITHOUT FUEL
      REAL STATE(14)
C        CHASE VEHICLE STATE VECTOR
      REAL STEP
C        STEP SIZE FOR INTEGRATION
      REAL T
C        ELAPSED TIME
      REAL TEMPST(14)
C        TEMPORARY STATE VECTOR
C
      COMMON/MASPRP/FULDIS,INERCV,MEMPTY
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* COMPUTE TEMPORARY STATE *)
         DO 10 I=1,14
            TEMPST(I)=STATE(I)+0.5*K1(I)
   10    CONTINUE
C     (* DETERMINE TEMPORARY INERTIA AS FUNCTION OF TEMPORARY
C        STATE MASS *)
         CALL MSCL(INERFL,FULDIS,3,3,TEMPST(14)-MEMPTY)
         CALL MADD(INERTA,INERCV,INERFL,3,3)
C     (* 902 - COMPUTE DERIVATIVE AT TEMPORARY STATE *)
         CALL STPRIM(TEMPST,F,INERTA,T+.5*STEP,DSTATE)
C     (* COMPUTE K2=STEP*(DERIVATIVE AT TEMPORARY STATE *)
         DO 20 I=1,14
            K2(I)=STEP*DSTATE(I)
   20    CONTINUE
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE COMPK3(F,K2,STATE,STEP,T,K3)
C     (* 423 - DETERMINE VECTOR K3, FOR RUNGE-KUTTA INTEGRATION *)
C
C     CALLS
C        MADD,MSCL,STPRIM
C     CALLED BY
C        PROPTR
C
C     INPUT
C        F,FULDIS,INERCV,K2,MEMPTY,STATE,STEP,T
C     OUTPUT
C        K3
C
      REAL DSTATE(14)
C        TIME DERIVATIVE OF STATE VECTOR
      REAL F(14)
C        FORCES FROM THRUSTERS AFTER SELECTION
      REAL FULDIS(3,3)
C        CHASE VEHICLE FUEL-DISTRIBUTION INERTIA TENSOR
      INTEGER I
C        DO LOOP INDEX
      REAL INERCV(3,3)
C        INERTIA OF EMPTY CHASE VEHICLE
      REAL INERFL(3,3)
C        INERTIA OF FUEL
      REAL INERTA(3,3)
C        INERTIA OF CHASE VEHICLE (LOADED)
      REAL K2(14),K3(14)
C        VECTORS K2 AND K3 USED IN RUNGE-KUTTA INTEGRATION
      REAL MEMPTY
C        MASS OF CHASE VEHICLE WITHOUT FUEL
      REAL STATE(14)
C        CHASE VEHICLE STATE VECTOR
      REAL STEP
C        STEP SIZE FOR INTEGRATION
      REAL T
C        ELAPSED TIME
      REAL TEMPST(14)
C        TEMPORARY STATE VECTOR
C
      COMMON/MASPRP/FULDIS,INERCV,MEMPTY
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* COMPUTE TEMPORARY STATE *)
        DO 10 I=1,14
           TEMPST(I)=STATE(I)+0.5*K2(I)
   10   CONTINUE
C     (* DETERMINE TEMPORARY INERTIA AS A FUNCTION OF TEMPORARY
C              STATE MASS *)
        CALL MSCL(INERFL,FULDIS,3,3,TEMPST(14)-MEMPTY)
        CALL MADD(INERTA,INERCV,INERFL,3,3)
C     (* 902 - COMPUTE DERIVATIVE AT TEMPORARY STATE *)
        CALL STPRIM(TEMPST,F,INERTA,T+.5*STEP,DSTATE)
C     (* COMPUTE K3=STEP*(DERIVATIVE AT TEMPORARY STATE *)
        DO 20 I=1,14
           K3(I)=STEP*DSTATE(I)
   20   CONTINUE
      RETURN
C
      END
```



```
C
      SUBROUTINE COMPK4(F,K3,STATE,STEP,T,K4)
C     (* 424 - DETERMINE VECTOR K4, FOR RUNGE-KUTTA INTEGRATION *)
C
C     CALLS
C        MADD,MSCL,STPRIM
C     CALLED BY
C        PROPTR
C
C     INPUT
C        F,FULDIS,INERCV,K3,MEMPTY,STATE,STEP,T
C     OUTPUT
C        K4
C
      REAL DSTATE(14)
C        TIME DERIVATIVE OF STATE VECTOR
      REAL F(14)
C        FORCES FROM THRUSTERS AFTER SELECTION
      REAL FULDIS(3,3)
C        CHASE VEHICLE FUEL-DISTRIBUTION INERTIA TENSOR
      INTEGER I
C        DO LOOP INDEX
      REAL INERCV(3,3)
C        INERTIA OF EMPTY CHASE VEHICLE
      REAL INERFL(3,3)
C        INERTIA OF FUEL
      REAL INERTA(3,3)
C        INERTIA OF CHASE VEHICLE (LOADED)
      REAL K3(14),K4(14)
C        VECTORS K3 AND K4 USED IN RUNGE-KUTTA INTEGRATION
      REAL MEMPTY
C        MASS OF CHASE VEHICLE WITHOUT FUEL
      REAL STATE(14)
C        CHASE VEHICLE STATE VECTOR
      REAL STEP
C        STEP SIZE FOR INTEGRATION
      REAL T
C        ELAPSED TIME
      REAL TEMPST(14)
C        TEMPORARY STATE VECTOR
C
      COMMON/MASPRP/FULDIS,INERCV,MEMPTY
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* COMPUTE TEMPORARY STATE *)
        DO 10 I=1,14
           TEMPST(I)=STATE(I)+K3(I)
   10   CONTINUE
C     (* DETERMINE TEMPORARY INERTIA AS A FUNCTION OF TEMPORARY
C              STATE MASS *)
        CALL MSCL(INERFL,FULDIS,3,3,TEMPST(14)-MEMPTY)
        CALL MADD(INERTA,INERCV,INERFL,3,3)
C     (* 902 - COMPUTE DERIVATIVE AT TEMPORARY STATE *)
        CALL STPRIM(TEMPST,F,INERTA,T+STEP,DSTATE)
C     (* COMPUTE K4=STEP*(DERIVATIVE AT TEMPORARY STATE *)
        DO 20 I=1,14
           K4(I)=STEP*DSTATE(I)
   20   CONTINUE
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE POINT(STATE)
C     (* 425 - POSITION SIMULATION CAMERA *)
C     (* THIS DUMMY SUBROUTINE IS FOR LATER EXPANSION TO PHYSICAL SIMULATION*)
C
C     CALLS
C        <NONE>
C     CALLED BY
C        PROPTR
C
      RETURN
C
      END
```



```
C
      SUBROUTINE SELEC1(AOLD, ANEW, P, ESTATE, CONTRL, TRACK)
C     (* 430 - SELECT TARGET ATTITUDE INTERPRETATION *)
C
C     CALLS
C        TSTATT
C     CALLED BY
C        DOCK
C
C     INPUT
C        AOLD, P, ESTATE
C     OUTPUT
C        AOLD, P, ESTATE, CONTRL, TRACK
C
      INTEGER CONTRL, TRACK
C        SUBSCRIPTS TO INDICATE VERSIONS OF ESTATE AND P FOR CONTROL AND
C        KEEPING TRACK OF
      REAL ESTATE(6,2),P(6,6,2)
C        STATE ESTIMATES AND COVARIANCE MATRICES FOR 2 IMAGE INTERPRETATIONS
      INTEGER I,J
C        DO LOOP INDICES
      INTEGER OBEST, NBEST
C        SUBSCRIPTS SELECTING BEST-MATCHING OLD AND NEW DIRECTION COSINE
C        MATRICES
      INTEGER OWORST
C        SUBSCRIPT SELECTING WHATEVER OBEST DOES NOT SELECT
      REAL BESTA
C        SMALLEST EULER ANGLE BETWEEN ANY PAIR OF DIRECTION COSINE MATRICES
C        (IGNORING ANY ROLL COMPONENT)
      REAL AOLD(3,3,2),ANEW(3,3,2)
C        OLD AND NEW DRECTION COSINE MATRIX PAIRS
      REAL ANGLE
C        ANGLE BETWEEN A PAIR OF DIRECTION COSINE MATRICES, IGNORING ROLL
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* COMPARE BOTH OLD, NEW INTERPRETATIONS TO SELECT BEST MATCH *)
         BESTA=1 E6
         DO 10 I=1,2
            DO 10 J=1,2
               CALL TSTATT(AOLD(1,1,I),ANEW(1,1,J),ANGLE)
               IF(ANGLE GE BESTA) GO TO 10
                  OBEST=I
                  NBEST=J
   10    CONTINUE
C     (* SELECT STATE ESTIMATE & COVARIANCE ACCORDINGLY *)
         IF(OBEST EQ 1) GO TO 20
            OWORST=1
            GO TO 30
   20    CONTINUE
            OWORST=2
   30    CONTINUE
         DO 50 I=1,6
            DO 40 J=1,6
               P(I,J,OWORST)=P(I,J,OBEST)
   40       CONTINUE
            ESTATE(I,OWORST)=ESTATE(I,OBEST)
   50    CONTINUE
C     (* SELECT NEW INTERPRETATION FOR CONTROL & ALTERNATE INTERPRETATION
C        TO KEEP TRACK OF *)
         CONTRL=NBEST
         IF(NBEST EQ 1) GO TO 60
            TRACK=1
            GO TO 70
   60    CONTINUE
            TRACK=2
   70    CONTINUE
C     (* SET AOLD TO ANEW *)
         DO 80 I=1,3
            DO 80 J=1,3
               AOLD(I,J,1)=ANEW(I,J,1)
               AOLD(I,J,2)=ANEW(I,J,2)
   80    CONTINUE
      RETURN
C
      END
```

```
C
      SUBROUTINE TSTATT(A1,A2,ANGLE)
C     (* 431 - CALCULATE ANGLE BETWEEN ATTITUDES SPECIFIED BY DIRECTION
C        COSINE MATRICES A1 AND A2, IGNORING ANY ROLL COMPONENT *)
C
C     CALLS
C        TOQUAT,ACOS,SQRT
C     CALLED BY
C        SELECI
C
C     INPUT
C        A1,A2
C     OUTPUT
C        ANGLE
C
      REAL A1(3,3),A2(3,3)
C        DIRECTION COSINE MATRICES TO BE COMPARED
      REAL Q1(4),Q2(4)
C        QUATERNIONS CORRESPONDING TO A1,A2
      REAL A,B
C        INTERMEDIATE RESULTS
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* 431 1 - CONVERT DIRECTION COSINE MATRICES TO QUATERNIONS *)
         CALL TOQUAT(A2,Q2)
         CALL TOQUAT(A1,Q1)
C     (* FIND ANGLE BETWEEN ATTITUDES, IGNORING ROLL COMPONENTS *)
         A=Q2(1)*Q1(1)+Q2(2)*Q1(2)+Q2(3)*Q1(3)+Q2(4)*Q1(4)
         B=Q2(1)*Q1(4)+Q2(2)*Q1(3)-Q2(3)*Q1(2)-Q2(4)*Q1(1)
         ANGLE=ACOS(SQRT(A**2+B**2))
      RETURN
C
      END
```



```
C
      SUBROUTINE TOQUAT(A,Q)
C     (* 431 1 - CONVERT DIRECTION COSINE MATRIX TO QUATERNION *)
C
C     CALLS
C        SQR1
C     CALLED BY
C        TSTATT
C
C     INPUT
C        A
C     OUTPUT
C        Q
C
      REAL A(3,3)
C        DIRECTION COSINE MATRIX
      REAL Q(4)
C        CORRESPONDING QUATERNION
      REAL FACTOR
C        INTERMEDIATE RESULTS
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      Q(4)=0.5*SQR1(1.+A(1,1)+A(2,2)+A(3,3))
      IF(Q(4).LT.0.5) GO TO 10
         FACTOR=0.25/Q(4)
         Q(1)=FACTOR*(A(2,3)-A(3,2))
         Q(2)=FACTOR*(A(3,1)-A(1,3))
         Q(3)=FACTOR*(A(1,2)-A(2,1))
         GO TO 40
   10 Q(1)=0.5*SQR1(1.+A(1,1)-A(2,2)-A(3,3))
      IF(Q(1).LT.0.5) GO TO 20
         FACTOR=0.25/Q(1)
         Q(2)=FACTOR*(A(1,2)+A(2,1))
         Q(3)=FACTOR*(A(1,3)+A(3,1))
         Q(4)=FACTOR*(A(2,3)-A(3,2))
         GO TO 40
   20 Q(2)=0.5*SQR1(1.+A(2,2)-A(1,1)-A(3,3))
      IF(Q(2).LT.0.5) GO TO 30
         FACTOR=0.25/Q(2)
         Q(1)=FACTOR*(A(1,2)+A(2,1))
         Q(3)=FACTOR*(A(2,3)+A(3,2))
         Q(4)=FACTOR*(A(3,1)-A(1,3))
         GO TO 40
   30 CONTINUE
         Q(3)=0.5*SQR1(1.+A(3,3)-A(2,2)-A(1,1))
         FACTOR=0.25/Q(3)
         Q(1)=FACTOR*(A(1,3)+A(3,1))
         Q(2)=FACTOR*(A(2,3)+A(3,2))
         Q(4)=FACTOR*(A(1,2)-A(2,1))
   40 CONTINUE
      IF(Q(4).GE.0.) GO TO 50
         Q(1)=-Q(1)
         Q(2)=-Q(2)
         Q(3)=-Q(3)
         Q(4)=-Q(4)
   50 CONTINUE
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE ATITUD(ACV,ACVT,RELPOS,RHO,THETA,ANEW,CVPOS)
C     (* 440 - DETERMINE TARGET ATTITUDE AND CHASE VEHICLE POSITION IN PRIMARY
C        REFERENCE FRAME *)
C
C     CALLS
C        FINDCV,DIRMAT,QUATRN,MADD,MMLT,MSUB
C     CALLED BY
C        DOCK
C
C     INPUT
C        RELPOS,RHO,THETA,ACV,ACVT,HC,HT
C     OUTPUT
C        ANEW,CVPOS
C
      REAL ACV(3,3),ACVT(3,3)
C        DIRECTION COSINE MATRIX DESCRIBING CURRENT CHASE VEHICLE ATTITUDE
C        RELATIVE TO PRIMARY REFERENCE FRAME, AND ITS TRANSPOSE
      REAL ANEW(3,3,2),ATT(3,3)
C        DIRECTION COSINE MATRIX OF TARGET SPACECRAFT (RELATIVE TO PRIMARY
C        REFERENCE FRAME, 2 INTERPRETATIONS), AND TRANSPOSE OF THIS MATRIX
      REAL CAMPOS(3)
C        MEASURED CAMERA POSITION IN COORD. SYS. CENTERED AT CENTER OF
C        DOCKING AID AND PARALLEL TO PRIMARY REFERENCE FRAME
      REAL CVPOS(3,2)
C        MEASURED CHASE VEHICLE POSITION IN PRIMARY REF  FRAME (TWO INTER-
C        PRETATIONS)
      REAL HC(3),HT(3)
C        CAMERA POSITION IN CHASE VEHICLE BODY FRAME AND DOCKING AID POSITION
C        IN TARGET SPACECRAFT BODY FRAME, RESPECTIVELY
      INTEGER INTERP
C        DO LOOP INDEX   INTERPRETATION NUMBER
      REAL QT(4,2)
C        QUATERNIONS CORRESPONDING TO AT
      REAL RELPOS(3)
C        RELATIVE CHASE VEHICLE COORDINATES IN CURRENT TARGET SPACECRAFT
C        REFERENCE FRAME
      REAL RHO
C        DISTANCE BETWEEN LIGHTS AND CAMERA
      REAL XC,YC
C        COORDINATES OF CENTER OF ELLIPSE
      REAL V1(3),V2(3),V3(3)
C        INTERMEDIATE RESULTS WITH NO PROBLEM-RELATED SIGNIFICANCE
C
      COMMON/HNOOFF/HC,HT,DUMMY(6)
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* 441 - COMPUTE CAMERA POSITION WITH RESPECT TO DOCKING AID *)
      CALL FINDCV(ACVT,RHO,XC,YC,CAMPOS)
C     (* 442 - COMPUTE TARGET ATTITUDE IN PRIMARY REFERENCE FRAME *)
      CALL QUATRN(CAMPOS,RELPOS,QT,ACV,THETA)
C     (* FIND HC IN PRIMARY REFERENCE FRAME *)
      CALL MMLT(V3,ACVT,HC,3,3,1)
      DO 10 INTERP=1,2
C        (* 901 - COMPUTE TARGET DIRECTION COSINE MATRIX FROM QUATERNION *)
         CALL DIRMAT(QT(1,INTERP),ANEW(1,1,INTERP),ATT)
C        (* COMPUTE POSITION OF CHASE VEHICLE CENTER OF GRAVITY IN PRIMARY REF
C           FRAME *)
         CALL MMLT(V2,ATT,HT,3,3,1)
         CALL MADD(V1,CAMPOS,V2,3,1)
         CALL MSUB(CVPOS(1,INTERP),V1,V3,3,1)
   10 CONTINUE
      RETURN
C
      END
```



```
C
      SUBROUTINE FINDCV(ACVT,RHO,XC,YC,CAMPOS)
C     (* 441 - COMPUTE POSITION OF CAMERA FRAME PARALLEL TO PRIMARY REF  FRAME
C        BUT CENTERED AT CENTER OF DOCKING AID *)
C
C     CALLS
C        MMLT,SQRT
C     CALLED BY
C        ATTITUD
C
C     INPUT
C        ACVT,RHO,XC,YC,FOCLEN
C     OUTPUT
C        CAMPOS
C
      REAL ACVT(3,3)
C        DIRECTION COSINE MARTIX DESCRIBING CURRENT CHASE VEHICLE ATTITUDE
C        RELATIVE TO PRIMARY REFERENCE FRAME
      REAL CAMPOS(3)
C        MEASURED CAMERA POSITION IN FRAME PARALLEL TO PRIMARY REF  FRAME
C        BUT CENTERED AT CENTER OF DOCKING AID
      REAL FOCLEN
C        CAMERA LENS FOCAL LENGTH, IN METERS
      REAL RATIO
C        INTERMEDIATE RESULTS, NO PROBLEM RELATED SIGNIFICANCE
      REAL RHO
C        MEASURED DISTANCE BETWEEN DOCKIN AID CENTER AND CAMERA
      REAL XC,YC
C        HORIZONTAL AND VERTICAL COMPONENTS OF THE TARGET CENTER
      REAL VEC1(3)
C        NEGATIVE OF TARGET POSITION IN CAMERA FRAME
C
      COMMON/OPTSYS/DUMMY1(2),FOCLEN
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* COMPUTE SCALAR MULTIPLIER *)
      RATIO=RHO/SQRT(FOCLEN**2+XC**2+YC**2)
C     (* FIND NEG  OF TARGET POSITION IN CAMERA FRAME *)
      VEC1(1)=-FOCLEN*RATIO
      VEC1(2)=XC*RATIO
      VEC1(3)=-YC*RATIO
C     (* CONVERT TO CAMERA POSITION IN TARGET CENTER-LIGHT FRAME *)
      CALL MMLT(CAMPOS,ACVT,VEC1,3,3,1)
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
      SUBROUTINE QUATRN(CAMPOS,RELPOS,QT,ACV,THETA)
C     (* 442 - COMPUTE TARGET ATTITUDE QUATERNION IN PRIMARY REFERENCE FRAME
C        FROM MEASUREMENT *)
C
C     CALLS
C        ATAN2 SIN,SQRT,COS,MSCL,DIRMAT,MMLT
C     CALLED BY
C        ATITUD
C
C     INPUT
C        CAMPOS,RELPOS,ACV,THETA
C     OUTPUT
C        QT
C
      REAL ACV(3,3)
C        CHASE VEHICLE'S MEASURED DIRECTION COSINE MATRIX
      REAL AT(3,3),TRNAT(3,3)
C        DIRECTION COSINE MATRIX FOR TARGET (WITH RESPECT TO PRIMARY RE
C        FRAME  BEFORE CORRECTION) AND ITS TRANSPOSE
      REAL CAMPOS(3)
C        MEASURED CAMERA POSITION IN REFERENCE FRAME PARALLEL TO PRIMARY
C        REF  FRAME BUT CENTERED AT CENTER DOCKING AID LIGHT
      REAL DOTPRD
C        DOT PRODUCT OF CAMPOS, RELPOS
      REAL PHI
C        EULER ROTATION ANGLE
      REAL PRDMAG
C        MAGNITUDE OF CROSS PRODUCT
      REAL QC(4)
C        CORRECTION QUATERNION
      REAL QT1(4),QT(4,2)
C        TARGET ATTITUDE QUATERNION (MAYBE WITH ROTATION ABOUT LINE OF SIGHT)
C        AND CORRECTED VERSIONS OF SAME (2 POSSIBLE IMAGE INTERPRETATIONS)
      REAL RELPOS(3)
C        CAMERA COORDINATES IN CURRENT DOCKING AID REFERENCE FRAME
      REAL ROTLOS
C        COMPENSATING ROTATION ANGLE ABOUT LINE OF SIGHT
      REAL THETA
C        ANGLE BETWEEN CAMERA HORIZONTAL AND MAJOR AXIS OF ELLIPSE
      REAL V1(3)
C        INTERMEDIATE RESULTS   CROSS PRODUCT COMPONENTS
      REAL V2(3),V3(3)
C        INTERMEDIATE RESULTS
```



```
C
C     (* COMPUTE COMPONENTS OF CROSS PRODUCT TO DEFINE ROTATION AXIS FROM
C        RELPOS TO CAMPOS *)
         V1(1)=-CAMPOS(2)*RELPOS(3)+CAMPOS(3)*RELPOS(2)
         V1(2)=-CAMPOS(3)*RELPOS(1)+CAMPOS(1)*RELPOS(3)
         V1(3)=-CAMPOS(1)*RELPOS(2)+CAMPOS(2)*RELPOS(1)
         PRDMAG=SQRT(V1(1)**2+V1(2)**2+V1(3)**2)
         DOTPRD=CAMPOS(1)*RELPOS(1)+CAMPOS(2)*RELPOS(2)+CAMPOS(3)*
     A      RELPOS(3)
         IF(PRDMAG GT 0 ) GO TO 20
C           (* VECTORS ARE PARALLEL OR ANTIPARALLEL  FIND OUT WHICH *)
               QT1(1)=0
               QT1(2)=0
               IF(DOTPRD GT 0 ) GO TO 10
                  QT1(3)=1 0
                  QT1(4)=0 0
                  GO TO 30
   10          CONTINUE
                  QT1(3)=0
                  QT1(4)=1 0
                  GO TO 30
   20    CONTINUE
C           (* COMPUTE EULER ROTATION ANGLE *)
               PHI=ATAN2(PRDMAG,DOTPRD)
C           (* COMPUTE QUATERNION COMPONENTS *)
               CALL MSCL(Q 1,V1,3,1,SIN(PHI* 5)/PRDMAG)
               QT1(4)=COS(PHI* 5)
   30 CONTINUE
C     (* CORRECT FOR ROTATION ABOUT LINE OF SIGHT *)
C        (* 901 - CONVERT QT1 TO DIRECTION COSINE MATRIX *)
            CALL DIRMAT(QT1,AT,TRNAT)
C        (* FIND ROTATION OF LINE IN -X DIRECTION ON TARGET, PREDICTED
C           BY 'AT', AS COMPARED TO OBSERVED ELLIPSE MAJOR AXIS *)
            V2(1)=-TRNAT(1,1)
            V2(2)=-TRNAT(2,1)
            V2(3)=-TRNAT(3,1)
            CALL MMLT(V3,ACV,V2,3,3,1)
            ROTLOS=ATAN2(V3(2),V3(3))-THETA
            IF(ROTLOS GT 3 141593) ROTLOS=ROTLOS-6 283185
            IF(ROTLOS LT -3 141593) ROTLOS=ROTLOS+6 283185
C        (* FORM QUATERNION FOR CORRECTING ROTATION *)
            CALL MSCL(QC,RELPOS,3,1,SIN( 5*ROTLOS)/SQRT(RELPOS(1)**2+
     A         RELPOS(2)**2+RELPOS(3)**2))
            QC(4)=COS( 5*ROTLOS)
C        (* COMPUTE NET ROTATION OF QT1 FOLLOWED BY QC *)
            QT(1,1)= QC(4)*QT1(1)+QC(3)*QT1(2)-QC(2)*QT1(3)+QC(1)*
     A         QT1(4)
            QT(2,1)=-QC(3)*QT1(1)+QC(4)*QT1(2)+QC(1)*QT1(3)+QC(2)*
     A         QT1(4)
            QT(3,1)= QC(2)*QT1(1)-QC(1)*QT1(2)+QC(4)*QT1(3)+QC(3)*
     A         QT1(4)
            QT(4,1)=-QC(1)*QT1(1)-QC(2)*QT1(2)-QC(3)*QT1(3)+QC(4)*
     A         QT1(4)
C     (* FORM QUATERNION WITH ADDED HALF TURN ABOUT LINE OF SIGHT (ALTER-
C        NATE ORIENTATION WITH SAME IMAGE APPEARANCE) *)
         CALL MSCL(QC,RELPOS,3,1,1 /SQRT(RELPOS(1)**2+RELPOS(2)**2+
     A      RELPOS(3)**2))
         QT(1,2)= QC(3)*QT(2,1)-QC(2)*QT(3,1)+QC(1)*QT(4,1)
         QT(2,2)=-QC(3)*QT(1,1)+QC(1)*QT(3,1)+QC(2)*QT(4,1)
         QT(3,2)= QC(2)*QT(1,1)-QC(1)*QT(2,1)+QC(3)*QT(4,1)
         QT(4,2)=-QC(1)*QT(1,1)-QC(2)*QT(2,1)-QC(3)*QT(3,1)
      RETURN
C
      END
```

```
C
      SUBROUTINE INCORP(CVPOS,P,ESTATE)
C     (* 450 - INCORPORATE MEASUREMENT *)
C
C     CALLS
C        UPDCOV,UPDSTA,COMPG,ESTCOV,KALGAN
C     CALLED BY
C        DOCK
C
C     INPUT
C        ESTATE,P,R
C     OUTPUT
C        P,ESTATE
C
      REAL CVPOS(3)
C        MEASURED CHASE VEHICLE POSITON IN PRIMARY REFERENCE FRAME
      REAL ESTATE(6)
C        ESTIMATE OF STATE VECTOR
      REAL G(3,6)
C        PARTIAL OF MEASUREMENT WITH RESPECT TO STATE
      REAL KGAIN(6,3)
C        KALMAN GAIN MATRIX
      REAL P(6,6)
C        COVARIANCE OF STATE ESTIMATE
      REAL R(3,3)
C        MEASUREMENT COVARIANCE
C
C- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C
      (* 451 - COMPUTE JACOBIAN G *)
      CALL COMPG(ESTATE,G)
C
      (* 452 - ESTIMATE MEASUREMENT COVARIANCE *)
      CALL ESTCOV(ESTATE,R)
C
      (* 453 - COMPUTE KALMAN GAIN MATRIX *)
      CALL KALGAN(R,P,G,KGAIN)
C
      (* 454 - UPDATE STATE *)
      CALL UPDSTA(ESTATE,KGAIN,CVPOS)
C
      (* 455 - UPDATE COVARIANCE MATRIX *)
      CALL UPDCOV(P,KGAIN,G)
      RETURN
C
      END
```



```
C
      SUBROUTINE COMPG(ESTATE,G)
C     (* 451 - COMPUTE PARTIAL OF MEASUREMENT WITH RESPECT TO STATE *)
C
C     CALLS
C        (NONE)
C     CALLED BY
C        INCORP
C
C     INPUT
C        ESTATE
C     OUTPUT
C        G
C
      REAL ESTATE(6)
C        STATE ESTIMATE
      REAL G(3,6)
C        JACOBIAN
      INTEGER I,J
C        DO LOOP INDICES
C
C- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C
      (* COMPUTE G *)
         DO 10 I=1,3
            DO 10 J=1,6
               G(I,J)=0
   10    CONTINUE
         DO 20 I=1,3
            G(I,I)=1
   20    CONTINUE
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE ESTCOV(ESTATE,R)
C     (* 452 - ESTIMATE MEASUREMENT COVARIANCE *)
C
C     CALLS
C        <NONE>
C     CALLED BY
C        INCORP
C
C     INPUT
C        ESTATE
C     OUTPUT
C        R
C
      REAL ESTATE(6)
C        ESTIMATE OF STATE VECTOR
      INTEGER I,J
C        DO LOOP INDICES
      REAL R(3,3)
C        MEASUREMENT COVARIANCE
      REAL VARNCE
C        ESTIMATED VARIANCE (PER AXIS)
C- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      DO 10 I=1,3
         DO 10 J=1,3
            R(I,J)=0.0
   10 CONTINUE
      VARNCE=1.904E-7*(ESTATE(1)**2+ESTATE(2)**2+ESTATE(3)**2)**2+21.9
      DO 20 I=1,3
         R(I,I)=VARNCE
   20 CONTINUE
      RETURN
C
      END
```



```
C
      SUBROUTINE KALGAN(R,P,G,KGAIN)
C     (* 453 - COMPUTE KALMAN GAIN MATRIX *)
C
C     CALLS
C        MADD,MINV,MMLT
C     CALLED BY
C        INCORP
C
C     INPUT
C        R,P,G
C     OUTPUT
C        KGAIN
C
      INTEGER ERR
C        ERROR CODE (=0 IF NO ERROR)
      REAL G(3,6)
C        PARTIAL OF MEASUREMENT WITH RESPECT TO STATE
      REAL GT(6,3)
C        TRANSPOSE OF G
      INTEGER I,J
C        DO LOOP INDICES
      REAL KGAIN(6,3)
C        KALMAN GAIN MATRIX
      REAL P(6,6)
C        STATE ESTIMATE COVARIANCE
      REAL R(3,3)
C        MEASUREMENT COVARIANCE
      REAL SCRACH(4,6)
C        SCRATCHPAD FOR ROUTINE MINV
      REAL TEMP1(6,3),TEMP2(3,3),TEMP3(3,3)
C        INTERMEDIATE RESULTS WITH NO PROBLEM-RELATED SIGNIFICANCE
      INTEGER TERMNL
C        FORTRAN UNIT NUMBER FOR TERMINAL
C
      COMMON/SIMUL/TERMNL,IDUMMY
C
C- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* COMPUTE KGAIN=P*TRN(G)*INV(R+G*P*TRN(G)) *)
         DO 10 I=1,3
            DO 10 J=1,6
               GT(J,I)=G(I,J)
   10    CONTINUE
         CALL MMLT(TEMP1,P,GT,6,6,3)
         CALL MMLT(TEMP2,G,TEMP1,3,6,3)
         CALL MADD(TEMP3,R,TEMP2,3,3)
         CALL MINV(TEMP2,TEMP3,3,SCRACH,4,6,ERR)
         IF(ERR.NE.0) GO TO 20
         CALL MMLT(TEMP1,GT,TEMP2,6,3,3)
         CALL MMLT(KGAIN,P,TEMP1,6,6,3)
         RETURN
C
   20 CONTINUE
C     (* REPORT ERROR *)
         WRITE(TERMNL,901)
         STOP
C
  901 FORMAT('MATRIX INVERSION FAILURE IN SUBROUTINE KALGAN')
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE UPDSTA(ESTATE,KGAIN,CVPOS)
C     (* 454 - UPDATE STATE ESTIMATE *)
C
C     CALLS
C        MMLT
C     CALLED BY
C        INCORP
C
C     INPUT
C        RELPOS,AT,ESTATE
C     OUTPUT
C        ESTATE
C
      REAL CVPOS(3)
C        MEASURED CHASE VEHICLE POSITION IN PRIMARY REFERENCE FRAME
      REAL DELTA(6)
C        ADJUSTMENT TO ESTATE
      REAL ESTATE(6)
C        ESTIMATE OF STATE VECTOR
      INTEGER I
C        DO LOOP INDEX
      REAL KGAIN(6,3)
C        KALMAN GAIN MATRIX
      REAL POSERR(3)
C        MEASURED POSITION MINUS PREDICTED POSITION
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* COMPUTE ESTATE=ESTATE+KGAIN*(CVPOS-PREDICTED POSITION) *)
      DO 10 I=1,3
         POSERR(I)=CVPOS(I)-ESTATE(I)
   10 CONTINUE
      CALL MMLT(DELTA,KGAIN,POSERR,6,3,1)
      DO 20 I=1,6
         ESTATE(I)=ESTATE(I)+DELTA(I)
   20 CONTINUE
      RETURN
C
      END
```



```
C
      SUBROUTINE UPDCOV(P,KGAIN,G)
C     (* 455 - UPDATE COVARIANCE MATRIX FROM (P=(I-K*G)*P)  *)
C
C     CALLS
C        MMLT
C     CALLED BY
C        INCORP
C
C     INPUT
C        P,KGAIN,G
C     OUTPUT
C        P
C
      REAL G(3,6)
C        PARTIAL OF MEASUREMENT WITH RESPECT TO STATE
      INTEGER I,J
C        DO LOOP INDICES
      REAL KGAIN(6,3)
C        KALMAN GAIN MATRIX
      REAL P(6,6)
C        COVARIANCE OF STATE ESTIMATE
      REAL TEMP(6,6),P1(6,6)
C        INTERMEDIATE RESULTS WITH NO PROBLEM-RELATED SIGNIFICANCE
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* COMPUTE I-KGAIN*G *)
      CALL MMLT(TEMP,KGAIN,G,6,3,6)
      DO 10 I=1,6
         DO 10 J=1,6
            TEMP(I,J)=-TEMP(I,J)
   10 CONTINUE
      DO 20 I=1,6
         TEMP(I,I)=TEMP(I,I)+1.0
   20 CONTINUE
C     (* COMPUTE NEW P *)
      CALL MMLT(P1,TEMP,P,6,6,6)
      DO 30 I=1,6
         DO 30 J=1,6
            P(I,J)=P1(I,J)
   30 CONTINUE
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE PROPES(P,ESTATE,F,STEP,ACVT)
C     (* 460 - PROPAGATE STATE ESTIMATE AND COVARIANCE *)
C
C     CALLS
C        MMLT,MSCL
C     CALLED BY
C        DOCK
C
C     INPUT
C        P,ESTATE,ACVT,F,STEP
C     OUTPUT
C        P,ESTATE
C
      REAL ACCEL(3)
C        ESTIMATED ACCELERATION
      REAL ACVT(3,3)
C        TRANSPOSE OF CHASE VEHICLE DIRECTION COSINE MATRIX
      REAL AK2(3,14)
C        MATRIX RELATING FORCES FROM INDIVIDUAL THRUSTERS TO NET FORCE VECTOR
      REAL AMAGSQ
C        SQUARE OF MAGNITUDE OF TOTAL ACCELERATION
      REAL AVAR
C        ESTIMATED ACCELERATION VARIANCE
      REAL DSTATE(6)
C        CHANGE IN STATE
      REAL DP(6,6)
C        CHANGE IN COVARIANCE
      REAL ESTATE(6)
C        ESTIMATED STATE
      REAL F(14)
C        FORCES FROM THRUSTERS AFTER SELECTION
      REAL FBOD(3)
C        FORCES ALONG CHASE VEHICLE BODY AXES
      REAL FORCE(3)
C        FORCES IN PRIMARY REFERENCE FRAME
      INTEGER I,J
C        DO LOOP INDICES
      REAL P(6,6)
C        COVARIANCE OF STATE ESTIMATE
      REAL STEP
C        INTEGRATION TIME STEP
      PARAMETER AVGMAS=3650
C        ESTIMATED CHASE VEHICLE MASS
C
      COMMON/VEHIC/DUMMY1(14),AK2,DUMMY2(57)
```



```
C
C     (* COMPUTE FORCE=ACVT*AK2*F *)
      CALL MMLT(FBOD,AK2,F,3,14,1)
      CALL MMLT(FORCE,ACVT,FBOD,3,3,1)
C     (* ESTIMATE ACCELERATION *)
      CALL MSCL(ACCEL,FORCE,3,1,1./AVGMAS)
C     (* ESTIMATE ACCELERATION VARIANCE *)
      AMAGSQ=ACCEL(1)**2+ACCEL(2)**2+ACCEL(3)**2
      AVAR=0.05*AMAGSQ+1.E-7
C     (* FIND POSITION CHANGE *)
      DO 10 I=1,3
         DSTATE(I)=(ESTATE(I+3)+.5*ACCEL(I)*STEP)*STEP
10    CONTINUE
C     (* FIND VELOCITY CHANGE *)
      DO 20 I=4,6
         DSTATE(I)=STEP*ACCEL(I-3)
20    CONTINUE
C     (* FIND COVARIANCE CHANGE *)
      DO 30 I=4,6
         DO 30 J=1,6
            DP(I,J)=0.0
30    CONTINUE
      DP(4,4)=AVAR
      DP(5,5)=AVAR
      DP(6,6)=AVAR
      DO 40 I=1,3
         DO 40 J=1,6
            DP(I,J)=P(I+3,J)
40    CONTINUE
      DO 50 I=1,6
         DO 50 J=1,3
            DP(I,J)=DP(I,J)+P(I,J+3)
50    CONTINUE
C     (* COMPUTE NEW STATE AND COVARIANCE *)
      DO 60 I=1,6
         ESTATE(I)=ESTATE(I)+DSTATE(I)
         DO 60 J=1,6
            P(I,J)=P(I,J)+DP(I,J)
60    CONTINUE
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE SETGOL(ESTATE,P,T,ACV,AT,GXL,GXH,GYL,GYH,GZL,GZH,
     A   DEDLIN,VALID)
C     (* 470 - SET GOAL *)
C
C     CALLS
C        SQR1,AMAX1,MTRN,MMLT,MSUB
C     CALLED BY
C        DOCK
C
C     INPUT
C        ESTATE,P,DOKRAD,T,ACV,AT
C     OUTPUT
C        GXL,GXH,GYL,GYH,GZL,GZH,DEDLIN
C
      REAL ACV(3,3),AT(3,3),TRNAT(3,3)
C        DIRECTION COSINE MATRICES FOR CHASE VEHICLE & TARGET SPACECRAFT
C        AND TRANSPOSE OF TARGET DIRECTION COSINE MATRIX
      REAL D
C        3-SIGMA POSITION UNCERTAINTY
      REAL DEDLIN
C        TIME BY WHICH STRATEGY LOGIC WANTS CHASE VEHICLE IN GOAL 'BOX'
      REAL DOKRAD
C        RADIUS FROM TARGET SPACECRAFT WITHIN WHICH THE SPACECRAFT ARE
C        CONSIDERED DOCKED
      REAL ESTATE(6)
C        ESTIMATE OF STATE VECTOR
      REAL GXL,GXH,GYL,GYH,GZL,GZH
C        X,Y,Z LOWER AND UPPER LIMITS OF GOAL 'BOX'
      REAL HDC(3),HDT(3)
C        DOCKING FIXTURES' POSITIONS IN THEIR RESPECTIVE SPACECRAFT
C        COORDINATE SYSTEMS
      INTEGER I,J
C        DO LOOP INDICES
      REAL P(6,6)
C        COVARIANCE OF STATE ESTIMATE
      REAL SLOP
C        ALLOWED ERROR IN POSITION (PER AXIS)
      REAL T
C        ELAPSED TIME
      REAL V1(3),V2(3),V3(3),R(3),V4(3),V5(3)
C        INTERMEDIATE RESULTS
      LOGICAL VALID
C        TRUE IF LAST MEASUREMENT WAS VALID
C
      COMMON/VEHIC/DUMMY1(98),DOKRAD,DUMMY2(14)
      COMMON/HNOOFF/DUMMY3(6),HDC,HDT
```



```
C
      IF(SQR1(ESTATE(1)**2+ESTATE(2)**2+ESTATE(3)**2) LT 8    AND
     A   SQR1(ESTATE(4)**2+ESTATE(5)**2+ESTATE(6)**2) LT  1)
     B   GO TO 10
         D=3 *SQR1(P(1,1)+P(2,2)+P(3,3))
         GO TO 20
   10 CONTINUE
         D=0
   20 CONTINUE
C     (* COMPUTE AIM POINT ON TARGET X AXIS *)
         R(1)=-D+HDT(1)
         R(2)=0 +HDT(2)
         R(3)=0 +HDT(3)
         CALL MMLT(V4,AT,ESTATE,3,3,1)
         IF(V4(2)**2+V4(3)**2 GT 225 0  OR   NOT  VALID) R(1)=-D-20
         CALL MMLT(V5,R,HDC,3,1)
C     (* CONVERT COORDINATES AND SUBTRACT CURRENT POSITION *)
         CALL MTRN(TRNAT,AT,3)
         CALL MMLT(V1,TRNAT,V5,3,3,1)
         CALL MSUB(V2,V1,ESTATE,3,1)
         CALL MMLT(V3,ACV,V2,3,3,1)
      SLOP=0 25*AMIN1( 1*ABS(V4(1)), 25)
      GXH=V3(1)
      IF(GXH LT 8 ) GXH=GXH+SLOP
      GYH=V3(2)+SLOP
      GZH=V3(3)+SLOP
      GXL=GXH-2 0*SLOP
      GYL=V3(2)-SLOP
      GZL=V3(3)-SLOP
      DEDLIN=T+ 125*SQRT(V3(1)**2+V3(2)**2+V3(3)**2)
      RETURN
C
      END
```

ORIGINAL PAGE IS OF POOR QUALITY


```
C
      SUBROUTINE THRUST(F,ESTATE,GXL,GXH,GYL,GYH,GZL,GZH,TLIM,ACV,
     A   ATRATE,RPYERR)
C     (* 480 -  SELECT THRUSTERS *)
C
C     CALLS
C        CNTLAW,DEFINF,FIRTHR,SELECT
C     CALLED BY
C        DOCK
C
C     INPUT
C        ESTATE GXL,     ,GZH,TLIM,ESTATE,ACV,ATRATE,RPYERR
C     OUTPUT
C        F
C
      REAL ACV(3,3)
C        MEASURED CHASE VEHICLE ATTITUDE IN PRIMARY REF FRAME
      REAL ATRATE(3)
C        MEASURED ATTITUDE RATES OF CHASE VEHICLE
      REAL E(14)
C        THRUSTER COMMAND ENTRIES FROM THRUSTER SELECT LOGIC
      REAL ESTATE(6)
C        STATE ESTIMATE
      REAL F(14)
C        FORCES FROM THRUSTERS AFTER SELECTION
      REAL F1(14)
C        LIST OF MAXIMUM THRUSTER FORCES
      REAL GXL,GXH,GYL,GYH,GZL,GZH
C        LOWER AND UPPER LIMITS OF GOAL 'BOX'
      INTEGER I
C        DO LOOP INDEX
      REAL ROTCMD(3),XLTCMD(3)
C        ROTATIONAL AND TRANSLATIONAL COMMANDS FROM CONTROL LAW
      REAL RPYERR(3)
C        MEASURED ATTITUDE ERROR IN ROLL, PITCH, AND YAW (RADIANS)
      REAL TLIM
C        TIME LIMIT
C
      COMMON/VEHIC/DUMMY(99),F1
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* 481 - USE CONTROL LAW TO DETERMINE NEEDED THRUST *)
      CALL CNTLAW(ROTCMD,XLTCMD,ESTATE,ACV,TLIM,GXL,GXH,GYL,GYH,GZL,
     A       GZH,ATRATE,RPYERR)
C     (* 482 - SELECT THRUSTER SET TO GIVE NEEDED THRUST *)
      CALL SELECT(ROTCMD,XLTCMD,E)
      DO 10 I=1,14
         F(I)=E(I)*F1(I)
   10 CONTINUE
C     (* 483 - FIRE THRUSTERS *)
      CALL FIRTHR(E)
      RETURN
C
      END
```



```
C
      SUBROUTINE CNTLAW(ROTCMD,XLTCMD,ESTATE,ACV,TLIM,GXL,GXH,GYL,GYH,
     A   GZL,GZH,ATRATE,RPYERR)
C     (* 481 - CONTROL LAW *)
C
C     CALLS
C        MMLT,ACCEL
C     CALLED BY
C        THRUST
C
C     INPUT
C        ESTATE,ACV,TLIM,GXL,     ,GZH,ATRATE,RPYERR
C     OUTPUT
C        ROTCMD,XLTCMD
C
      REAL ACV(3,3)
C        MEASURED CHASE VEHICLE ATTITUDE IN PRIMARY REFERENCE FRAME
      REAL ATRATE(3)
C        MEASURED CHASE VEHICLE ATTITUDE RATES
      REAL ESTATE(6)
C        STATE ESTIMATE
      REAL GXL,GXH,GYL,GYH,GZL,GZH
C        LOWER AND UPPER LIMITS OF GOAL 'BOX'
      REAL ROTCMD(3),XLTCMD(3)
C        ROTATIONAL AND TRANSLATIONAL COMMANDS FROM CONTROL LAW
      REAL RPYERR(3)
C        MEASURED ATTITUDE ERROR IN ROLL, PITCH, AND YAW
      REAL TLIM
C        TIME LIMIT
      REAL VBOD(3)
C        VELOCITY IN CHASE VEHICLE BODY COORDINATE SYSTEM
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* CONVERT VELOCITY TO BODY COORD. SYS. *)
      CALL MMLT(VBOD,ACV,ESTATE(4),3,3,1)
C     (* 481 1 - FIND IDEAL ACCELERATIONS *)
      XLTCMD(1)=ACCEL(VBOD(1),GXL,GXH,TLIM,1.233333,.03,.1)
      XLTCMD(2)=ACCEL(VBOD(2),GYL,GYH,TLIM,1.233333,.03,.1)
      XLTCMD(3)=ACCEL(VBOD(3),GZL,GZH,TLIM,1.233333,.03,.1)
      ROTCMD(1)=ACCEL(ATRATE(1),(-RPYERR(1)-.02),(-RPYERR(1)+.02),
     A       1.233333,1.233333,.02,.074)
      ROTCMD(2)=ACCEL(ATRATE(2),-RPYERR(2)-.02,-RPYERR(2)+.02,
     A       1.233333,1.233333,.02,.074)
      ROTCMD(3)=ACCEL(ATRATE(3),-RPYERR(3)-.02,-RPYERR(3)+.02,TLIM,
     A       1.233333,.02,.074)
      RETURN
C
      END
```

```
C
      FUNCTION ACCEL(XDOT, XMIN, XMAX, TLIM, DT, AMIN, AMAX)
C     (* 481 1 - ESTIMATE ACCELERATION FOR ONE AXIS *)
C
C     CALLS
C        (NONE)
C     CALLED BY
C        CNTLAW
C
C     INPUT
C        X, XDOT, XMIN, XMAX, TLIM, DT
C     OUTPUT
C        (FUNCTION VALUE ONLY)
C
      REAL AMIN, AMAX
C         MIN AND MAX RELATIVE THRUST
      REAL DT
C         TIME STEP BETWEEN DECISIONS
      REAL K1, K2, K3, K4
C         CONSTANTS BASED ON MIN, MAX POSSIBLE ACCELERATION
      REAL TLIM
C         TIME LIMIT - SECONDS UNTIL DEADLINE
      REAL XDOT
C         VELOCITY IN THIS AXIS
      REAL XMIN, XMAX
C         MIN, MAX ACCEPTABLE FINAL X VALUE
```



```
C
C
      K1= 5*AMAX*DT**2
      K2=AMAX*DT
      K3= 5/AMIN
      K4=AMIN* 5
      IF(XMAX GE 0 ) GO TO 50
         IF(XDOT LT 0 ) GO TO 10
            ACCEL=-1
            RETURN
 10      IF(XDOT*DT-K3*XDOT**2 GE XMIN) GO TO 20
            ACCEL=1
            RETURN
 20      IF(XDOT*DT-K1-K3*(K2-XDOT)**2 GE XMIN) GO TO 30
            ACCEL=0
            RETURN
 30      IF(XDOT*TLIM-K4*(TLIM-DT)**2 GT XMAX) GO TO 40
            ACCEL=0
            RETURN
 40      CONTINUE
            ACCEL=-1
            RETURN
 50   IF(XMIN LE 0 ) GO TO 100
         IF(XDOT GT 0 ) GO TO 60
            ACCEL=1
            RETURN
 60      IF(XDOT*DT+K3*XDOT**2 LE XMAX) GO TO 70
            ACCEL=-1
            RETURN
 70      IF(XDOT*DT+K1+K3*(K2+XDOT)**2 LE XMAX) GO TO 80
            ACCEL=0
            RETURN
 80      IF(XDOT*DT+K4*(TLIM-DT)**2 LT XMIN) GO TO 90
            ACCEL=0
            RETURN
 90      CONTINUE
            ACCEL=1
            RETURN
100   IF(XDOT LE 0 ) GO TO 120
         IF(XDOT*DT+K3*XDOT**2 LE XMAX) GO TO 110
            ACCEL=-1
            RETURN
110      CONTINUE
            ACCEL=0
            RETURN
120   IF(XDOT*DT-K3*XDOT**2 GE XMIN) GO TO 130
         ACCEL=1
         RETURN
130   CONTINUE
         ACCEL=0
         RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
      SUBROUTINE SELECT(ROTCMD,XLTCMD,E)
C     (* 482 - FROM CONTROL LAW OUTPUTS SELECT WHICH THRUSTERS TO FIRE *)
C
C     CALLS
C        TABLE1, TABLE2, TABLE3
C     CALLED BY
C        THRUST
C
C     INPUT
C        ROTCMD, XLTCMD
C     OUTPUT
C        E
C
      REAL E(14)
C        THRUSTER COMMAND ENTRIES FROM THRUSTER SELECT LOGIC
      REAL ROTCMD(3),XLTCMD(3)
C        ROTATIONAL AND TRANSLATIONAL COMMANDS FROM CONTROL LAW
      INTEGER I,J
C        DO LOOP INDICES
      INTEGER INDEX1,INDEX2,INDEX3
C        INDICES FOR COMMAND TABLES
      INTEGER ROTCOD(3),TRLCOD(3)
C        ROTATION AND TRANSLATION CODES USED TO COMPUTE INDICES
C        FOR TABLES
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* DETERMINE CODE FOR THRUSTER ACTIVATION *)
      DO 40 I=1,3
         IF(XLTCMD(I)) 1 ,20,30
   10    TRLCOD(I)=1
         GO TO 40
   20    TRLCOD(I)=0
         GO TO 40
   30    TRLCOD(I)=2
   40 CONTINUE
      DO 80 I=1,3
         IF(ROTCMD(I)) 50,60,70
   50    ROTCOD(I)=1
         GO TO 80
   60    ROTCOD(I)=0
         GO TO 80
   70    ROTCOD(I)=2
   80 CONTINUE
C     (* ZERO OUT THRUSTER COMMANDS *)
      DO 90 J=1,14
         E(J)=0.0
   90 CONTINUE
C     (* DETERMINE WHICH TABLE TO CONSULT FOR THRUSTER COMMANDS *)
      INDEX1=TRLCOD(1)+3*ROTCOD(2)+9*ROTCOD(3)
      INDEX2=TRLCOD(2)+3*ROTCOD(1)+9*ROTCOD(3)
      INDEX3=TRLCOD(3)+3*ROTCOD(1)+9*ROTCOD(2)
C     (* 482.1 THRU 482.3 - SELECT THRUSTER SET *)
      IF(INDEX1.EQ.0) GO TO 100
         CALL TABLE1(INDEX1,E)
  100 CONTINUE
      IF(INDEX2.EQ.0) GO TO 110
         CALL TABLE2(INDEX2,E)
  110 CONTINUE
      IF(INDEX3.EQ.0) GO TO 120
         CALL TABLE3(INDEX3,E)
  120 CONTINUE
      RETURN
C
      END
```



```
C
      SUBROUTINE TABLE1(INDEX1,E)
C     (* 482.1 -  FIND COMINATION OF TABLE ENTRIES THAT SATISFY THRUSTER
C        SELECTION REQUIREMENTS *)
C
C     CALLS
C        <NONE>
C     CALLED BY
C        SELECT
C
C     INPUT
C        INDEX1
C     OUTPUT
C        E
C
      REAL E(14)
C        THRUSTER COMMAND ENTRIES FROM THRUSTER SELECT LOGIC
      INTEGER INDEX1
C        THRUSTER TABLE INDEX
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* DO SELECTION BASED ON INDEX *)
      GOTO(10,20,30,40,50,60,70,80,90,100,110,120,130,140,150,160,170,
     A     180,190,200,210,220,230,240,250,260), INDEX1
   10 E(1)=1.0
      GO TO 300
   20 E(14)=1.0
      GO TO 300
   30 E(10)=1.0
      E(13)=1.0
      GO TO 300
   40 E(1)=1.0
      E(10)=1.0
      E(13)=1.0
      GO TO 300
   50 E(10)=1.0
      E(13)=1.0
      E(14)=1.0
      GO TO 300
   60 E(11)=1.0
      E(12)=1.0
      GO TO 300
   70 E(1)=1.0
      E(11)=1.0
      E(12)=1.0
      GO TO 300
   80 E(11)=1.0
      E(12)=1.0
      E(14)=1.0
      GO TO 300
   90 E(2)=1.0
      E(4)=1.0
      GO TO 300
  100 E(1)=1.0
      E(2)=1.0
      E(4)=1.0
      GO TO 300
  110 E(2)=1.0
      E(4)=1.0
      E(14)=1.0
      GO TO 300
  120 E(2)=1.0
      E(4)=1.0
      E(10)=1.0
      E(13)=1.0
      GO TO 300
  130 E(1)=1.0
      E(2)=1.0
      E(4)=1.0
      GO TO 300
  140 E(2)=1.0
      E(4)=1.0
      E(10)=1.0
      E(13)=1.0
      E(14)=1.0
      GO TO 300
```

```
150 E(2)=1.0
    E(4)=1.0
    E(11)=1.0
    E(12)=1.0
    GO TO 300
160 E(1)=1.0
    E(2)=1.0
    E(4)=1.0
    E(11)=1.0
    E(12)=1.0
    GO TO 300
170 E(2)=1.0
    E(4)=1.0
    E(11)=1.0
    E(12)=1.0
    E(14)=1.0
    GO TO 300
180 E(3)=1.0
    E(5)=1.0
    GO TO 300
190 E(1)=1.0
    E(3)=1.0
    E(5)=1.0
    GO TO 300
200 E(3)=1.0
    E(5)=1.0
    E(14)=1.0
    GO TO 300
210 E(3)=1.0
    E(5)=1.0
    E(10)=1.0
    E(13)=1.0
    GO TO 300
220 E(1)=1.0
    E(3)=1.0
    E(5)=1.0
    E(10)=1.0
    E(13)=1.0
    GO TO 300
230 E(3)=1.0
    E(5)=1.0
    E(10)=1.0
    E(13)=1.0
    E(14)=1.0
    GO TO 300
240 E(3)=1.0
    E(5)=1.0
    E(11)=1.0
    E(12)=1.0
    GO TO 300
250 E(1)=1.0
    E(3)=1.0
    E(5)=1.0
    E(11)=1.0
    E(12)=1.0
    GO TO 300
260 E(3)=1.0
    E(5)=1.0
    E(11)=1.0
    E(12)=1.0
    E(14)=1.0
300 RETURN
C
    END
```



```
C
      SUBROUTINE TABLE2(INDEX2,E)
C     (* 4B2 2 -  FIND COMINATION OF TABLE ENTRIES THAT SATISFY THRUSTER
C        SELECTION REQUIREMENTS *)
C
C     CALLS
C        <NONE>
C     CALLED BY
C        SELECT
C
C     INPUT
C        INDEX2
C     OUTPUT
C        E
C
      REAL E(14)
C        THRUSTER COMMAND ENTRIES FROM THRUSTER SELECT LOGIC
      INTEGER INDEX2
C        THRUSTER TABLE INDEX
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* DO SELECTION BASED ON INDEX *)
      GOTO(10,20,30,40,50,60,70,80,90,100,110,120,130,140,150,160,170,
     A     180,190,200,210,220,230,240,250,260),INDEX2
   10 E(4)=1.0
      E(5)=1.0
      GO TO 300
   20 E(2)=1.0
      E(3)=1.0
      GO TO 300
   30 E(7)=1.0
      E(8)=1.0
      GO TO 300
   40 E(4)=1.0
      E(5)=1.0
      E(7)=1.0
      E(8)=1.0
      GO TO 300
   50 E(2)=1.0
      E(3)=1.0
      E(7)=1.0
      E(8)=1.0
      GO TO 300
   60 E(6)=1.0
      E(9)=1.0
      GO TO 300
   70 E(4)=1.0
      E(5)=1.0
      E(6)=1.0
      E(9)=1.0
      GO TO 300
   80 E(2)=1.0
      E(3)=1.0
      E(6)=1.0
      E(9)=1.0
      GO TO 300
   90 E(2)=1.0
      E(4)=1.0
      GO TO 300
  100 E(2)=1.0
      E(4)=1.0
      E(6)=1.0
      E(8)=1.0
      GO TO 300
  110 E(2)=1.0
      E(4)=1.0
      E(7)=1.0
      E(9)=1.0
      GO TO 300
  120 E(2)=1.0
      E(4)=1.0
      E(7)=1.0
      E(8)=1.0
      GO TO 300
  130 E(4)=1.0
      E(8)=1.0
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
      GO TO 300
140   E(2)=1.0
      E(7)=1.0
      GO TO 300
150   E(2)=1.0
      E(4)=1.0
      E(6)=1.0
      E(9)=1.0
      GO TO 300
160   E(4)=1.0
      E(6)=1.0
      GO TO 300
170   E(2)=1.0
      E(9)=1.0
      GO TO 300
180   E(3)=1.0
      E(5)=1.0
      GO TO 300
190   E(3)=1.0
      E(5)=1.0
      E(6)=1.0
      E(8)=1.0
      GO TO 300
200   E(3)=1.0
      E(5)=1.0
      E(7)=1.0
      E(9)=1.0
      GO TO 300
210   E(3)=1.0
      E(5)=1.0
      E(7)=1.0
      E(8)=1.0
      GO TO 300
220   E(5)=1.0
      E(8)=1.0
      GO TO 300
230   E(3)=1.0
      E(7)=1.0
      GO TO 300
240   E(3)=1.0
      E(5)=1.0
      E(6)=1.0
      E(9)=1.0
      GO TO 300
250   E(5)=1.0
      E(6)=1.0
      GO TO 300
260   E(3)=1.0
      E(9)=1.0
300   RETURN
C
      END
```



```
C
      SUBROUTINE TABLE3(INDEX3,E)
C     (* 482 3 -  FIND COMINATION OF TABLE ENTRIES THAT SATISFY THRUSTER
C        SELECTION REQUIREMENTS *)
C
C     CALLS
C        <NONE>
C     CALLED BY
C        SELECT
C
C     INPUT
C        INDEX3
C     OUTPUT
C        E
C
      REAL E(14)
C        COMMAND ENTRIES FROM THRUSTER SELECT LOG
      INTEGER INDEX3
C        THRUSTER TABLE INDEX
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* DO SELECTION BASED ON INDEX *)
      GOTO(10,20,30,40,50,60,70,80,90,100,110,120,130,140,150,160,170,
     A 180,190,200,210,220,230,240,250,260), INDEX3
10    E(10)=1.0
      E(11)=1.0
      GO TO 300
20    E(12)=1.0
      E(13)=1.0
      GO TO 300
30    E(7)=1.0
      E(8)=1.0
      GO TO 300
40    E(7)=1.0
      E(8)=1.0
      E(10)=1.0
      E(11)=1.0
      GO TO 300
50    E(7)=1.0
      E(8)=1.0
      GO TO 300
60    E(6)=1.0
      E(9)=1.0
      GO TO 300
70    E(6)=1.0
      E(9)=1.0
      E(10)=1.0
      E(11)=1.0
      GO TO 300
80    E(6)=1.0
      E(9)=1.0
      E(12)=1.0
      E(13)=1.0
      GO TO 300
90    E(10)=1.0
      E(13)=1.0
      GO TO 300
100   E(10)=1.0
      GO TO 300
110   E(13)=1.0
      GO TO 300
120   E(7)=1.0
      E(8)=1.0
      E(10)=1.0
      E(13)=1.0
      GO TO 300
130   E(7)=1.0
      E(7)=1.0
      E(10)=1.0
      GO TO 300
140   E(7)=1.0
      E(8)=1.0
      E(13)=1.0
      GO TO 300
150   E(6)=1.0
      E(9)=1.0
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
      E(10)=1.0
      E(13)=1.0
      GO TO 300
160   E(6)=1.0
      E(9)=1.0
      E(10)=1.0
      GO TO 300
170   E(6)=1.0
      E(9)=1.0
      E(13)=1.0
      GO TO 300
180   E(11)=1.0
      E(12)=1.0
      GO TO 300
190   E(11)=1.0
      GO TO 300
200   E(12)=1.0
      GO TO 300
210   E(7)=1.0
      E(8)=1.0
      E(11)=1.0
      E(12)=1.0
      GO TO 300
220   E(7)=1.0
      E(8)=1.0
      E(11)=1.0
      GO TO 300
230   E(7)=1.0
      E(8)=1.0
      E(12)=1.0
      GO TO 300
240   E(6)=1.0
      E(9)=1.0
      E(11)=1.0
      E(12)=1.0
      GO TO 300
250   E(6)=1.0
      E(9)=1.0
      E(12)=1.0
      GO TO 300
260   E(6)=1.0
      E(9)=1.0
      E(12)=1.0
300   RETURN
C
      END
```



```
C
      SUBROUTINE FIRTHR(E)
C     (* 483 - PUT THRUSTER COMMANDS IN COMMAND PORTS *)
C
C     CALLS
C        <NONE>
C     CALLED BY
C        THRUST
C
C     INPUT
C        E
C     OUTPUT
C        <NONE>
C
      REAL E(14)
C        THRUSTER COMMAND ENTRIES FROM THRUSTER SELECT LOGIC
C
C     (* TEMPORARY DUMMY SUBROUTINE *)
      RETURN
C
      END
```

```
C
      SUBROUTINE IMU(STATE,ACV,ACVT,ATRATE)
C     (* 490 - SIMULATE IMU MEASUREMENT *)
C
C     CALLS
C        DIRMAT,MADD,MSCL,ANGVEC
C     CALLED BY
C        DOCK
C
C     INPUT
C        QO,STATE,FULDIS,INERCV,MEMPTY
C     OUTPUT
C        ACV,ACVT,ATRATE
C
      REAL ACV(3,3),ACVT(3,3)
C        DIRECTION COSINE MATRIX AND ITS TRANSPOSE FOR 'MEASURED' ATTITUDE
C        OF CHASE VEHICLE WITH RESPECT TO PRIMARY REFERENCE FRAME
      REAL ATRATE(3)
C        MEASURED ATTITUDE RATES ABOUT CHASE VEHICLE AXES
      REAL FULDIS(3,3)
C        TENSOR DESCRIBING FUEL DISTRIBUTION
      REAL INERCV(3,3)
C        TENSOR DESCRIBING INERTIA OF EMPTY CHASE VEHICLE
      REAL INERFL(3,3)
C        FUEL INERTIA TENSOR
      REAL INERTA(3,3)
C        INERTIA OF CHASE VEHICLE (LOADED)
      REAL MEMPTY
C        MASS OF EMPTY CHASE VEHICLE
      REAL QO(4),QP(4)
C        INITIAL CHASE VEHICLE ATTITUDE QUATERNION IN 'TRUTH' REFERENCE FRAME
C        AND CURRENT PRIMARY REFERENCE FRAME, RESPECTIVELY
      REAL STATE(14)
C        TRUE CHASE VEHICLE STATE
      REAL TACV(3,3),TACVT(3,3)
C        CHASE VEHICLE DIRECTION COSINE MATRIX WITH RESPECT TO TRUTH
C        COORDINATE SYSTEM AND INVERSE OF THIS MATRIX
C
      COMMON/QIF/QO
      COMMON/MAEPRP/FULDIS,INERCV,MEMPTY
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* SUBTRACT INITIAL ATTITUDE FROM CURRENT ATTITUDE TO GET CURRENT
C     ATTITUDE IN PRIMARY REF. FRAME *)
      QP(1)=QO(4)*STATE(10)+QO(3)*STATE(11)-QO(2)*STATE(12)-
     A   QO(1)*STATE(13)
      QP(2)=-QO(3)*STATE(10)+QO(4)*STATE(11)+QO(1)*STATE(12)-
     A   QO(2)*STATE(13)
      QP(3)=QO(2)*STATE(10)-QO(1)*STATE(11)+QO(4)*STATE(12)-
     A   QO(3)*STATE(13)
      QP(4)=QO(1)*STATE(10)+QO(2)*STATE(11)+QO(3)*STATE(12)+
     A   QO(4)*STATE(13)
C     (* 9O, - CONVERT QUATERNIONS TO DIRECTION COSINE MATRICES *)
      CALL DIRMAT(QP,ACV,ACVT)
      CALL DIRMAT(STATE(10),TACV,TACVT)
C     (* DETERMINE INERTIA *)
      CALL MSCL(INERFL,FULDIS,3,3,STATE(14)-MEMPTY)
      CALL MADD(INERTA,INERCV,INERFL,3,3)
C     (* 9O4 - FIND ANGULAR VELOCITY VECTOR *)
      CALL ANGVEC(INERTA,STATE(7),ATRATE,TACV)
      RETURN
C
      END
```



```
C
      SUBROUTINE RPY(XC,YC,RPYERR)
C     (* 4AO - CALCULATE ROLL, PITCH, AND YAW ERRORS *)
C
C     CALLS
C        ATAN2
C     CALLED BY
C        DOCK
C
C     INPUT
C        XC,YC
C     OUTPUT
C        RPYERR
C
      REAL RPYERR(3)
C        ROLL, PITCH AND YAW ERRORS (RADIANS)
      REAL XC,YC
C        COORDINATES OF CENTER OF ELLIPSE, IN FOCAL PLANE, IN METERS
C
      COMMON/OPTSYS/DUMMY1(2),FOCLEN
C
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* CALCULATE PITCH ERROR *)
      RPYERR(2)=ATAN2(YC,FOCLEN)
C     (* CALCULATE YAW ERROR *)
      RPYERR(3)=ATAN2(XC,FOCLEN)
C     (* CALCULATE ROLL ERROR *)
      RPYERR(1)=0
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE ESTRPY(ESTATE,ACVT,RPYERR)
C     (* 480 - ESTIMATE ATTITUDE ERROR FROM STATE ESTIMATE *)
C
C     CALLS
C        ATAN2,ABIN,MSCL
C     CALLED BY
C        DOCK
C
C     INPUT
C        ESTATE,ACVT
C     OUTPUT
C        RPYERR
C
      REAL ACVT(3,3)
C        TRANSPOSE OF CHASE VEHICLE DIRECTION COSINE MATRIX (WITH RESPECT
C        TO PRIMARY REFERENCE FRAME)
      REAL DOTPRD
C        DOT PRODUCT OF VECTORS DEFINING LINE TO TARGET AND VEHICLE X AXIS
      REAL ESTATE(6)
C        ESTIMATED STATE
      REAL PHI
C        EULER ERROR ANGLE
      REAL PRDMAG
C        MAGNITUDE OF CROSS PRODUCT OF VECTORS DEFINING LINE TO TARGET AND
C        VEHICLE X AXIS
      REAL Q(4)
C        ATTITUDE ERROR QUATERNION
      REAL R1(3),R2(3)
C        INTERMEDIATE RESULTS
      REAL RPYERR(3)
C        ERRORS IN ROLL  PITCH AND YAW, RESPECTIVELY (RADIANS)
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      R1(1)=-ESTATE(2)*ACVT(3,1)+ESTATE(3)*ACVT(2,1)
      R1(2)=-ESTATE(3)*ACVT(1,1)+ESTATE(1)*ACVT(3,1)
      R1(3)=-ESTATE(1)*ACVT(2,1)+ESTATE(2)*ACVT(1,1)
      PRDMAG=SQRT(R1(1)**2+R1(2)**2+R1(3)**2)
      DOTPRD=ESTATE(1)*ACVT(1,1)+ESTATE(2)*ACVT(2,1)+ESTATE(3)*
     A   ACVT(3,1)
      IF(PRDMAG GT 0 ) GO TO 30
         Q(1)=0
         Q(2)=0
         IF(DOTPRD GT 0 ) GO TO 10
            Q(3)=1
            Q(4)=0 0
            GO TO 20
 10      CONTINUE
            Q(3)=0 0
            Q(4)=1 0
 20      CONTINUE
         GO TO 40
 30   CONTINUE
         CALL MSCL(R2,R1,3 1,1 /PRDMAG)
         PHI=ATAN2(PRDMAG,DOTPRD)
         CALL MSCL(Q,R2,3,1,SIN(PHI* 5))
         Q(4)=COS(PHI* 5)
 40   CONTINUE
      RPYERR(1)=0
      RPYERR(2)=-ASIN(2 *(Q(1)*Q(3)-Q(2)*Q(4)))
      IF(ABS(RPYERR(2)) GT 1 57) GO TO 50
         RPYERR(3)=ATAN2(2 *(Q(1)*Q(2)+Q(3)*Q(4)),Q(1)**2-Q(2)**2
     A      -Q(3)**2+Q(4)**2)
         GO TO 60
 50   CONTINUE
         RPYERR(3)=0
 60   CONTINUE
      RETURN
C
      END
```



```
C
      SUBROUTINE DIRMAT(Q,A,TRNA)
C     (* 901 - COMPUTE A DIRECTION COSINE MATRIX FROM A QUATERNION *)
C
C     CALLS
C        MTRN
C     CALLED BY
C        ATITUD,IMU,LOCATE,QUATRN,STPRIM
C
C     INPUT
C        Q
C     OUTPUT
C        A,TRNA
C
      REAL A(3,3)
C        DIRECTION COSINE MATRIX FORMED FROM QUATERNION Q
      REAL TRNA(3,3)
C        TRANPOSE OF A
      REAL Q(4)
C        QUATERNION
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* COMPUTE ELEMENTS OF A *)
         A(1,1)=Q(1)**2-Q(2)**2-Q(3)**2+Q(4)**2
         A(1,2)=2 *(Q(1)*Q(2)+Q(3)*Q(4))
         A(1,3)=2 *(Q(1)*Q(3)-Q(2)*Q(4))
         A(2,1)=2 *(Q(1)*Q(2)-Q(3)*Q(4))
         A(2,2)=-Q(1)**2+Q(2)**2-Q(3)**2+Q(4)**2
         A(2,3)=2 *(Q(2)*Q(3)+Q(1)*Q(4))
         A(3,1)=2 *(Q(1)*Q(3)+Q(2)*Q(4))
         A(3,2)=2 *(Q(2)*Q(3)-Q(1)*Q(4))
         A(3,3)=-Q(1)**2-Q(2)**2+Q(3)**2+Q(4)**2
C     (* COMPUTE TRANSPOSE OF A *)
         CALL MTRN(TRNA,A,3)
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE STPRIM(STATE,F,INERTA,T,DSTATE)
C (* 902 - DETERMINE TIME DERIVATIVE OF STATE VECTOR *)
C
C     CALLS
C        ANGVEC,FORCE,LINACL,LPRIME,MAKROT,MPRIME,QPRIME,TORQUE,DIRMAT
C     CALLED BY
C        COMPK1,COMPK2,COMPK3,COMPK4
C
C     INPUT
C        F,T,STATE,INERTA
C     OUTPUT
C        DSTATE
C
      REAL A(3,3)
C        'TRUE' DIRECTION COSINE MATRIX FOR CHASE VEHICLE
      REAL BODVEL(3)
C        ANGULAR VELOCITY ABOUT AXES OF CHASE VEHICLE
      REAL DSTATE(14)
C        TIME DERIVATIVE OF STATE VECTOR
      REAL F(14)
C        FORCES FROM CHASE VEHICLE THRUSTERS
      REAL INERTA(3,3)
C        INERTIA OF CASE VEHICLE (INCL  FUEL)
      REAL N(3)
C        TORQUE ON SPACECRAFT ABOUT ITS CENTER OF MASS
      REAL NTFORC(3)
C        NET FORCE FROM THRUSTER OPERATION
      REAL OMEGA(4,4)
C        ROTATION MATRIX OMEGA, TO BE APPLIED TO QUATERNION MATRIX 'Q'
C        TO FORM TIME DERIVATIVE OF Q (Q=ELEMENTS 10-13 OF STATE)
      REAL STATE(14)
C        CHASE VEHICLE STATE VECTOR
      REAL T
C        ELAPSED TIME
      REAL TRNA(3,3)
C        TRANSPOSE OF 'A'
C
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C (* 901 - COMPUTE A,TRNA FROM QUATERNION  *)
      CALL DIRMAT(STATE(10),A,TRNA)
C (* 902 1 - DETERMINE NET FORCE FROM THRUSTERS IN  'RUTH' COORD SYS  *)
      CALL FORCE(F,NTFORC,TRNA)
C (* 902 2 - COMPUTE TIME DERIVATIVE OF SPACECRAFT MASS *)
      CALL MPRIME(F,DSTATE(14))
C (* 902 3 - COMPUTE LINEAR ACCELERATIONS *)
      CALL LINACL(T,STATE(14),NTFORC,STATE(1),DSTATE(4))
C (* 904 - FIND ANGULAR VELOCITY VECTOR *)
      CALL ANGVEC(INERTA,STATE(7),BODVEL,A)
C (* 902 4 - CALCULATE TORQUE ON SPACECRAFT FROM THRUSTER OPERATION *)
      CALL TORQUE(F,N,TRNA)
C (* 902 5 - COMPUTE TIME DERIVATIVE OF ANGULAR MOMENTUM VECTOR *)
      CALL LPRIME(N,BODVEL,STATE(7),DSTATE(7),TRNA)
C (* 902 6 - FORM ROTATION MATRIX OMEGA *)
      CALL MAKROT(BODVEL,OMEGA,A)
C (* 902 7 - CALCULATE THE TIME DERIVATIVE OF QUATERNION *)
      CALL QPRIME(STATE(10),OMEGA,DSTATE(10))
C (* DEFINE VELOCITY ELEMENTS OF DSTATE *)
      DSTATE(1)=STATE(4)
      DSTATE(2)=STATE(5)
      DSTATE(3)=STATE(6)
      RETURN
C
      END
```



```
C
      SUBROUTINE FORCE(F,NTFORC,TRNA)
C (* 902 1 -  COMPUTE NET FORCE FROM THRUSTER OPERATION USING EQUATION
C          NET FORCE= A TRANSPOSE * AK2 * F *)
C
C     CALLS
C        MMLT
C     CALLED BY
C        STPRIM
C
C     INPUT
C        F,AK2,Q
C     OUTPUT
C        NTFORC
C
      REAL AK2(3,14)
C        CONSTANT MATRIX RELATING INDIVIDUAL THRUSTER OUTPUTS
C        TO NET THRUST IN SPACECRAFT REFERENCE FRAME
C        (THIS ACCOMODATES MISALIGNMENT OF THRUSTERS)
      REAL B(3)
C        INTERMEDIATE RESULTS
      REAL F(14)
C        FORCE VECTOR OF CHASE VEHICLE AFTER THRUSTER SELECTION
      REAL NTFORC(3)
C        NET FORCE FROM THRUSTER OPERATION IN TRUTH COORD  SYS
      REAL TRNA(3,3)
C        TRANSPOSE OF DIRECTION COSINE MATRIX A
C
      COMMON/VEHIC/DUMMY1(14),AK2,DUMMY2(57)
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C (* COMPUTE NET FORCE *)
      CALL MMLT(B,AK2,F,3,14,1)
      CALL MMLT(NTFORC,TRNA,B,3,3,1)
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE MPRIME(F,DMDT)
      (* 902 2 -  COMPUTE TIME DERIVATIVE OF SPACECRAFT MASS *)
C
C     CALLS
C        MMLT
C     CALLED BY
C        STPRIM
C
C     INPUT
C        AK1,F
C     OUTPUT
C        DMDT
C
      REAL AK1(14)
C        CONSTANT VECTOR DERIVED FROM ENGINE SPECIFIC IMPULSE DATA
      REAL DMDT
C        TIME DERIVATIVE OF MASS
      REAL F(14)
C        FORCES FROM THRUSTERS AFTER SELECTION
C
      COMMON/VEHIC/AK1,DUMMY(99)
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* COMPUTE DMDT = AK1*F *)
         CALL MMLT(DMDT,AK1,F,1,14,1)
C     (* CHANGE SIGN   MASS LOSS IS NEGATIVE *)
         DMDT=-DMDT
      RETURN
C
      END
```



```
C
      SUBROUTINE LINACL(T,M,NTFORC,CURPOS,ACCEL)
      (* 902 3 -  COMPUTE LINEAR ACCELERATIONS IN X, Y, AND Z DIRECTIONS *)
C
C     CALLS
C        SIN,COS
C     CALLED BY
C        STPRIM
C
C     INPUT
C        T,M,NTFORC,CURPOS
C     OUTPUT
C        ACCEL
C
      REAL ACCEL(3)
C        SPACECRAFT ACCELERATIONS
      REAL AGGX,AGGY,AGGZ
C        ACCELERATIONS DUE TO GRAVITY GRADIANT
      REAL CURPOS(3)
C        CURRENT CHASE VEHICLE POSITION VECTOR
      REAL M
C        MASS OF SPACECRAFT IS M
      REAL NTFORC(3)
C        NET FORCE FROM THRUSTER OPERATION
      REAL SWT,CWT,A,B
C        THESE ARE INTERMEDIATE RESULTS
      REAL T
C        ELAPSED TIME
      REAL WO
C        RELATIVE ANGULAR ACCELERATION IN RADS/SEC
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* PRECOMPUTE VALUES USED SEVERAL PLACES *)
         WO=1 1629E-3
         A=2 67671E-6
         B=WO*T
         SWT=SIN(B)
         CWT=COS(B)
C     (* COMPUTE GRAVITY GRADIENT ACCELERATION *)
         AGGX=A*SWT*(CURPOS(1)*SWT+CURPOS(3)*CWT)
         AGGY=0 0
         AGGZ=A*CWT*(CURPOS(1)*SWT+CURPOS(3)*CWT)
C     (* COMPUTE NET ACCELERATION (GRAVITY PLUS THRUSTERS) *)
         ACCEL(1)=AGGX NTFORC(1)/M
         ACCEL(2)=AGGY+NTFORC(2)/M
         ACCEL(3)=AGGZ+NTFORC(3)/M
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

C-3

```
C
      SUBROUTINE TORQUE(F,N,TRNA)
C     (* 902 4 - CALCULATE TORQUE (N) *)
C
C     CALLS
C        MMLT MADD MSCL
C     CALLED BY
C        STPRIM
C
C     INPUT
C        AK3,F,TRNA
C     OUTPUT
C        N
C
      REAL AK3(3,14)
C        MATRIX RELATING THRUSTER INPUTS TO TORQUES IN 'CV' REF  FRAME
      REAL F(14)
C        FORCES FROM THRUSTERS AFTER SELECTION
      REAL N(3)
C        TORQUE ON SPACECRAFT ABOUT ITS CENTER OF MASS IN TRUTH COORD  SYS
      REAL N1(3)
C        TORQUE IN C V  REFERENCE FRAME
      REAL TRNA(3,3)
C        TRANSPOSE OF DIRECTION COSINE MATRIX THAT GIVES C V  ATTITUDE
C
      COMMON/VEHIC/DUMMY1(56),AK3,DUMMY2(15)
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* CALCULATE N=TRNA*AK3*F *)
         CALL MMLT(N1,AK3,F,3,14,1)
         CALL MMLT(N,TRNA,N1,3,3,1)
      RETURN
C
      END
```



```
C
      SUBROUTINE LPRIME(N,BODVEL,ANGMNT,DLDT,TRNA)
C     (* 902 5 - COMPUTE TIME DERIVATIVE OF ANGULAR MOMENTUM *)
C
C     CALLS
C        MSUB MMLT
C     CALLED BY
C        STPRIM
C
C     INPUT
C        N ANGVEL ANGMNT
C     OUTPUT
C        DLDT
C
      REAL ANGMNT(3)
C        ANGULAR MOMENTUM VECTOR IN 'TRUTH' COORDINATE SYSTEM
      REAL ANGVEL(3)
C        CHASE VEHICLE ANGULAR VELOCITY IN 'TRUTH' COORDINATE SYSTEM
      REAL BODVEL(3)
C        CHASE VEHICLE ANGULAR VELOCITY IN CV COORDINATE SYSTEM
      REAL B(3)
C        TEMPORARY STORAGE VECTOR
      REAL DLDT(3)
C        TIME DERIVATIVE OF ANGULAR MOMENTUM VECTOR
      REAL N(3)
C        TORQUE ON C V  ABOUT ITS CENTER OF MASS IN 'TRUTH' COORDINATE SYSTEM
      REAL TRNA(3,3)
C        TRANSPOSE OF DIRECTION COSINE MATRIX
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* CONVERT ANGULAR VELOCITY TO 'TRUTH' COORDINATE SYSTEM *)
         CALL MMLT(ANGVEL,TRNA,BODVEL,3,3,1)
C     (* COMPUTE B = CROSS PRODUCT OF ANGVEL AND ANGMNT *)
         B(1)= ANGVEL(2)*ANGMNT(3)-ANGVEL(3)*ANGMNT(2)
         B(2)= ANGVEL(3)*ANGMNT(1)-ANGVEL(1)*ANGMNT(3)
         B(3)= ANGVEL(1)*ANGMNT(2)-ANGVEL(2)*ANGMNT(1)
C     (* COMPUTE DLDT=N-B *)
         CALL MSUB(DLDT,N,B,3,1)
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE MAKROT(BODVEL,OMEGA,A)
C     (* 902 6 -  FORM ROTATION MATRIX OMEGA FROM ANGULAR VELOCITY VECTOR *)
C
C     CALLS
C        <NONE>
C     CALLED BY
C        STPRIM
C
C     INPUT
C        ANGVEL
C     OUTPUT
C        OMEGA
C
      REAL A(3,3)
C        DIRECTION COSINE MATRIX GIVING TRUE C V  ATTITUDE
      REAL BODVEL(3)
C        ANGULAR VELOCITY IN CV COORDINATE SYSTEM
      INTEGER I
C        DO LOOP INDEX
      REAL OMEGA(4,4)
C        ROTATION MATRIX OMEGA, TO BE APPLIED TO QUATERNION
C        TO FORM TIME DERIVATIVE OF QUATERNION
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      DO 10 I=1,4
         OMEGA(I,I)=0 0
10       CONTINUE
      OMEGA(1,2)=BODVEL(3)
      OMEGA(1,3)=-BODVEL(2)
      OMEGA(1,4)=BODVEL(1)
      OMEGA(2,1)=-BODVEL(3)
      OMEGA(2,3)=BODVEL(1)
      OMEGA(2,4)=BODVEL(2)
      OMEGA(3,1)=BODVEL(2)
      OMEGA(3,2)=-BODVEL(1)
      OMEGA(3,4)=BODVEL(3)
      OMEGA(4,1)=-BODVEL(1)
      OMEGA(4,2)=-BODVEL(2)
      OMEGA(4,3)=-BODVEL(3)
      RETURN
C
      END
```



```
C
      SUBROUTINE QPRIME(Q,OMEGA,QPRM)
C     (* 902 7 -  COMPUTE QPRM = TIME DERIVATIVE OF QUATERNION Q *)
C
C     CALLS
C        MMLT,MSCL
C     CALLED BY
C        STPRIM
C
C     INPUT
C        Q,OMEGA
C     OUTPUT
C        QPRM
C
      REAL OMEGA(4,4)
C        MATRIX FORMED FROM ANGULAR VELOCITY COMPONENTS
      REAL Q(4)
C        ATTITUDE QUATERNIION OF CHASE VEHICLE
      REAL QINT(4)
C        INTERMEDIATE VALUES, NO PROBLEM RELATED SIGNIFICANCE
      REAL QPRM(4)
C        TIME DERIVATIVE OF OF Q
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* COMPUTE TIME DERIVATIVE OF Q = 0 5*OMEGA*Q *)
         CALL MMLT(QINT,OMEGA,Q,4,4,1)
         CALL MSCL(QPRM,QINT,4,1,0 5)
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
      FUNCTION SQR1(X)
C* 903 - RETURNS SQUARE ROOT BUT ALLOWS SLIGHTLY NEGATIVE ARGUMENT *)
      CALLS
         SQRT AMAX1
      CALLED BY
         ELIP, EPAR, TOQUAT, QUATRN, SETCOL
      INPUT
         X
      OUTPUT
         SQR1 (FUNCTION VALUE ONLY)
      REAL X
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      SQR1=SQRT(AMAX1(X,0.))
      RETURN
      END
```



```
      SUBROUTINE ANGVEC(INERTA,ANGMNT,BODVEL,A)
C* 904 -  COMPUTE TRUE ANGULAR VELOCITY VECTOR ANGVEL
          IN CV BODY COORDINATE SYSTEM  *)
      CALLS
         MINV, MMLT
      CALLED BY
         STPRIM IMU
      INPUT
         INERTA, ANGMNT, TERMNL A
      OUTPUT
         BODVEL
      REAL A(3,3)
         DIRECTION COSINE MATRIX (CHASE VEHICLE ATTITUDE WRT 'TRUTH' AXES)
      REAL ANGMNT(3)
         ANGULAR MOMENTUM VECTOR
      REAL BODVEL(3)
         CHASE VEHICLE ANGULAR VELOCITY IN CV COORDINATE SYSTEM
      REAL INERTA(3,3)
         INERTIA OF CHASE VEHICLE (LOADED)
      REAL INVIN(3,3)
         INVERSE OF MOMENT OF INERTIA TENSOR
      REAL TEMP1(3)
         INTERMEDIATE RESULTS WITH NO PROBLEM-RELATED SIGNIFICANCE
      REAL WORK(4,6)
         AN INTERMEDIATE WORKSPACE
      INTEGER IERR
         ERROR FLAG (=0 IF NO ERROR)
      INTEGER TERMNL
         FORTRAN LOGICAL UNIT NUMBER OF USER TERMINAL
      COMMON/SIMUL/TERMNL IDUMMY
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      (* FORM INVERSE OF INERTIA TENSOR *
         CALL MINV(INVIN,INERTA,3,WORK,4,6,IERR)
      IF(IERR NE 0) GO TO 10
       * CALCULATE BODVEL=INVIN*A*ANGMNT *)
         CALL MMLT(TEMP1,A,ANGMNT,3,3,1)
         CALL MMLT(BODVEL,INVIN,TEMP1,3,3,1)
      RETURN
   10 CONTINUE
      (* REPORT ERROR CONDITION AND HALT *)
         WRITE(TERMNL,901)
         STOP
  901 FORMAT(' MATRIX INVERSION FAILURE IN SUBROUTINE ANGVEC')
      END
```

```
C
      SUBROUTINE TRGATT(TRNAT,T)
C     (* 905 - COMPUTE TARGET ATTITUDE AS A FUNCTION OF TIME *)
C
C     CALLS
C        COS,SIN
C     CALLED BY
C        LOCATE,DOCK
C
C     INPUT
C        T
C     OUTPUT
C        TRNAT
C
      REAL CHGATT(3,3)
C        TRANSPOSE OF DIRECTION COSINE MATRIX REPRESENTING TRUE TARGET
C        ATTITUDE CHANGE
      INTEGER ITUMBL
C        TARGET TUMBLE AXIS--VALUE OF 1, 2, OR 3 REPRESENTS YAW,
C        PITCH, OR ROLL TUMBLE
      REAL PHI
C        ROTATION ANGLE
      REAL T
C        ELAPSED TIME
      REAL TRNAT(3,3)
C        TRANSPOSE OF TARGET'S TRUE-ATTITUDE DIRECTION COSINE MATRIX
      REAL TRNATO(3,3)
C        INITIAL VALUE OF TRNAT
      REAL TUMRAT
C        TUMBLE RATE (IN RAD/SEC)
C
      COMMON/ATINFO/TRNATO,ITUMBL,TUMRAT
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      PHI=TUMRAT*T
C     (* SELECT APPROPRIATE SET OF FORMULAS FOR TARGET ATTITUDE *)
         GO TO (10,20,30), ITUMBL
   10    CONTINUE
C        (* COMPUTE ATTITUDE CHANGE RESULTING FROM YAW TUMBLE *)
            CHGATT(1,1)=COS(PHI)
            CHGATT(2,1)=SIN(PHI)
            CHGATT(3,1)=0.0
            CHGATT(1,2)=-SIN(PHI)
            CHGATT(2,2)=COS(PHI)
            CHGATT(3,2)=0.0
            CHGATT(1,3)=0.0
            CHGATT(2,3)=0.0
            CHGATT(3,3)=1.0
            GO TO 40
   20    CONTINUE
C        (* COMPUTE ATTITUDE CHANGE RESULTING FROM PITCH TUMBLE *)
            CHGATT(1,1)=COS(PHI)
            CHGATT(2,1)=0.0
            CHGATT(3,1)=-SIN(PHI)
            CHGATT(1,2)=0.0
            CHGATT(2,2)=1.0
            CHGATT(3,2)=0.0
            CHGATT(1,3)=SIN(PHI)
            CHGATT(2,3)=0.0
            CHGATT(3,3)=COS(PHI)
            GO TO 40
   30    CONTINUE
C        (* COMPUTE ATTITUDE CHANGE RESULTING FROM A ROLL TUMBLE *)
            CHGATT(1,1)=1.0
            CHGATT(2,1)=0.0
            CHGATT(3,1)=0.0
            CHGATT(1,2)=0.0
            CHGATT(2,2)=COS(PHI)
            CHGATT(3,2)=SIN(PHI)
            CHGATT(1,3)=0.0
            CHGATT(2,3)=-SIN(PHI)
            CHGATT(3,3)=COS(PHI)
   40 CONTINUE
C     (* COMPUTE TRUE TARGET ATTITUDE *)
         CALL MMLT(TRNAT,TRNATO,CHGATT,3,3,3)
      RETURN
```



```
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

Appendix D
Attitude Parameterization

Appendix D Attitude Parameterization

APPENDIX D--ATTITUDE PARAMETERIZATION

The three computer simulations performed in this study required frequent conversions among coordinate systems. For example, the camera locations are fixed in the chase vehicle's body coordinate system, but this system is constantly rotating with respect to the inertial frame used for navigation. Similarly, the docking aid's position is fixed in the target spacecraft's coordinate system, which also rotates with respect to the inertial frame. After each observation, the onboard computer must compute the position of the chase vehicle's center of mass with respect to the target's center of mass. Because the camera and the docking aid are not at the target's center of mass, the computer must convert all the position data to a consistent coordinate system.

One of the most convenient ways to convert a vector from one coordinate system to another is to multiply the vector by the direction cosine matrix that expresses the relationship between the two coordinate systems.

Because the target's direction cosine matrix is not known in advance, it must be measured. The calculations required to do this are simplified somewhat if quaternions are used as an intermediate step. Quaternions were also found useful for a number of other operations in the simulation programs.

Direction cosine matrices and quaternions are alternative ways to express exactly the same information, and the computations to convert from one to the other are not difficult. Subroutine DIRMAT in the computer programs in Appendices A through C converts from quaternion notation to direction cosine matrix notation. Subroutine TOQUAT in Appendix C converts from direction cosine matrix notation to quaternion notation.

Suppose that A is a 3x3 direction cosine matrix that expresses the attitude of a rotated coordinate system with respect to a fixed system, and that $\underline{v}_r$ and $\underline{v}_f$ are 3x1 matrices that express the vector $\underline{v}$ in the rotated and fixed frames, respectively. Then $\underline{v}_r$ and $\underline{v}_f$ are related by the equations

$$\underline{v}_r = A\,\underline{v}_f \;, \qquad \text{(D-1)}$$

$$\underline{v}_f = A^T\,\underline{v}_r \;. \qquad \text{(D-2)}$$

The second equation makes use of the fact that direction cosine matrices are members of a special class of matrices whose inverses are equal to their transposes. The superscript T can therefore be thought of as denoting either the inverse or the transpose in this context.

The rows and columns of a direction cosine matrix have a simple physical interpretation: the columns are the unit vectors that define the fixed coordinate system, expressed in the rotated coordinate system. Similarly, the rows are the unit vectors defining the rotated coordinate system, expressed in the fixed coordinate system.

For example, if the fixed coordinate system is defined by the vectors $\underline{i}$, $\underline{j}$, and $\underline{k}$, and the rotated system's unit vectors are $\underline{u}$, $\underline{v}$, and $\underline{w}$, and if $A = [a_{rs}]$, $\underline{i}$ can be expressed in the fixed system as $(1, 0, 0)^T$ or in the rotated system as $(a_{11}, a_{21}, a_{31})^T$. Likewise, $\underline{u}$ can be expressed in the rotated frame as $(1, 0, 0)^T$ or in the fixed frame as $(a_{11}, a_{12}, a_{13})^T$.

There is a simple formula for computing the direction cosine matrix that represents the results of two successive rotations. If A_{21} represents the attitude of coordinate system 2 with respect to system 1, and A_{32} represents the attitude of coordinate ystem 3 with respect to system 2, then A_{31}, which gives the attitude of system 3 with respect to system 1, can be computed from the formula

$$A_{31} = A_{32}A_{21} \quad \text{(D-3)}$$

Quaternions are based on an entirely different view of attitude. There is a theorem that states that no matter how a rotated frame is oriented with respect to a fixed reference frame, the orientation can be accounted for by assuming a single rotation about a single axis. The angle of rotation is called the Euler angle (ϕ), and the axis is called the Euler axis ($\underline{e}$). The quaternion representing the attitude of the rotated frame is a 4x1 matrix*

$$\underline{q} = \begin{bmatrix} e_1 \sin(\Phi/2) \\ e_2 \sin(\Phi/2) \\ e_3 \sin(\Phi/2) \\ \cos(\Phi/2) \end{bmatrix} \equiv \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} \quad \text{(D-4)}$$

where e_1, e_2, and e_3 are the components of the Euler axis in the fixed coordinate system.

The direction cosine matrix can be expressed in terms of the quaternion elements as

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

and subroutine DIRMAT is a straightforward implementation of this formula. The reverse operation can be performed by any of four equivalent sets of simple formulas, but an appropriate set must be used to avoid subtraction of nearly equal quantities or division by zero. Subroutine TOQUAT contains all four sets of formulas and the logic required to select an appropriate set.

---

*In mathematics texts where quaternions are viewed as hypercomplex numbers, the elements are usually shown in a different order. The order given here is used consistently throughout this report.

Quaternions are useful when orientation is known (or sought) in terms of the Euler angle and axis. They also prove especially useful in numerical integration of angular velocity because there is a simple formula for the integration and because they are easy to renormalize.

Renormalization is required because roundoff and other errors in numerical integration cause the norm of the quaternion to drift. [The norm is the square root of the sum of the squares of the four elements, and must be exactly 1.0 for the definition in equation (D-4) to be valid.] Renormalization is a simple matter of dividing each element of the quaternion by the norm.

Numerical integration with direction cosine matrices requires only about 30 percent more additions and multiplications, but normalization is much more difficult. If a direction cosine matrix is not normalized, after several integration steps it begins to represent a mapping into a distorted coordinate system and not a simple rotation. This happens because roundoff errors leave rows and columns representing vectors that are not of unit length or vectors that do not define the axes of a rectangular coordinate system.

The formula used in the simulations for integrating angular velocity is

$$d\underline{q}/dt = \tfrac{1}{2}\Omega\underline{q} \qquad \text{(D-6)}$$

where:

$$\Omega \equiv \begin{bmatrix} 0 & \omega_w & -\omega_v & \omega_u \\ -\omega_w & 0 & \omega_u & \omega_v \\ \omega_v & -\omega_u & 0 & \omega_w \\ -\omega_u & -\omega_v & -\omega_w & 0 \end{bmatrix}, \qquad \text{(D-7)}$$

a matrix made up of angular velocity components about the spacecraft body axes. The integration starts with an initial value for $\underline{q}$ and uses equation (D-4) to calculate $\underline{q}$ at future times.

A quaternion version of equation (D-3) can be used to combine the effects of multiple rotations. If $\underline{q}_{21}$ represents the attitude of coordinate system 2 with respect to system 1, and $\underline{q}_{32}$ represents the attitude of coordinate system 3 with respect to system 2, then $\underline{q}_{31}$, which gives the attitude of system 3 with respect to system 1, can be computed from the formula

$$\underline{q}_{31} = Q\,\underline{q}_{21} \qquad \text{(D-8)}$$

where Q is a 4x4 matrix formed from the elements of $\underline{q}_{32}$

$$Q \equiv \begin{bmatrix} q_4 & q_3 & -q_2 & q_1 \\ -q_3 & q_4 & q_1 & q_2 \\ q_2 & -q_1 & q_4 & q_3 \\ -q_1 & -q_2 & -q_3 & q_4 \end{bmatrix}, \qquad \text{(D-9)}$$

in which

$$\underline{q}_{32} \equiv \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} . \qquad \text{(D-10)}$$

An alternate formula that gives identical results is

$$\underline{q}_{31} = Q' \, \underline{q}_{32} \qquad \text{(D-11)}$$

in which Q' is a 4x4 matrix formed from the elements of $\underline{q}_{21}$:

$$Q' = \begin{bmatrix} q_4' & -q_3' & q_2' & q_1' \\ q_3' & q_4' & -q_1' & q_2' \\ -q_2' & q_1' & q_4' & q_3' \\ -q_1' & -q_2' & -q_3' & q_4' \end{bmatrix} \qquad \text{(D-12)}$$

where

$$\underline{q}_{21} = \begin{bmatrix} q_1' \\ q_2' \\ q_3' \\ q_4' \end{bmatrix} \qquad \text{(D-13)}$$

Equations (D-8) and (D-11) can be considered definitions of quaternion "multiplication." The multiplication defined is ($\underline{q}_{21}\,\underline{q}_{32}$), which does not equal ($\underline{q}_{32}\,\underline{q}_{21}$). Quaternion multiplication has an identity quaternion $(0\ 0\ 0\ 1)^T$ that is analogous to the identity matrix in matrix arithmetic. The product of this quaternion with any quaternion $\underline{q}$ is $\underline{q}$, whether $\underline{q}$ is on the left or on the right in the multiplication. Also, any quaternion has an inverse $\underline{q}^{-1}$. The product $\underline{q}\underline{q}^{-1} = \underline{q}^{-1}\underline{q}$ equals the identity quaternion. The inverse is formed by simply negating the first three elements of the quaternion.

Appendix E
Adapting the Kalman Filter for Other Simulations

Appendix E Adapting the Kalman Filter for Other Simulations

APPENDIX E--ADAPTING THE KALMAN FILTER FOR OTHER SIMULATIONS

The Kalman filter used in the three simulations presented in this report can be adapted for other rendezvous and docking simulations if the appropriate changes are made.

The Kalman filter accepts a 3-element measurement vector representing the measured position of the chase vehicle's center of mass in the primary reference frame. (The primary reference frame is a nonrotating right-handed rectangular coordinate system that is centered at the target spacecraft's center of mass but aligned with the body axes of the chase vehicle the instant the video guidance system takes control.) The measurement data in the simulation programs are obtained from video information. The Kalman filter can be used for simulations that use any other sensor and target that produces X, Y, and Z position data, provided that the necessary changes are made. Subroutine ESTCOV, which estimates the measurement covariance, must be altered so the measurement covariance matrix is reasonable for the accuracy of the sensor and target used. There are two convenient methods of deriving the measurement covariance matrix. It can be calculated by fitting a curve to data from a Monte Carlo simulation or it can be derived from an analytical determination of the sensor's accuracy. The simulation programs in this report used curve-fitting and a Monte Carlo simulation to determine the equation for VARNCE, the estimated variance per axis. The estimated measurement covariance matrix R is then assigned to be a diagonal matrix whose elements along the diagonal are equal to VARNCE. It is important to realize that system performance is the issue--as long as the Kalman filter performs well, an accurate measurement covariance estimate is not necessary.

If a different chase vehicle is desired for other simulations, a thorough reevaluation of the Kalman filter equations is recommended. The dynamics equations and the assumptions made to implement the Kalman filter could be totally wrong for another chase vehicle, and overlooking these differences could cause unsatisfactory system performance.

The numerical scheme that propagates the state estimate and the state covariance matrix between measurements is valid only for small time intervals between measurements. If intervals between measurements of about 1 second or greater are desired, a more accurate integration technique must be used. To use an alternative integration scheme, subroutine PROPES must be changed to implement the new integration technique.

Other design changes not mentioned here require a thorough reevaluation of the Kalman filter equations.