MCR-83-584
Contract No. NAS8-34679

Phase 2
Final
Report

June 1983

Development of an Autonomous Video Rendezvous and Docking System

(NASA-CR-170794) DEVELOPMENT OF AN AUTONOMOUS VIDEO RENDEZVOUS AND DOCKING SYSTEM, PHASE 2 Final Report (Martin Marietta Aerospace) 114 p HC A06/MF A01 CSCL 22B G3/18

N83-28065

Unclas
28099

MARTIN·MARIETTA

MCR-83-584
Contract No. NAS8-34679

Phase 2
Final
Report

June 1983

DEVELOPMENT OF AN AUTONOMOUS VIDEO RENDEZVOUS AND DOCKING SYSTEM

John C. Tietz and
Thomas E. Richardson

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

FOREWORD

This report presents the results of a 9-month study by Martin Marietta for the National Aeronautics and Space Administration's George C. Marshall Space Flight Center. The study was the second phase of Contract NAS8-34679, Development of an Autonomous Video Rendezvous and Docking System. It resulted in a physical laboratory simulation and the demonstration of a video guidance system. Significant benefits were obtained from previous related work under Martin Marietta IR&D Project D-11R.

CONTENTS

SUMMARY

The critical elements of an autonomous video rendezvous and docking system have been built and used successfully in a physical laboratory simulation. The laboratory system demonstrated that a small, inexpensive electronic package and a flight computer of modest size can analyze television images to derive guidance information for spacecraft.

In the ultimate application, the system would use a docking aid consisting of three flashing lights mounted on a passive "target" spacecraft. Television imagery of the docking aid would be processed aboard an active "chase vehicle" to derive relative positions and attitudes of the two spacecraft.

The demonstration system used scale models of the target spacecraft with working docking aids. A television camera mounted on a 6-degree-of-freedom (DOF) simulator provided imagery of the target to simulate observations from the chase vehicle. A hardware video processor extracted statistics from the imagery, from which a computer quickly computed position and attitude. Computer software known as a Kalman filter derived velocity information from position measurements. The filter also produced "synthetic measurements" that allowed dead reckoning when the docking aid was not visible.

Tests with this system produced data that are in good agreement with an all-software simulation conducted previously. Although these tests established the viability of the measurement technique, the control system needs improvement to effectively use the measurements in guiding the chase vehicle to dock with tumbling target spacecraft. Further work is already scheduled to develop these improvements.

I. INTRODUCTION

This study is the second phase of a contract to investigate techniques that could be used in an autonomous video rendezvous and docking system for spacecraft.

Under Phase 1, we identified several techniques that appeared suitable for such a system, defined the equations and algorithms these techniques would use, and evaluated video guidance control systems based on these techniques through computer simulation.

To ensure that practical problems were considered, the simulation modeled 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 guidance sensor field of view. The simulation also modeled the characteristics and limitations of practical spacecraft to reveal subtle incompatibilities that might otherwise go unnoticed. A mission model was 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 the Long Duration Exposure Facility (LDEF), which, it is assumed, has been modified for this operation and is in a circular orbit at an altitude of 300 km. We refer to LDEF as the "target spacecraft" because, although a specific mission model was used for the simulations, it was intended that the guidance method be usable on a variety of spacecraft.

In Phase 2 of the contract, we conducted a physical simulation of the best technique evaluated under Phase 1. This technique used a docking aid comprising three flashing lights mounted on the target spacecraft (Fig. I-1). This pattern of lights uniquely defined both the relative positions and the relative attitudes of the two spacecraft.

To simulate the entire operation from a range of 300 meters to contact, three target-spacecraft models were required. Each model was built to a different scale and was used in a different part of the simulation. The smallest model was 1/100th scale and was used for ranges greater than approximately 30 meters. A 1/10th scale model was used to simulate ranges between 3 and 30 meters. For the final seconds of the docking operation, a full-scale model of a portion of one side of the LDEF was used.

To simulate the servicer spacecraft or "chase vehicle," we mounted a television camera on a 6-DOF simulator (Fig. I-2). The simulation computer sent servo commands to position the camera so that the television image would correspond to the image seen by a flight camera on a real chase vehicle. Video processing electronics converted the imagery to a set of statistics that a computer can quickly analyze to determine the relative position and attitude of the two spacecraft. These statistics were transmitted to the simulation computer, which modeled both the activity of the simulated flight computer and the dynamics of the two spacecraft.

ORIGINAL PAGE IS
OF POOR QUALITY

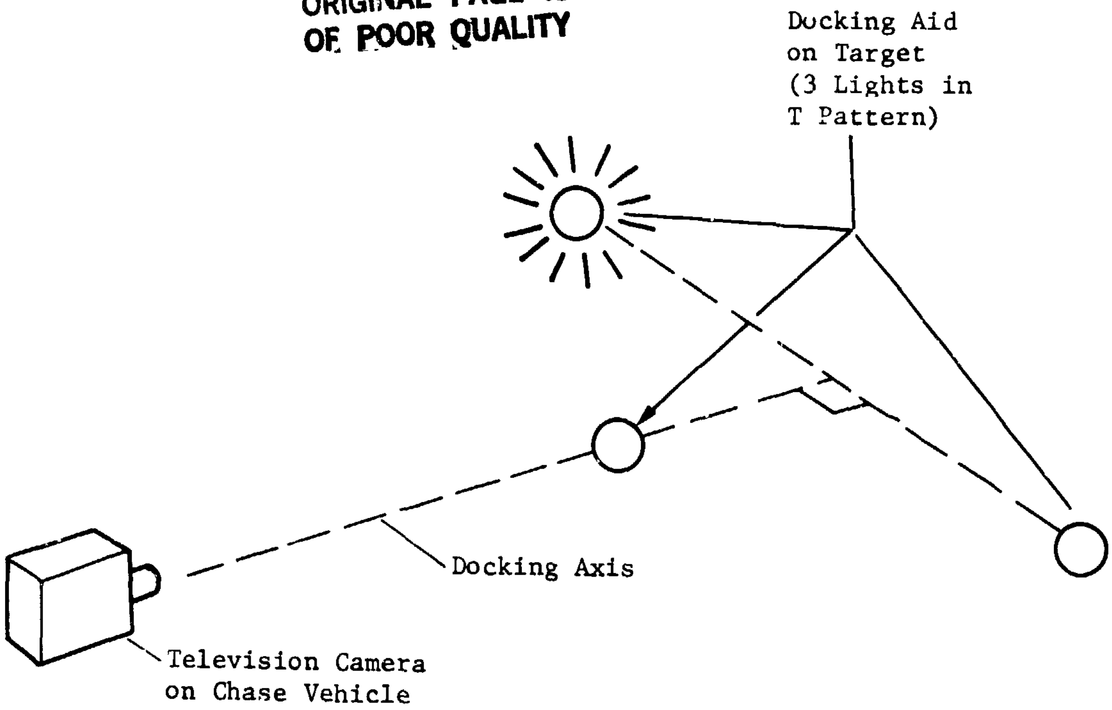


Figure I-1 Flashing-Light Docking Aid

This report concentrates on Phase 2 of the contract and repeats very little of the information that was published in the Phase 1 final report. The reader will find it advantageous to read at least the first three chapters of the Phase 1 report before reading the more technical sections of this report.

Figure I-2 Six-Degree-of-Freedom Simulator

II. CONCLUSIONS AND RECOMMENDATIONS

A. PHASE 1 CONCLUSIONS WERE CONFIRMED

The three-light docking aid and control system still appear practical, although improvements are needed.

Measurement accuracy was about twice as good in the physical simulation as in the software simulation performed under Phase 2. The improvement can be attributed to the use of a better camera than the software simulation modeled.

This increased accuracy did not materially improve control. The system still has trouble docking with target spacecraft tumbling over 1000 degrees per hour about the pitch or yaw axis. This problem underscores the need for the control system improvements that are planned under contract Phase 3.

B. OTHER USEFUL INFORMATION WAS GAINED

Although this study produced few surprises, it did produce valuable results:

1) The modeling for the all-software simulation was in close agreement with reality. This fact increases the confidence that can be placed in the Phase 1 study results. Because the Phase 3 study will also use an all-software simulation, this confirmation of models and assumptions is especially important.

2) Improvements were made in portions of the simulation program: the image interpretation algorithm now handles singularities, some minor errors were corrected, and dead reckoning was improved.

3) The real-time image interpretation scheme was shown to be practical, and we have a much better knowledge of the required size, complexity, and cost of the hardware it requires.

C. SEVERAL TOPICS WARRANT FURTHER STUDY

A third phase of this study will begin soon. In this phase the Kalman filter will be expanded to estimate target tumble parameters, and improvements will be made to the control system. These changes should improve the system's ability to deal with tumbling target spacecraft. However, several other topics should also be investigated. These topics are discussed briefly below.

1. Passive Docking Aid

The present design requires the target spacecraft to flash its docking aid lights, which may cause problems in some applications. For example, if the mission is refurbishment or retrieval of malfunctioning spacecraft, there is concern over whether the lights will operate after several years in orbit.

If the lamps are replaced with corner-cube reflectors, the same three-point docking aid can be used with no change to the interpretation algorithm, but the target can be completely inactive. However, this modification will make the chase vehicle system more complex. First, the reflectors will have to be illuminated. Supplying this illumination from the chase vehicle will require increased power. Second, some means will be required to distinguish among the three reflectors, because the algorithm requires a knowledge of which light is which.

Although we can envision methods for solving these problems, there are no test data to verify their practicality. For example, colored filters could be used to distinguish among the reflectors, but there are no data on how well a color camera could separate the three images, how much light would be required, or what wavelengths and bandwidths should produce the best results.

A study of these problems and questions might result in a system that can be used in a wider variety of missions.

2. Full-Scale Long-Range Test

Useful data could be collected at low cost by running tests with the full-scale model's docking aid located outdoors, a full 300 meters from the television camera. Tests run at various times during the day and at night would establish how well the system copes with sunlit background clutter and how effective a shutter would be in rejecting the background. In these tests, the docking aid would be moved manually, and no attempt would be made to simulate spacecraft dynamics or control algorithms.

3. Acquisition Strategies

The work to date has concentrated on what happens after the video system has taken control. If the system is to be practical, it must be able to initially locate the target spacecraft. Some effort should be spent in determining how it will do this. A complicating factor is that a tumbling target's docking aid may be visible only intermittently.

4. Camera Mounting

In the present system, the television camera is mounted rigidly to the chase vehicle. This causes no problems at long range, but in the last few meters before contact far too much fuel is used in attitude maneuvers simply to keep the docking aid within the camera's field of view. It is possible that a gimballed camera could save more than enough fuel to pay for itself. However, we do not presently have enough information to draw this conclusion.

If the camera is gimballed, the control system will need to be modified to include logic for gimbal control and for determining when attitude maneuvers are required.

5. Multiple Docking Aids

If a target is tumbling, the docking aid may be visible infrequently or may be invisible from the chase vehicle's position. Adding one or two redundant docking aids to other sides of the target spacecraft would solve this problem. If the lamps are controlled from the chase vehicle, as they are in the present control system, the guidance algorithm could simply select the docking aid that is most easily viewed.

III. SIMULATION RESULTS AND DISCUSSION

A. THE SYSTEM PERFORMED AS EXPECTED

The camera used for the physical simulations is slightly better than the camera modeled in the software simulation. The real camera has a resolution of 188 television lines horizontally by 122 vertically, and the software model assumed 128 by 128. Also, the real camera used a somewhat smaller field of view, which made the docking aid larger in the television images. We therefore expected improved accuracy from the physical simulation, but not a great deal of improvement.

The test results (Table III-1) confirmed these expectations. The position errors in the all-software simulation varied from 47 percent at 300 meters down to 3 percent at 20 meters. In the physical simulation, the corresponding errors were typically between 9 percent (on the docking axis) to 19 percent (60 degrees off the docking axis) at 300 meters and 1.5 to 2.5 percent at the 20-meter range.

Table III-1 Position Measurement Error

Range (Meters)	Error (Standard Deviation As Percent of Range)	Conditions		
		Camera Moved?	Light Orientation in Image (H = Horizontal, V = Vertical)	Approx. Distance from Docking Axis (Degrees)
307	10.91	No	V	0
298	5.94	Yes	H	0
293	7.27	No	H	0
212	13.11	No	H	60
162	0.98	No	H	0
160	2.23	No	V	0
99	7.39	No	H	60
95	9.39	Yes	H	60
91	4.37	Yes	H	0
58	1.58	Yes	H	0
15	1.26	Yes	H	0
13	0.46	No	H	0
9	0.51	Yes	H	60
8	0.69	Yes	H	0
8	1.04	No	H	60
5	0.28	No	H	0
4	2.23	No	H	0

ORIGINAL PAGE IS
OF POOR QUALITY

Table III-2 shows the errors in the measurements of the camera's position in the target spacecraft's coordinate system. These measurements are not used to estimate the chase vehicle's "state" vector; they are used only to determine the target's attitude. Thus, even gross errors are of little consequence at distances over 40 meters.

Table III-2 Errors in Measurements Used to Determine Target Attitude

Range (Meters)	Error (Standard Deviation As Percent of Range)	Conditions		
		Camera Moved?	Light Orientation in Image (H = Horizontal, V = Vertical)	Approx. Distance from Docking Axis (Degrees)
307	17.26	No	V	0
298	26.95	Yes	H	0
293	12.43	No	H	0
212	31.29	No	H	60
162	2.75	No	H	0
160	5.04	No	V	0
99	8.42	No	H	60
95	28.61	Yes	H	60
91	7.57	Yes	H	0
58	7.23	Yes	H	0
15	6.77	Yes	H	0
13	4.08	No	H	0
9	3.12	Yes	H	60
8	2.53	Yes	H	0
8	9.44	No	H	60
5	1.19	No	H	0
4	5.04	No	H	0

The error tests were performed two different ways. For the first series of tests, we placed the camera at selected positions and took approximately 12 measurements at each position without moving the camera between measurements. In the second series of tests, the camera's attitude was changed slightly between measurements so that no lamp's image appeared twice at the same place in the picture. We found no consistent difference in the error magnitude, but errors in the second series were never below 0.51 percent.

The measurement errors listed in both tables represent "noise" or random variations in the measurements, rather than the deviation from coordinates defined by theoretical arguments. We felt the "noise" level was more useful and more meaningful for four reasons:

1) In working with the small model, we found that our eyes were not nearly as good as the video system in determining where the theoretical docking axis was. The most accurate way to align the model and simulator coordinate systems was to use the video processor. Further, an error of only two degrees in positioning the model could result in an apparent error of over 10 (scaled) meters at the maximum range. We could easily observe such a bias in the data, but we could not estimate model angles that accurately.

2) In a flight system, such biases can be eliminated in calibration.

3) For the anticipated applications, an error of one or two degrees in determining the location of the docking axis will be of little consequence as long as the system is consistent, but random errors will degrade system performance.

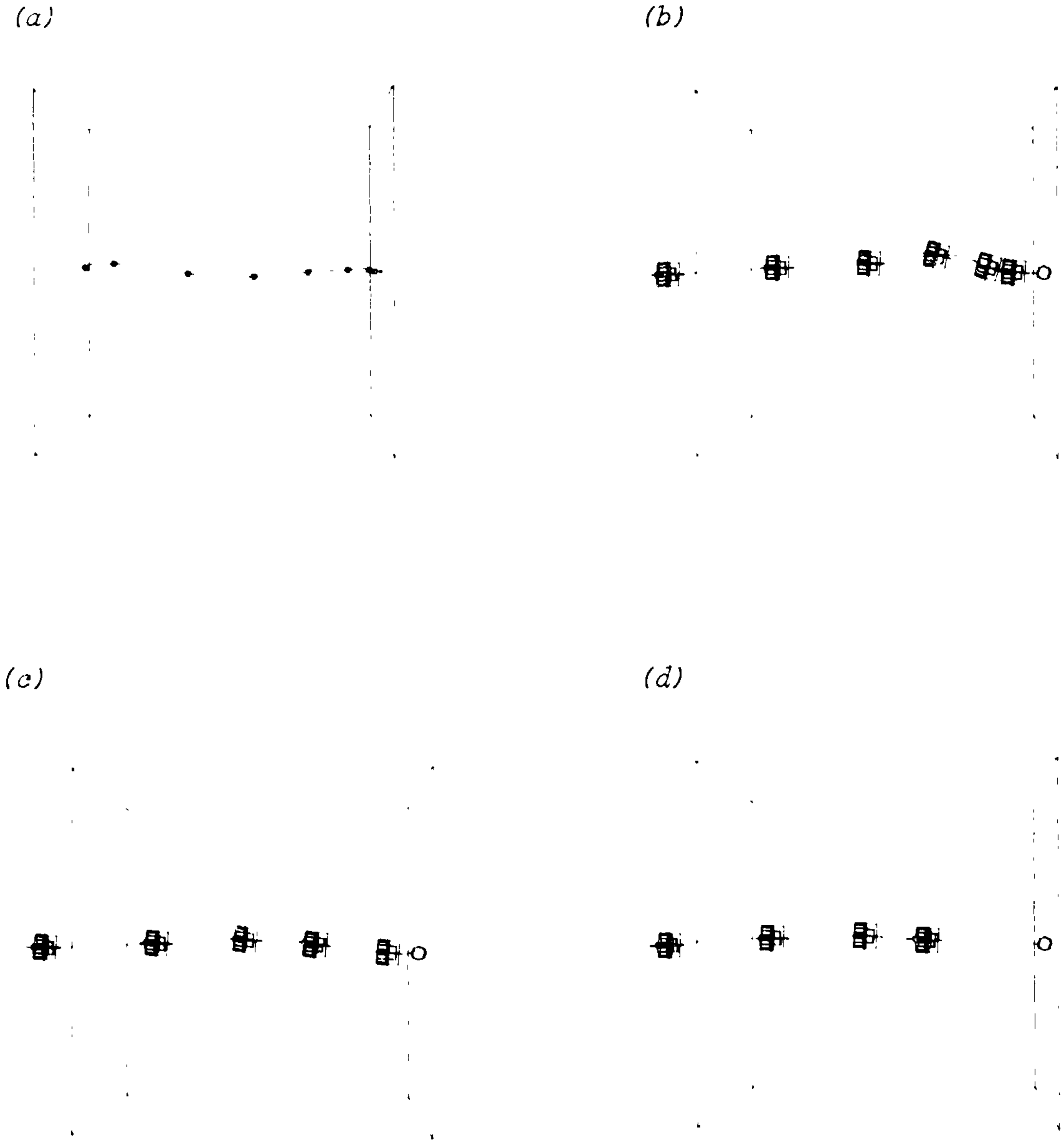
4) When the camera was close to the model, the biases were dominated by factors irrelevant to a flight system, such as simulator servo errors, a shift of a millimeter or a degree in positioning models between simulation phases, etc.

The ultimate performance of a flight system is not necessarily limited to what we achieved in the simulation. A camera with better resolution would certainly help.

B. TRAJECTORIES DID NOT CHANGE

The trajectory plots from the physical simulation were indistinguishable from those produced with the all-software simulation. The control system appeared to have the same limitations in working with tumbling targets, and, as before, the problem was not in measurement accuracy but in the control algorithms.

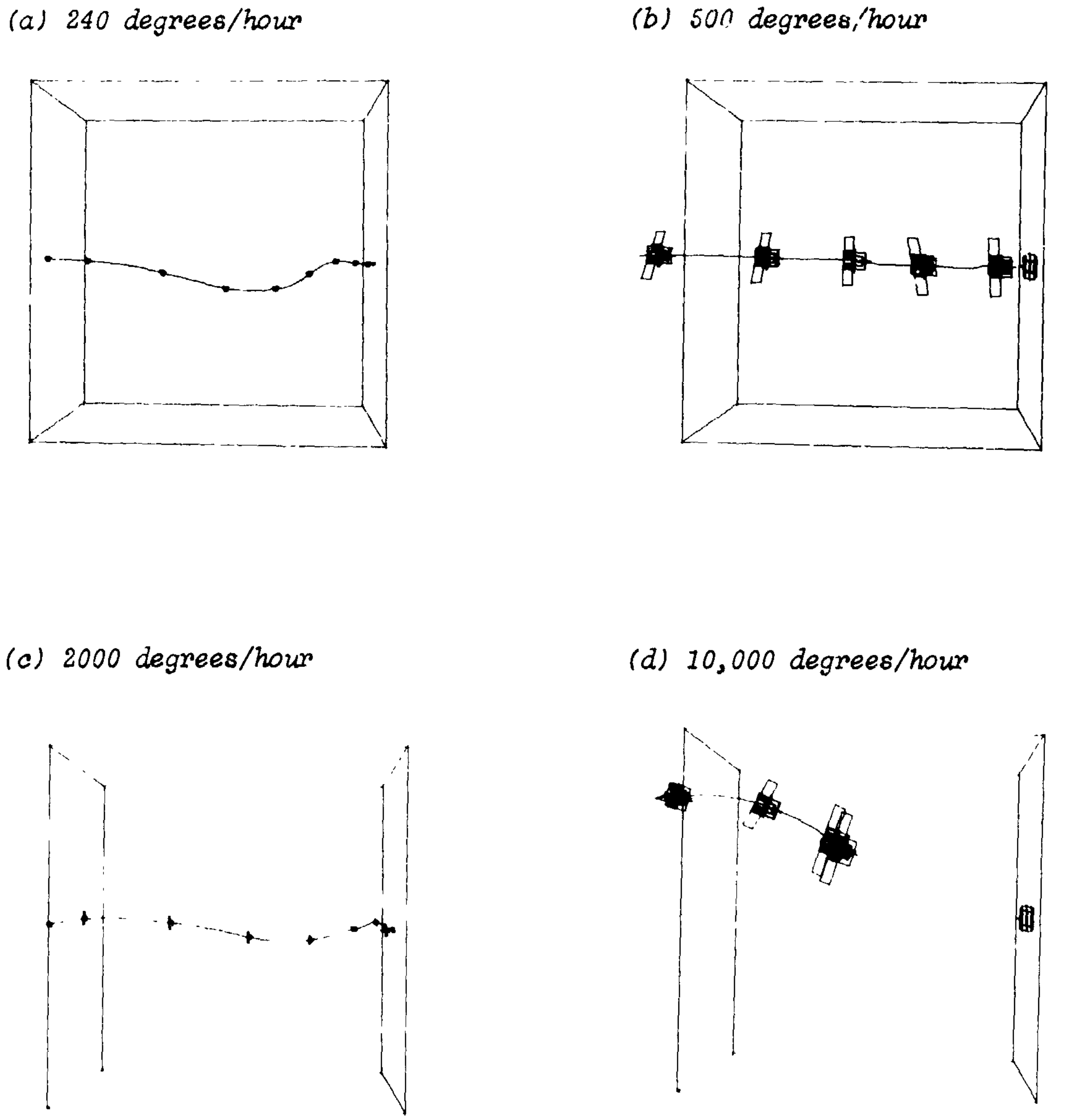
Figures III-1 through III-4 illustrate typical performance with inertially stable targets and with targets tumbling at various rates about their principal axes. In many cases we could not determine whether the docking would be successful, because the simulator reached the limit of its travel and the simulation had to be stopped. For example, a target with a high yaw rate can turn 90 degrees before the chase vehicle gets close. To simulate the end-on view that the chase vehicle would have of the target, the simulator would have to position the camera outside the room or possibly outdoors. However, the close agreement between the all-software simulation and the physical simulation with low-attitude rates strongly suggests similar agreement at higher rates.



Note:

(b) is a closeup of *(a)*; in *(c)*, a servo command limit was reached, stopping the simulation. All simulations looked good as far as they went.

Figure III-1 Typical Trajectories with Inertially Stable Targets

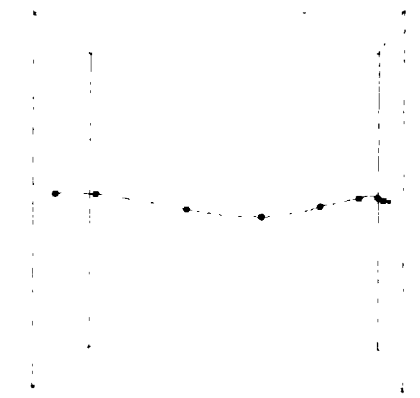


Note:

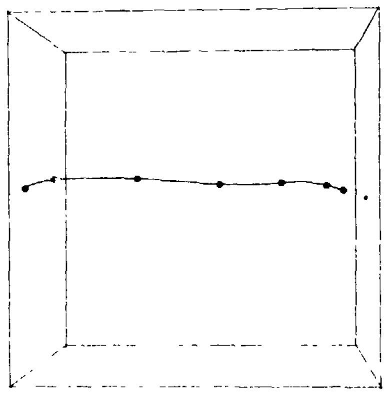
All but *(a)* stopped when servo limit was reached. The system appears well behaved up to 10,000 degrees/hour.

Figure III-2

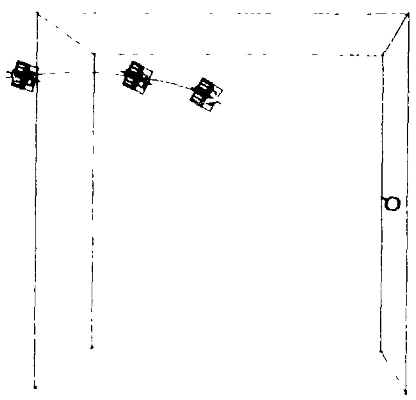
Trajectories with Various Target Tumble Rates about the Roll Axis *(a) 240 degrees/hour*



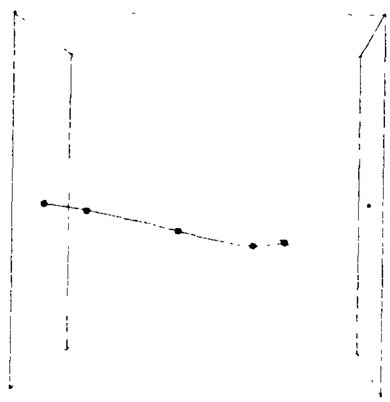
(b) 500 degrees/hour



(c) 1000 degrees/hour



(d) 2000 degrees/hour

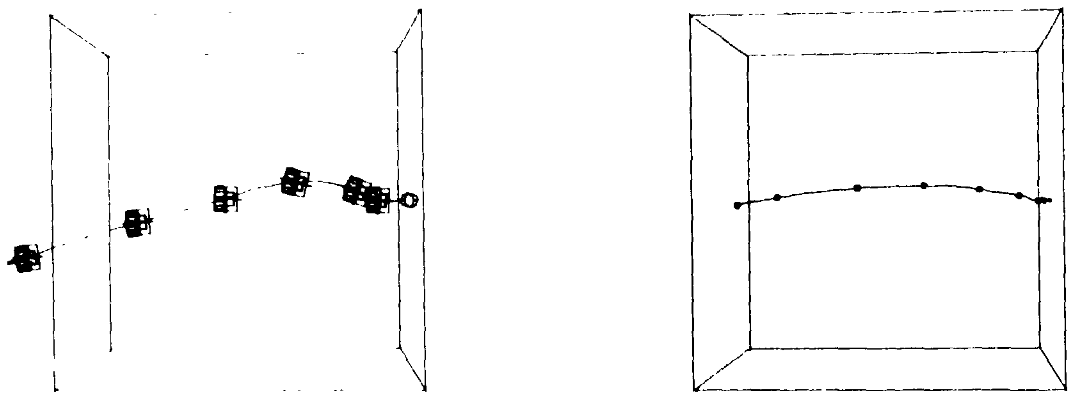


<u>Note</u>:

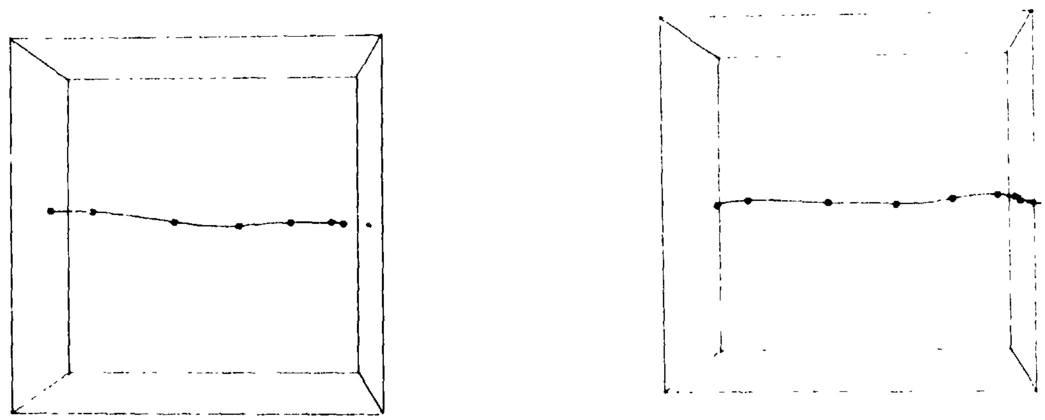
All but *(a)* stopped in simulation Phase Two because servo limits were reached. At 2000 degrees/hour, the chase vehicle could not keep the target in view, but *(c)* looked good as far as it went.

Figure III-3

Trajectories with Various Target Tumble Rates about the Pitch Axis *(a) 240 degrees/hour* *(b) 500 degrees/hour*



(c) 1000 degrees/hour *(d) 2000 degrees/hour*



> **Note:**
>
> Runs *(c)* and *(d)* reached simulator servo limits in simulation Phase Three. The chase vehicle was not out of control in any of these runs, although it overshot the docking axis in *(d)*.

Figure III-4
Trajectories with Various Target Tumble Rates about the Yaw Axis

IV. LABORATORY MODELS AND HARDWARE

The laboratory equipment for the simulation included scale models of the target spacecraft with working docking aid lights and a television camera mounted on a 6-DOF simulator (Figures IV-1 and IV-2). Video processing electronics (described in Chapter VI) extracted information from the television images to send to the simulation computer.

Figure IV-1 Simulation Camera with Small Target Model

A. THREE SCALE MODELS WERE USED

Each simulation started at a simulated range of approximately 300 meters, and a 1/100th scale model was used. When the simulated range reached approximately 30 meters, the computer interrupted the simulation to allow the operator to switch to the 1/10th scale model. The camera then moved away from the model to simulate the position where it left off, but at a scale of 1/10. Similarly, there was a switch to the large full-scale model for the final seconds of the simulation.

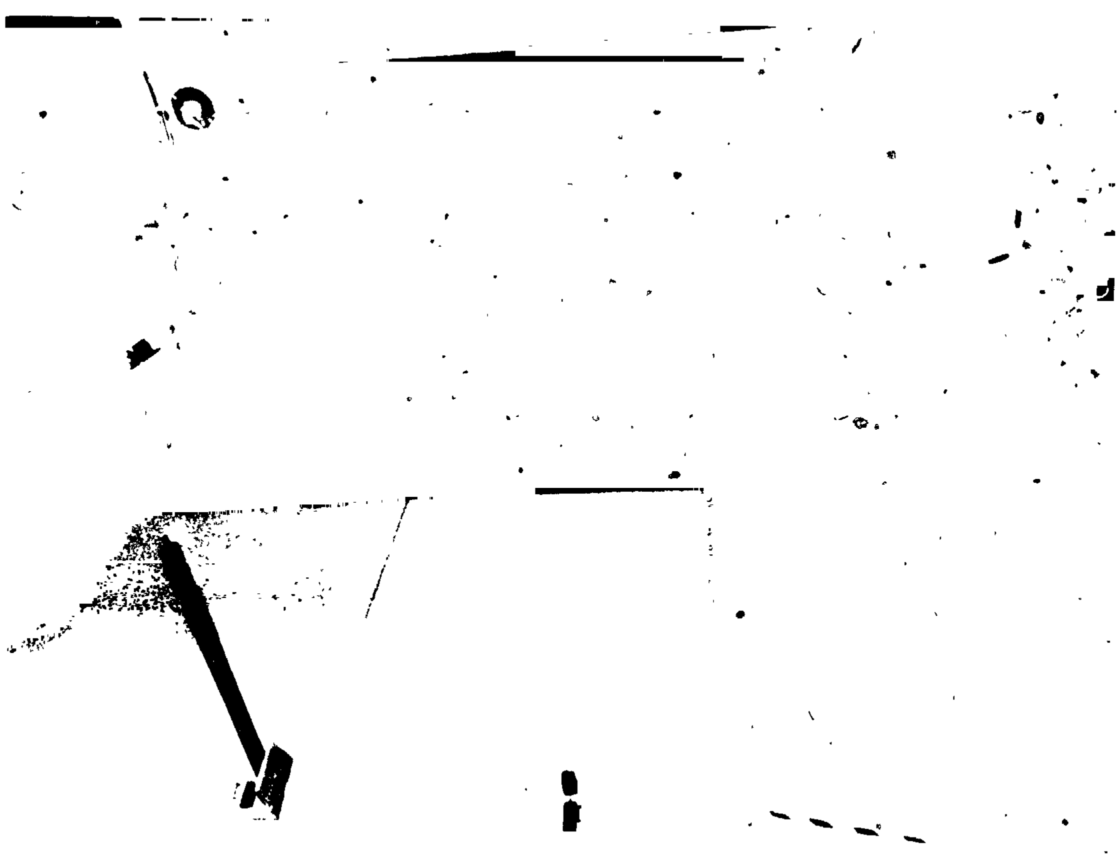


Figure IV-2 Simulator with All Three Target Models

The use of three models provided increased detail at close range and reduced the effects of camera positioning tolerances on the simulation results. It also reduced problems with lens characteristics. Because the camera used a compound lens, it was difficult to pinpoint the spot along the optical axis that should be considered the camera's "location." When the scale of a simulation is 1/1, this is not much of a problem because the error will be only about 1 centimeter. But when a 1/100th scale model is used, the uncertainty approaches 1 meter. This is a significant fraction of the camera/lamp separation near the end of the simulation. Also, the lens has a minimum focusing distance. Although we found that measurement accuracy did not suffer greatly with images moderately out of focus, we would not trust data taken with the lens almost touching the model.

The models represented the Long Duration Exposure Facility, modified to include a nonimpact docking fixture, and the docking aid, which was mounted in two of the experiment trays. The other trays did not model specific experiments; they were simply an artist's conception of what a typical payload might look like.

ORIGINAL PAGE IS OF POOR QUALITY

Different construction techniques were used for each model. The large model was made of muslin stretched over wooden frames, much like a prop for a stage play. In the plane perpendicular to the docking axis, the model was full scale, but dimensions along the docking axis were shortened by 50 percent, except for the docking aid. Only a small portion of one side of the spacecraft would fit into the room, but the model was large enough to fill the camera's field of view. The three flash lamps were directly visible in this model.

The medium-scale model (Figure IV-3) was made of balsa wood and Gatorboard, a paper-covered foam material. The flash lamps were inside the model, and light was transmitted from the lamps through Lucite rods to the outside. The ends of the rods were rounded and sanded so that the transmitted light would radiate in all directions from the end. The rods were enclosed in metal tubes so that light could not be seen from any part of the rod except the end. Because the rod stock was 1/10th the diameter of a flash lamp, the rod end looked very much like a miniature lamp.

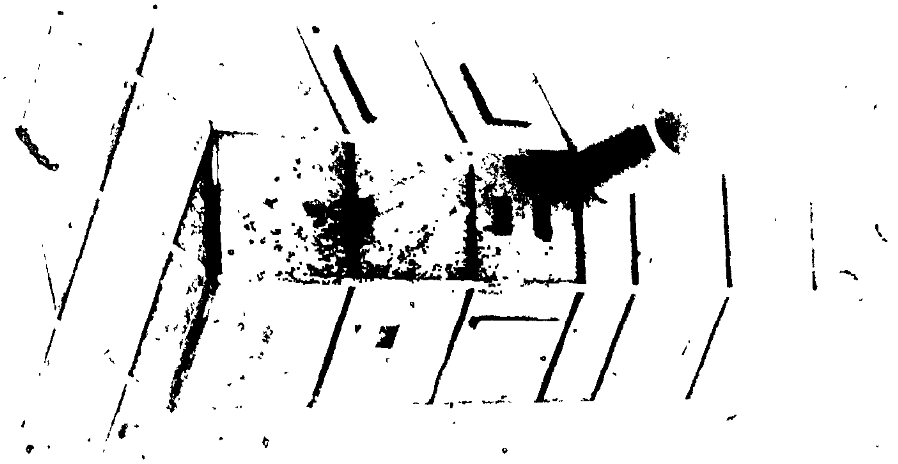


Figure IV-3 Medium-Scale Model

ORIGINAL PAGE IS
OF POOR QUALITY

The small model (Fig. IV-4) was also made of Gatorboard. Because it was too small to hold the flash lamps, the lamps were placed in a light-tight box near the model, and the light was transmitted through the model with fiber-optic cables. Because the diameter of the transparent core of the fiber optic cable was approximately 1/100th the diameter of a flash lamp, the roughened end of the cable was used as a subminiature "lamp," and the jacketed part of the cable served as its support rod.

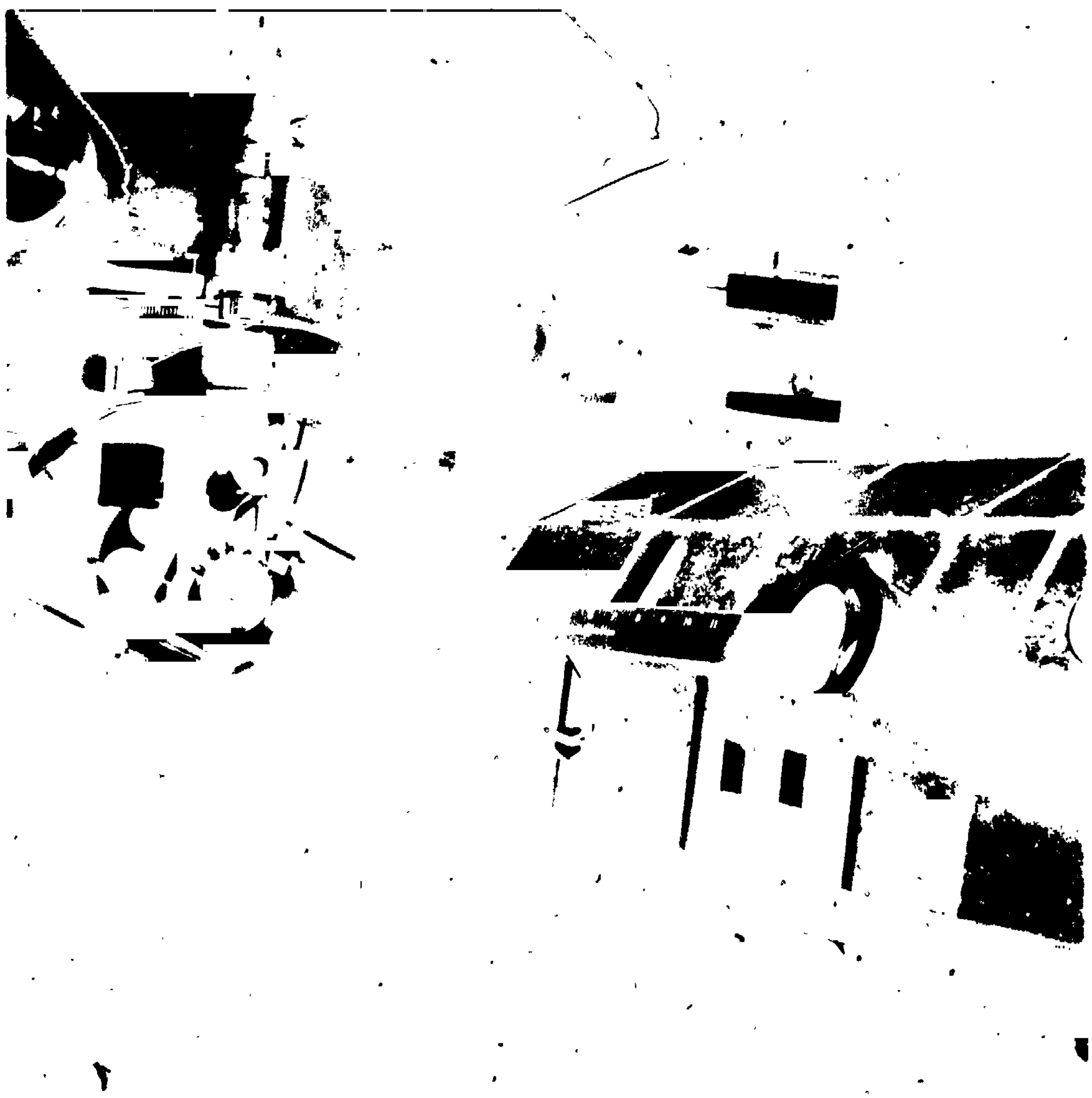


Figure IV-4 Small Model Shown with Medium-Scale Model and Camera

B. CAMERA REPRESENTED CHASE VEHICLE CAMERA

Because the chase vehicle does not appear in the camera's field of view, no scale model of it was required. In the software simulation, performed under Phase 1, the camera was not as far forward as it was for this simulation, and part of the chase vehicle structure would have appeared in the field of view. We found that in correcting this problem, we introduced another: the field of view had to be wide enough to allow the camera to see, from its new position, all three lights on the docking aid in the last few seconds before contact. When we installed a wide-angle lens, the image of the docking aid at a simulated range of 300 meters was too small for satisfactory operation of the video electronics because the lamp did not produce enough voltage in the video signal. The "normal" lens for the camera, which had twice the focal length, proved quite satisfactory at this range.

It is difficult to judge how much of this problem is due to the way the models are built; a flight system might not have the problem. However, if it does prove necessary to use two focal lengths in the flight system, a lens turret could be used to switch lenses, or two separate cameras could be used, each with a lens of fixed focal length.

This problem might also be solved by using a different camera technology. The simulation camera was a charge-injection device (CID), and its noise level was only about a factor of 40 below its saturation level. Charge-coupled devices (CCD) with considerably better characteristics are available, and both technologies have improved since our camera was manufactured.

The CID camera has one feature that is very useful in this application: it can cope with images whose brightness is far above the saturation level without "bleeding" from the bright spot into other parts of the picture. Such an image on a CCD camera may cause an entire column of the image, or even the whole image, to turn solid white. Furthermore, the CID camera does not have the after-image problem that other types of cameras have.

C. SIMULATOR POSITIONED THE CAMERA

The characteristics of the simulator we used are summarized in Table IV-1. Its characteristics determined, in part, what could be simulated. For example, because the model did not move during the simulations, the simulator could not place the camera in a location to simulate an initial target yaw of 90 degrees; the camera would have to be outside the room. Similarly, there were times during a simulation at which an unrealizable position was required, and the simulation had to be terminated. In general, this occurred only with high target attitude rates; thus, it was possible to simulate a wide variety of conditions.

Table IV-1 Simulator Characteristics

	Axis					
	x	y	z	Yaw	Pitch	Roll
Travel	4.56 m	1.84 m	2.99 m	3.19 rad	3.18 rad	6.50 rad
Speed	9.1 cm/s	8.7 cm/s	7.9 cm/s	3 mrad/s	3 mrad/s	3 mrad/s

The simulator was controlled by analog position command voltages from the computer. Because the simulator used position servos, the timing of the commands was not critical. If the computer did not respond quickly, the servos simply held the camera until the computer was ready.

V. GUIDANCE SYSTEM CIRCUITS

A. CENTROID DETECTOR — ALL DIGITAL

The video processing electronics uses a digital implementation of the centroid-detection algorithm described in Chapter IV of the Phase 1 final report. We found that an analog implementation cannot cope with the extreme dynamic range in the data it must process for a full rendezvous and docking operation. The centroid calculator's integrators must, for example, handle images in which the image of a flash lamp is in the center of the image and fills a miniscule fraction of the field of view. It must also handle the case in which a flash lamp's image is in the corner of the television picture and fills an appreciable fraction of the field of view. The ratio of two of the integrators' outputs in the latter case to their outputs in the former case can be 50,000:1. If the noise level is to remain below 10 percent of the signal over this dynamic range, the integrator must provide a signal-to-noise ratio of approximately 130 decibels. It is not practical to build such an integrator with analog circuitry.

The dynamic range could be made manageable if the camera used a zoom lens, but this solution introduces other complications. First, the zoom lens would need some kind of servo to control it. Second, using a lens of long focal length at the start of the rendezvous operation would greatly complicate acquisition and tighten attitude control requirements. We concluded that the digital approach would result in minimum system cost for a given level of performance.

The digital circuit uses "thresholded" video: the circuit converts portions of the picture, where the signal voltage is greater than a reference value, to pure white. All other portions of the image are converted to pure black. Thresholding has both advantages and disadvantages. Among the advantages are:

1) Improved rejection of background clutter; only the flash lamps will be visible in the image.

2) Simpler electronics; multiplication of the video by the deflection is reduced to a gating operation.

3) Potential adaptation to adverse lighting conditions; the computer could adjust the reference voltage until the white area in the image matches expectations for the estimated viewing position. The system might be able to cope with unexpected sun glint from a shiny surface by using this approach.

The main disadvantage is the all-or-nothing nature of the calculation. If the lamps are not as bright as a shiny part of the target spacecraft, they might be ignored altogether. Usually this situation will be detected for two reasons. First, the system always examines an image in which no lights are flashing and subtracts the values it receives from the video electronics on this frame from the values it receives on all other frames. Because the no-lamp image does not have time to change much between measurements, the resultant lamp area in the image will usually be close to zero or may be negative. The image interpretation algorithm can detect this kind of error and ignore the measurement. Second, even if the data masquerade as legitimate data, the image interpretation routine will generally provide a position measurement that is grossly different from what the Kalman filter is expecting. If the filter receives such a measurement, it will ignore it. However, to minimize the problem, the target spacecraft should not have shiny surfaces near the flash lamps. It would also help if the camera were equipped with a shutter that opens only during the flash. A shutter can reduce background clutter by a factor of 30 without attenuating the flash.

The digital implementation of the algorithm appears much more complex than the analog version, but it consumes only about twice the circuit card area. The algorithm and circuit details are discussed in Chapter VI.

B. IMAGE INTERPRETATION WAS IMPROVED

In the all-software simulation developed under Phase 1, image corruptions were simulated by adding random numbers to lamp-image coordinates. Because this approach made singularities highly unlikely, the scene-analysis subroutine (POSIT) did not thoroughly check for them.

When the subroutine was used with real imagery, we found that these singularities occur several times during a docking operation, because the television image comprises a finite number of lines and columns. The subroutine can no longer ignore the possibility that two lamps will lie on the same horizontal or vertical line in the image.

The improved version of subroutine POSIT tests for singularities. When it finds one, it uses an alternate formula to interpret the image Because the alternate formulas are derived from the normal formula by extracting limits, we will first consider the normal formula.

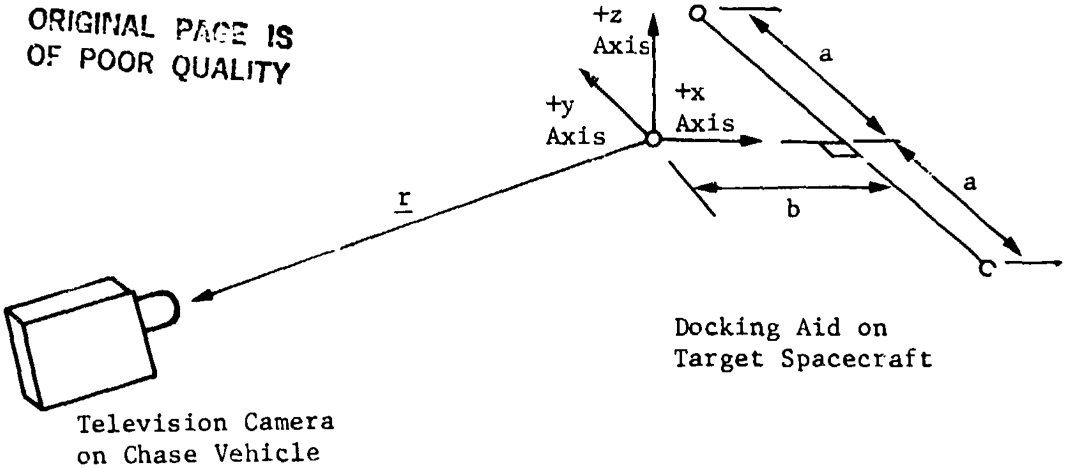
Subroutine POSIT calculates the position of the chase vehicle in the docking aid's coordinate system (Fig. V-1). Some simplification can be gained by working with image-plane quantities that are independent of camera rotation about the line of sight. Figure V-2 illustrates the set of quantities used in the formulas in the subroutine. These quantities can be measured on the image and can be defined in terms of the image-plane coordinates of the three lamp images, (u_1,v_1), (u_2,v_2), and (u_3,v_3):

a' is half the length of the line connecting (u_1,v_1) to (u_3,v_3);

b' is the length of the line that connects (u_2,v_2) to the midpoint, (u_m,v_m), between (u_1,v_1) and (u_3,v_3);

(u_c,v_c) is a point on the line between (u_1,v_1) and (u_3,v_3) at which a perpendicular to that line passes through (u_2,v_2);

h is the length of the line connecting (u_c,v_c) with (u_2,v_2).



Note:
Subroutine POSIT computes the vector r from an analysis of the video imagery. The lengths a and b are 1 meter in the simulation, but the formulas do not require any specific lengths as long as a and b are known.

Figure V-1 Definition of Docking Aid Coordinate System

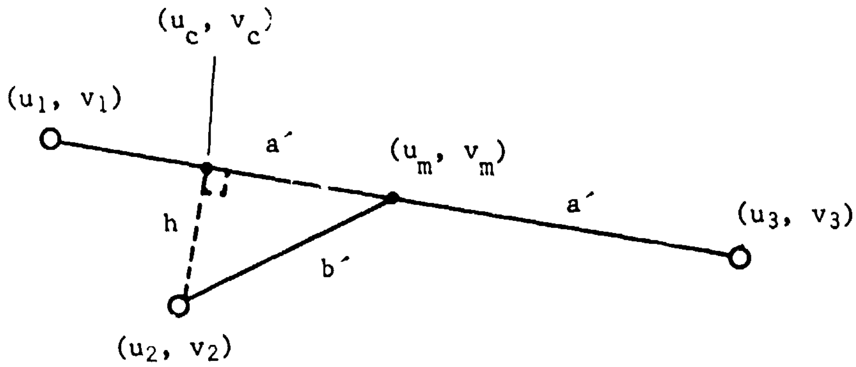


Figure V-2 Image-Plane Quantities Used for Analysis

The length a' is found by the Pythagorean theorem:

[1] $$a' = 0.5 \sqrt{(u_3 - u_1)^2 + (v_3 - v_1)^2} \ .$$

The slope of the line connecting (u_1,v_1) to (u_3,v_3) is

[2] $$s = \frac{v_1 - v_3}{u_1 - u_3} \ .$$

ORIGINAL PAGE IS
OF POOR QUALITY

The slope of the line of length h is

[3] $$s' = -1/s \ .$$

Thus, the equations of the latter two lines are

[4] $$v = s(u - u_1) + v_1$$

and

[5] $$v = s'(u - u_2) + v_2 \ .$$

The intersection is found by setting these expressions for v equal to each other:

[6] $$s(u_c - u_1) + v_1 = s'(u_c - u_2) + v_2,$$

which can be manipulated to give

[7] $$u_c = \frac{v_2 - v_1 + su_1 - s'u_2}{s - s'} \ .$$

If u_c substitutes for u in Eq 5,

[8] $$v_c = s'(u_c - u_2) + v_2.$$

Now h can be computed by the Pythagorean theorem:

[9] $$h = \sqrt{(u_c - u_2)^2 + (v_c - v_2)^2} \ .$$

Similarly,

[10] $$h' = \sqrt{(v_m - v_2)^2 + (u_m - u_2)^2} \ .$$

where $v_m = (v_1 + v_3)/2$ and $u_m = (u_1 + u_3)/2$.

We now have formulas for a', b', and h in terms of lamp coordinates in an image. Furthermore, these formulas are accurate in spite of any rotation about the line of sight except at a few singularities, which we will discuss later.

Now consider the three orthographic projections shown in Figure V-3. The arrow is a unit vector that points to the observer. The coordinates of its tip in the docking-aid coordinate system are (-x,-y,z), where the minus signs are used to make all the quantities positive. The orthographic projection, in which the vector appears as a point, will approximate what the television camera sees. The image is only an approximation for two important reasons:

1) It is too large, by the ratio r/f, where r is the distance from camera to target and f is the camera lens focal length. This fact will be used to advantage in calculating range.

2) It ignores perspective effects. The image distortion from perspective effects becomes significant at close range. However, in practice the error in image interpretation caused by ignoring the effects is small.

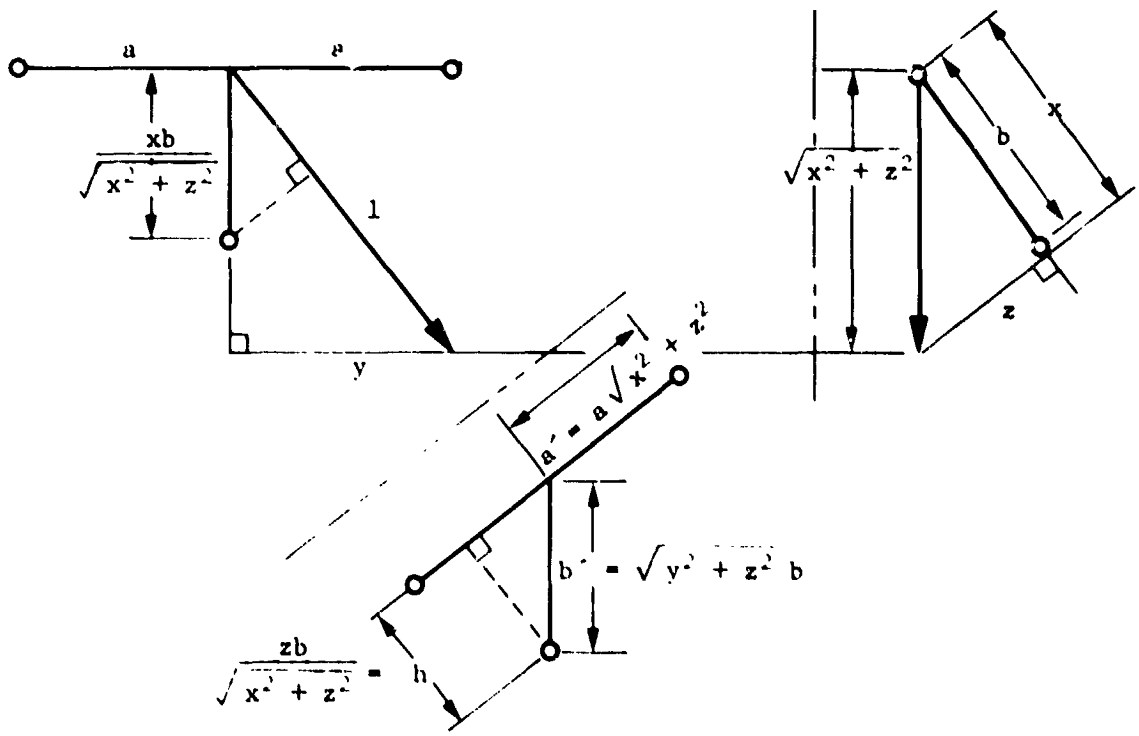


Figure V-3 Orthographic Projections Used to Derive Formulas

ORIGINAL PAGE IS
OF POOR QUALITY

The projections of Figure V-3 were produced by graphical construction. All the labeled dimensions can be determined from the drawing by finding similar triangles, noting that $x^2 + y^2 + z^2 = 1$, and applying the Pythagorean theorem. These techniques provide the following formulas for a', b', and h:

[11] $$a' = a\sqrt{x^2 + z^2}\ ;$$

[12] $$b' = b\sqrt{y^2 + z^2}\ ;$$

[13] $$h = bz/\sqrt{x^2 + z^2}\ ;$$

where a and b are the distances on the target spacecraft whose projections are the lines labeled a' and b' in Figure V-3. In the simulation, a = b = 1 meter, but the formulas do not require any particular values.

If we define

[14] $$D \triangleq \frac{(a'b)^2}{(ab')^2} = \frac{x^2 + z^2}{y^2 + z^2}\ ,$$

then by substitution, we can define

[15] $$k \triangleq \frac{b'(1 + D)}{2h\sqrt{D}} = \frac{1 + z^2}{2z}\ ,$$

which relates a set of observable quantities to the observer's position. From this definition,

[16] $$z^2 - 2kz + 1 = 0,$$

which can be solved by the quadratic formula to give

[17] $$z = k - \sqrt{k^2 - 1}.$$

Then substitution gives expressions for the other observer coordinates:

[18] $$x = \sqrt{\frac{D - z^2}{1 + D}}\ ;$$

$$[19] \quad y = \sqrt{\frac{1 - Dz^2}{1 + D}} .$$

ORIGINAL PAGE IS
OF POOR QUALITY

Now $(x\ y\ z)^t$ is a unit vector in the direction of the observer if the signs of x, y, and z prove correct; all the squaring and extracting of square roots have destroyed sign information. We will now restore the signs.

First, note that the formulas for x and y always give a positive answer. The formula for z also gives a positive answer, although it is less obvious. We can therefore simply multiply the vector elements that should be negative by -1.0.

The x component will always be negative, because the lights cannot be seen if the observer's position has a positive x component: the target spacecraft will be in the way. We can therefore unconditionally multiply the x component by -1.0.

The y component should be negative if the center light appears closer to the right-hand light than to the left-hand light in the television image. The lamp-image separations can be computed by the Pythagorean theorem; the y component should be negative if

$$[20] \quad \sqrt{(u_1 - u_2)^2 + (v_1 - v_2)^2} < \sqrt{(u_3 - u_2)^2 + (v_3 - v_2)^2} .$$

We can save some computation by comparing the squares of the lengths rather than the lengths themselves:

if $(u_1 - u_2)^2 + (v_1 - v_2)^2 < (u_3 - u_2)^2 + (v_3 - v_2)^2$ let $y = - y$.

To test for a negative z component, the computer need only decide whether the center light appears above or below the line joining the other two lights. However, the definitions of above and below should not change if the camera rotates about the line of sight. The test used in subroutine POSIT is the sign of the nonzero component of the cross product of two vectors. The first vector is a line from the left lamp to the right lamp in the image. The second vector is the line connecting the point (u_c,v_c) to the center-lamp image (u_2,v_2). If the third compenent of

$$[21] \quad \begin{bmatrix} u_3 - u_1 \\ v_3 - v_1 \\ 0 \end{bmatrix} \times \begin{bmatrix} u_2 - u_c \\ v_2 - v_c \\ 0 \end{bmatrix}$$

is positive, the center light is above the line joining the two side lights, no matter how the camera is rotated. This test can be rewritten as

if $(u_3 - u_1)(v_2 - v_c) > (v_3 - v_1)(u_2 - u_c)$ let $z = -z$.

If we can compute the distance from the docking aid to the camera, we can compute the camera's coordinates by multiplying x, y, and z by this distance. Recall that the construction in Figure V-3 makes the image larger than what the camera sees by a factor of r/f, where f is the known lens focal length and r is the docking-aid-to-camera distance we are seeking. Because a' in the figure is $a\sqrt{x^2 + z^2}$,

[22] $$\frac{r}{f} = \frac{a\sqrt{x^2 + z^2}}{a'}$$

or

[23] $$r = \frac{a\sqrt{x^2 + z^2}\, f}{a'} .$$

Thus, the camera's position in a coordinate system centered at the base of the docking aid is

[24] $$\begin{bmatrix} xr \\ yr \\ zr \end{bmatrix} .$$

The remainder of the simulation program needs to know the position in a coordinate system whose origin is at the center light. The difference is simply adding b to the x component of the position:

[25] $$\begin{bmatrix} xr + b \\ yr \\ zr \end{bmatrix} .$$

The formulas in subroutine POSIT are exactly those we have just discussed, but with the notation differences shown in Table V-1.

ORIGINAL PAGE IS OF POOR QUALITY

Table V-1 Notation in Subroutine POSIT

Notation in Formulas	Notation in Subroutine
a, b, h, D, s, u, v	[Same as Formulas]
x, y, z	XP, YP, ZP
r	RHO
s', a', b'	SP, AP, BP
u_m, v_m, u_c, v_c	UM, VM, UC, VC
k	AK
f	FOCLEN
[Final Position Vector]	RELPOS

Up to this point, we have ignored several possibilities that could result in division by zero when a computer tries to solve an equation. Subroutine POSIT uses the formulas we just derived, unless it detects a condition that would lead to division by zero. These formulas are provided in Appendix A between statement 20 and statement 30 and in all statements after statement 60 of subroutine POSIT.

The rest of the subroutine tests for perverse cases ($s=0$, $h=0$, $a'=0$, or $b'=0$) and uses alternate formulas. These formulas are derived from the basic formulas by extracting the limit as some parameter approaches a critical value. All the formulas are therefore equivalent to those derived here.

For example, if $u_1 = u_3$,

[26] $$a' = 0.5 \sqrt{(u_3 - u_1)^2 + (v_3 - v_1)^2} = 0.5 \left| v_3 - v_1 \right| ,$$

which results in a simpler formula for a', but

[27] $$s = \frac{v_1 - v_3}{u_1 - u_3} = \frac{v_1 - v_3}{0} ,$$

which would be an error. However, we can substitute the expression for s into each subsequent formula where s is used and extract the limit as u_1 approaches u_3:

[28] $$s' = \frac{-(u_1 - u_3)}{v_1 - v_3} = 0 ,$$

[29] $$u_c = \frac{v_2 - v_1 + \left(\frac{v_1 - v_3}{u_1 - u_3}\right) u_1}{\left(\frac{v_1 - v_3}{u_1 - u_3}\right)} \;;$$

[30] $$u_c = (v_2 - v_1)(u_1 - u_3) + \left(\frac{v_1 - v_3}{v_1 - v_3}\right) u_1 \;.$$

The limit as u_1 approaches u_3 is

[31] $$u_c = \left(\frac{v_1 - v_3}{v_1 - v_3}\right) u_1 \;,$$

or

[32] $$u_c = u_1 \;.$$

Similarly,

[33] $$v_c = v_1 \;, \text{ and}$$

and

[34] $$h = |u_1 - u_2| \;.$$

There is one situation in which nothing can be done: if all the lamps' images are at the same point, the subroutine cannot determine anything except the direction to the target. In this case, the subroutine returns a default set of coordinates so that processing can continue. These coordinates are subsequently thrown out by the Kalman filter so they do not affect navigation.

C. THE KALMAN FILTER HAS TWO IMPROVEMENTS

The Kalman filter has been changed very little from the original implementation reported in the Phase 1 final report. The state variables are still the three-position and three-velocity vector components, and the reference frame is still the nonrotating "primary" reference frame, which is centered at the target spacecraft's center of mass and aligned with the chase vehicle's body coordinate system at the moment the video guidance system takes control. The filter is still used to derive velocity from position measurements, to improve estimates of position, and to allow dead reckoning when the camera cannot observe the docking aid.

Only two changes have been made: the filter now rejects "unreasonable" data, and the algorithm used for dead reckoning for attitude control has been completely revised.

1. Reasonableness Check

The original Kalman filter weighted each measurement according to the confidence it had in the measurement versus the confidence it had in the position estimate provided by its mathematical dynamics model. It did not check measurements for reasonableness, so a single measurement that was orders of magnitude off could turn its estimates into complete nonsense. Because reflections from shiny surfaces and similar problems may occasionally produce faulty measurements, we added a test that throws out data that grossly disagree with the mathematical model.

Along with the state estimate, the model computes the likely error in the estimate. The error information is in the form of a covariance matrix, P. The diagonal elements of P correspond to the variance, or the square of the standard deviation, of each of the state elements. Matrix elements off the diagonal indicate how errors in one state element correlate with other elements. For example, if P(1,2) is positive, it indicates that elements 1 and 2 of the state estimate vector tend to have errors that are off in the same direction: if one is too high, the other probably is also too high. If P(1,2) is negative, the errors in the two elements have the opposite relationship. If one element is too high, the other is probably too low.

A simple scalar test does not take these interrelationships into account; the filter uses a more complex test that does. To test a measurement, subroutine INCORP computes

[35] $$c = (\underline{m} - \underline{e})^t P^{-1} (\underline{m} - \underline{e}),$$

where $\underline{m}$ is the measured position vector and $\underline{e}$ is the estimated position, the first three elements of ESTATE. The scalar c is a measure of how the error compares with the expected error.

If the errors have a Gaussian distribution, values of c greater than 2.4 can be expected about half the time, and values greater than 11.3 can be expected about 1 percent of the time. The subroutine throws out measurements that produce values of c greater than 16. In normal operation, such errors should occur only once or twice in a docking operation.

In practice, we find that these errors occur more often because the distribution is not Gaussian. However, the largest number of thrown-out measurements occurs at the switchover from one scale target model to the next at the end of a simulation phase. No matter how carefully we aligned the models, we could not align them accurately enough to keep the filter from detecting the difference, because the filter's state estimates at the time of the switchover are very good and the P matrix reflects their accuracy. The filter usually throws out several measurements before the probable error in its estimates increases enough to allow it to accept new measurements.

This problem introduces a philosophical question: should the computer program faithfully represent flight software, or should it be adjusted to accommodate the shortcomings of a physical simulation? We chose to allow the system to reject a few measurements. The decision does not appear to affect control system operation significantly because the misalignments are small.

2. Dead Reckoning

We completely revised subroutine ESTRPY. This subroutine allows attitude control when the chase vehicle cannot observe the docking aid. Its output is a vector of pointing errors, RPYERR.

The first element of the vector is the roll error. ESTRPY always returns a value of zero for this element, because it has no rational basis for estimating roll errors. The effect of returning a zero is that the chase vehicle will not attempt any roll corrections.

To compute pitch and yaw errors, which are the next two vector elements, ESTRPY first computes a unit vector (expressed in the chase vehicle's coordinate system) that points to the target:

[36] $$\underline{r} = -A_c \underline{e} / |\underline{e}| ,$$

where

$\underline{r}$ is the unit vector;

A_c (represented in the program as ACV) is the direction cosine matrix that describes the chase vehicle's attitude with respect to the "primary" reference frame;

$\underline{e}$ is the first three (position) elements of the state estimate vector ESTATE.

The projection of this vector onto the chase vehicle's y-z plane provides a synthetic "image" of the target from which yaw and pitch errors can be estimated: the pitch error is ATAN2(r_3,r_1), and the yaw error is ATAN2($-r_2$,r_1), where ATAN2 is the FORTRAN two-argument arc tangent function.

The values calculated by this method are only approximations, but they are adequate to get the target back into the field of view.

D. ERRORS IN THE SIMULATION PROGRAM WERE CORRECTED

Two errors were found in the simulation program used in Phase 2. Neither error alters the control system's performance enough to change the study conclusions.

The first error was a typographical error in subroutine QUATRN. In the formula for the variable QT(2), one subscript was wrong. This caused errors in measurements of the target spacecraft's attitude.

The second error was in the computation of "true" chase vehicle attitude. Subroutine LPRIME implemented a formula that was simply wrong. The error was introduced during a program modification to change the coordinate system used to express angular momentum. Its effect is a small fictitious torque on the chase vehicle. Because the magnitude of the torque is only about 1 percent of the torque from the thrusters, the error caused no observable change in spacecraft behavior. The correction affects subroutine LPRIME and the line in subroutine STPRIM where LPRIME is called.

ORIGINAL PAGE IS
OF POOR QUALITY

VI. VIDEO PROCESSING ELECTRONICS

A. OVERVIEW

The video processing electronics (Figure VI-1) implements the centroid-calculation portion of the algorithm described in Chapter V. Its inputs are video imagery from a television camera mounted on the simulator and control signals from the simulation computer. It produces numbers from which the computer can calculate the location of a flash lamp's image in the television picture.

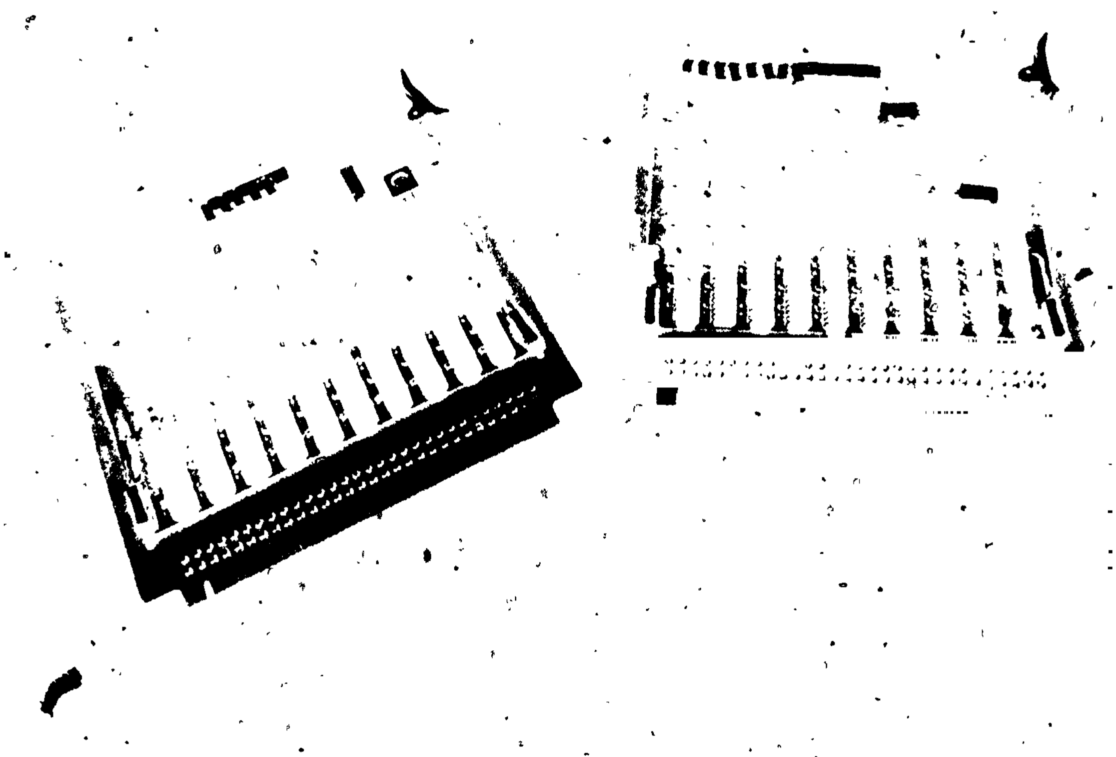


Figure VI-1 Video Processing Electronics

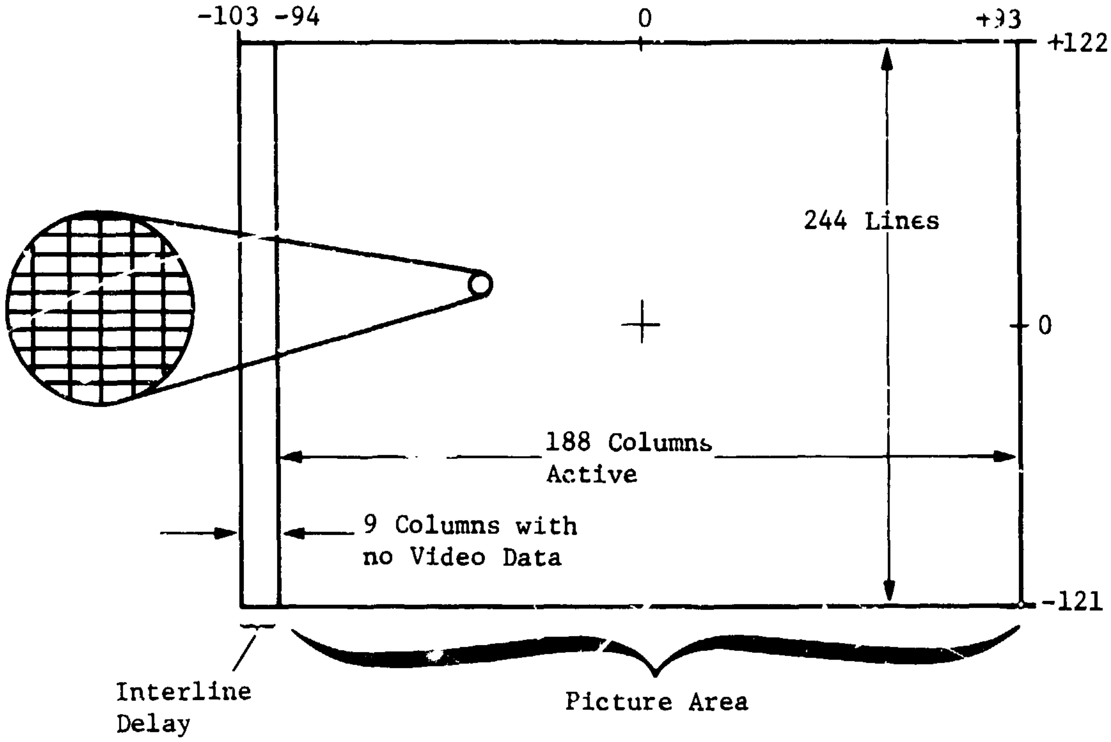
The television camera used in the simulation divides an image into 45,872 tiny spots of light arranged as a rectangular array of 244 lines by 188 columns. (Although there are 244 lines, the vertical-axis resolution is only 122 lines, because the detector elements are rescanned to create a signal compatible with standard video monitors.) The camera senses the average brightness of each spot of light and produces an electrical voltage that is proportional to the brightness. It cannot detect image details within a spot; it can measure only the average brightness of the spot as a whole. The spots are called "picture elements," or "pixels" for short.

The camera "scans" what it sees, much as a reader scans a page of text. It starts in the upper left corner of the picture and, reading from left to right, sends the voltage for each pixel in the top line to the video processing electronics, one at a time. Along with the pixel voltages, it sends a logic signal called the "pixel-rate clock," which is a series of pulses, one for each pixel. The video electronics uses these pulses two ways. First, the pulses tell the electronics exactly when each pixel is sent, so the electronics does not miss any pixels or examine any twice. Second, by counting the pulses, the electronics can keep track of the current column number. After the first line has been sent, the camera begins sending the second line, and so on. At the start of each line, the camera sends a second logic signal called the horizontal sync pulse, which helps the electronics identify the first pixel in each line. Similarly, just before the first line, the camera sends a third logic signal called the vertical sync pulse, which helps the electronics recognize the first line of a new "frame" of imagery. The camera used in the simulation provides each of these logic signals on a separate wire.

The video processing electronics first converts each pixel to an "on-or-off" signal. The effect is similar to a black and white television with the contrast very high; there are no shades of gray, only white and black or on and off. If the flash lamps are much brighter than the rest of the scene, this process leaves all the pixels "off" except for the few that represent the image of the flash lamp. The electronics uses three counters and two adders to find the coordinates (column and line numbers) of the center of the cluster of "on" pixels. This center is called the center of brightness or "centroid." The "pixel counter" counts the number of "on" pixels, the "column counter" keeps track of the current column number, and the "line counter" keeps track of the current line number. Each time an "on" pixel is detected, the column and line adders add the values in the column and line counters to totals maintained in two data storage circuits called accumulators. After the image is fully scanned, the coordinates are found by dividing the values in the accumulators by the number of "on" pixels.

When the camera starts to scan an image (Fig. VI-2), the adders and the pixel counter are all set to zero, the column counter is set to -103, and the line counter is set to +122. These starting values cause row 0 and line 0 to be in the center of the frame. During the scan, the column counter counts from -103 to +93 for each line. The first pixel of the line is received when the count is -94, because there is a delay between the end of one line and the start of the next. At the end of each line, the column counter is reset to -103, and the line counter counts down first to 121, then to 120, and so on to -121 at the bottom line of the image.

ORIGINAL PAGE IS
OF POOR QUALITY



Note:

One of two "fields" is shown. Each field contains the same information. Within each field, pairs of lines contain the same information, so the effective vertical resolution is 122 lines. Columns -103 through -95 do not contain video information.

Figure VI-2 Television Image Line and Column Scheme

Consider what happens if only three pixels in an image are "on," and their (column,line) coordinates are (0,-2), (1,-2), and (2,-3). The system computes the average column by adding the three column numbers (0 + 1 + 2 = 3) and dividing by the number of pixels that were "on." The result is 3/3, or column 1. Similarly, it computes the average line by adding the three line numbers [(-2) + (-2) + (-3) = -7] and dividing by the number of "on" pixels: the average line number is (-7)/3 = -2.333.

The electronics itself does not perform the division. It just supplies the numerator and denominator to the simulation computer, and the computer performs the division. The electronics, therefore, has the simple job of computing three totals: the sum of the column numbers, the sum of the line numbers, and the number of "on" pixels. It updates each of the three totals each time it receives an "on" pixel.

In this example, the first "on" pixel is detected in line -2. The column accumulator has 0 added to it, the line accumulator has -2 added to it, and the pixel counter is incremented by 1. At the next pixel, 1 is added to the column accumulator and -2 to the line accumulator, and the pixel counter is incremented again. In the next line, 2 is added to

ORIGINAL PAGE IS
OF POOR QUALITY

the column accumulator, -3 is added to the line accumulator, and the pixel counter is incremented a third time. Thus, at the end of the scan, the column accumulator contains 3, the line accumulator contains -7, and the pixel count is 3. Dividing the column and line values by the pixel count gives the average coordinates of the "on" pixels, (3/3,-7/3). In other words, column 1, row -2.333 is the location of the centroid.

B. CIRCUIT DETAILS

The hardware for the simulation consists of five major sections:

1) The Prime 550 simulation computer;

2) The video processing electronics;

3) The computer terminal;

4) The General Electric TN-2000 charge-injection device (CID) television camera;

5) The 6-degree-of-freedom simulator.

The simulation computer sends servo commands to the simulator and control signals to the video processing electronics. The camera sends video imagery and timing signals to the video electronics, which, in turn, sends its analysis of the imagery to the simulation computer and passes normal text from the computer to the terminal. The operator uses the terminal to type commands to the computer. The interconnections are shown in Figure VI-3.

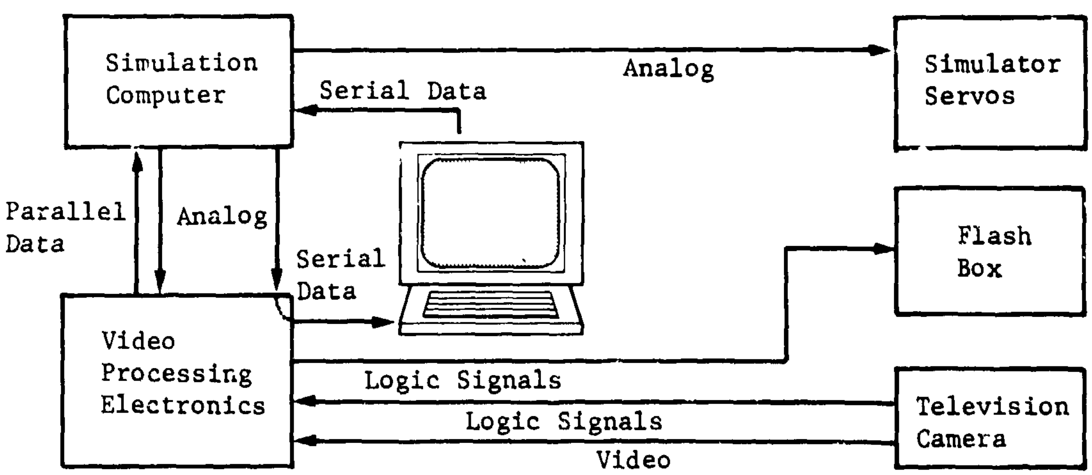


Figure VI-3 System Interconnections

Only the video electronics was designed for this contract; it is explained here in detail.

The video electronics comprises eight sections:

1) Microprocessor;

2) Synchronization with camera;

3) Comparator;

4) Counters;

5) Accumulators;

6) Multiplexer;

7) Drivers;

8) High-voltage flash box.

At the hub of the communications between the video electronics and the simulation computer is an MC68701 microprocessor. Its job is to pass data between the computer and the terminal. It also intercepts data intended for the video electronics by recognizing special characters in the text.

A separate section of the video electronics synchronizes the requests for action intercepted by the microprocessor with the start of a video image from the camera.

A comparator converts the video signal from analog to digital form. The reference voltage used in the comparison is generated by the simulation computer and varies during a simulation. The digital signal is "true" ("on") for a pixel in which a flash lamp is detected and "false" ("off") when a flash lamp is not detected.

There are three counters in the video electronics. The first counts the "on" pixels detected by the comparator. This is the "pixel" counter. The second or "column" counter counts up one count for each pixel in a line, whether the pixel is "true" or "false." This count starts at a negative value so that a count of zero occurs in the middle of each line, negative numbers occur in the left half frame, and positive numbers occur in the right half.

This third counter, or "line" counter, counts lines in the image. It starts counting from a positive value and counts down, so zero again occurs in the middle of the image. The count is positive for the top half of the frame and negative for the bottom half.

The outputs from the column and line counters are fed through adders to two accumulators, which are clocked by a pixel-rate "count load" clock signal that is synchronized with the camera. They accumulate counts from the column and line counters each time the comparator detects a

ORIGINAL PAGE IS
OF POOR QUALITY

pixel with voltage exceeding a "threshold" value; the count in the column counter is added to the column accumulator, and the count from the line counter is added to the line accumulator. Because pixel voltage normally exceeds the threshold only for pixels that represent the image of a flash lamp, the circuitry calculates the number of pixels that are in the image of the lamp, the sum of the row num'ers, and the sum of the column numbers for these pixels.

After a flash has occurred, data from the pixel counter, the column accumulator, and the line accumulator are fed to a multiplexer. The output of the multiplexer goes to differential drivers, which send these data to the simulation computer.

The high voltage flash box contains the voltage doublers and trigger circuit to fire the xenon flash lamps. The interconnections among these sections are shown in Figure VI-4.

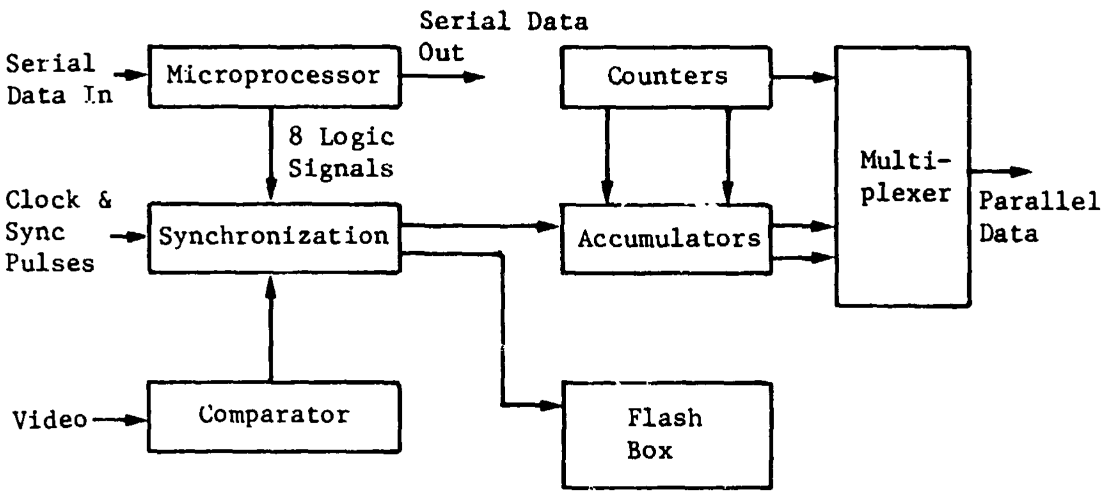


Figure VI-4 Interconnections within Processing Electronics

1. Microprocessor

The microprocessor examines the data sent from the simulation computer. If the data are intended for the operator, the microprocessor sends them to the terminal. If the data are for the video electronics, the microprocessor intercepts and directs them to the electronics. This is accomplished by preceding the data for the electronics with an ASCII "escape" character. The microprocessor checks every character sent by the computer. If the character is not an escape, the microprocessor sends it to the terminal. If an escape is detected, the next character received is sent to the electronics through the microprocessor's parallel output port, P4. Port 2, bit 3, is used as a serial input port from the computer, and port 2, bit 4, is used as a serial output port to the terminal. An MC1488 and MC1489 convert between logic circuit voltage levels and RS-232C levels so that the simulation computer can communicate with the microprocessor. The configuration is shown in Figure VI-5, and a flow chart of the microprocessor firmware appears in Figure VI-6.

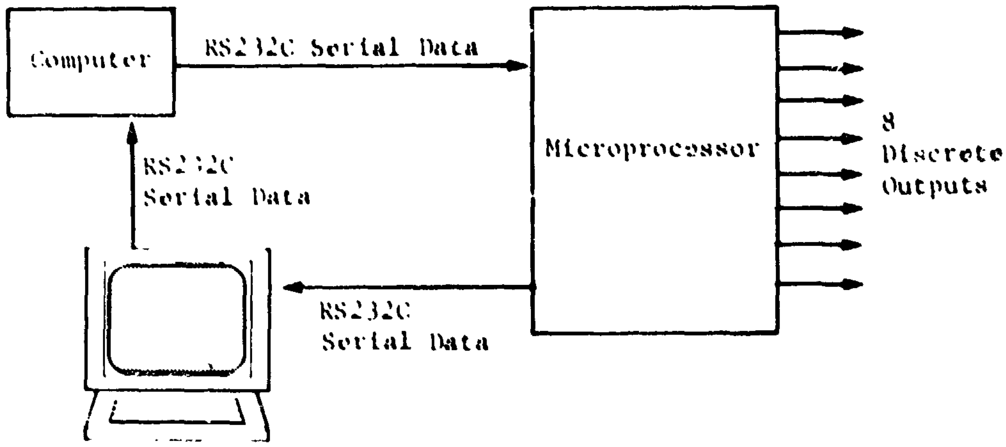


Figure VI-5 Scheme for Extracting Commands from Serial Line

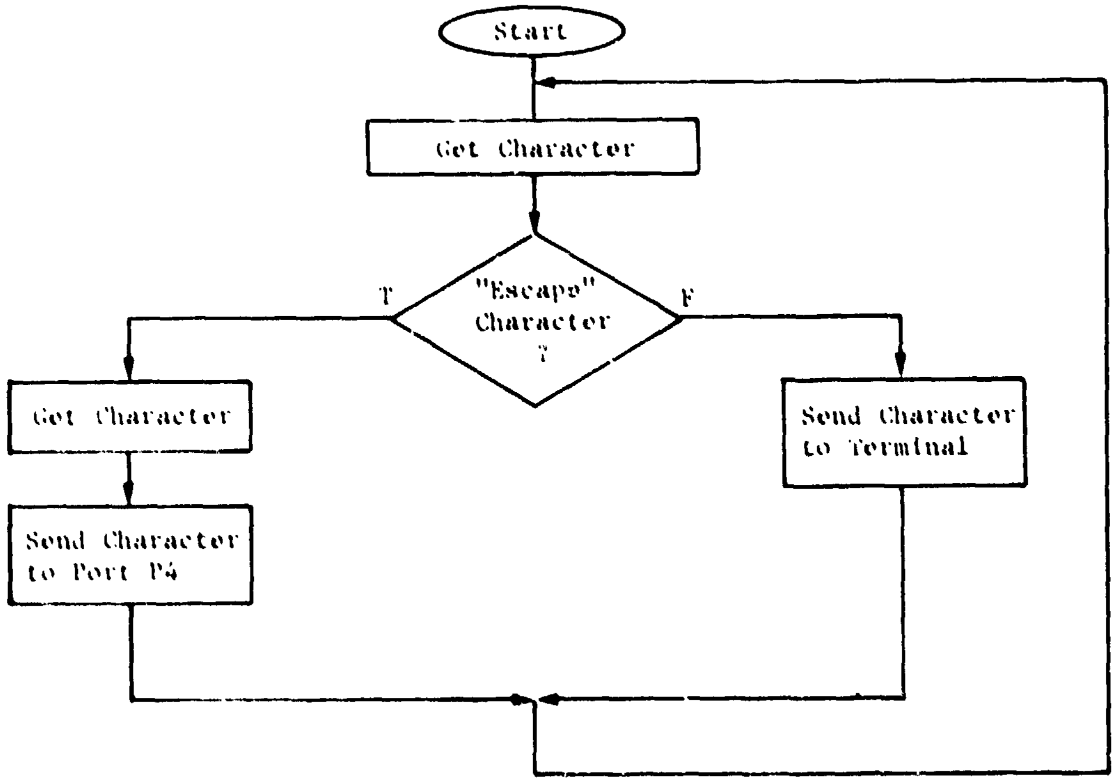


Figure VI-6 Microprocessor Firmware Flow Chart

ORIGINAL PAGE IS
OF POOR QUALITY

All eight bits of the P4 port are used. The assignments are shown in Table VI-1. The signal names will be explained as their destinations are discussed.

Table VI-1
Uses of Bits on Port P4

Bit Number	Signal Name
0	ONE
1	TWO
2	THREE
3	REQ
4	NEXT
5	ZERO
6	CMUX
7	GOTIT

The microprocessor circuitry is shown on Sheet 1 of the schematic diagrams in Appendix B, and the firmware for the microprocessor is presented in Appendix C.

2. <u>Synchronization</u>

There are several asynchronous events that must by synchronized in order for the data to be valid:

1) Start:

- Start of a frame of video imagery,
- Request for a lamp to flash.

2) Middle:

- Counters,
- Comparator.

3) End:

- Last load,
- Data valid,
- Reset arithmetic/logic unit,
- Change multiplexer,
- Reset multiplexer.

The video picture consists of two half frames or fields, which are interlaced to create the picture. In a broadcast-quality camera, these fields are slightly different: the first field contains the odd-numbered lines of a 488-line image, and the second field contains the even-numbered lines. The phasing between the horizontal and vertical sync pulses is different for the two fields: the horizontal pulses in the second field are delayed by half the time required to scan a line so that a television set will "paint" the lines of this field between the lines of the first field. However, the camera used in the simulation does not have the resolution of a studio camera. It has only 122 lines of detector elements and must rescan each line to create a 244-line field. To create a second field, it repeats the first field with the half-line delay to simulate scan interlacing. This makes the camera compatible with a standard television signal, but the second field contains no information that was not in the first field. Because all the information is contained in each field, the video electronics uses only the first field.

3. Start Synchronization

To ensure that the same half frame is used in each image analysis, a singleshot is used to shorten the vertical sync pulse from the camera. The shortened pulse is "ANDed" with the horizontal sync pulse. When the start of a frame occurs, the "ANDed" signal goes high. This signal is inverted and used to set a 74279 set-reset latch. The latch is cleared on the next occurrence of a vertical sync pulse. The setting of the latch is prevented during the start of the second half frame by the shortening effect of the singleshot (Figure VI-7). The result is that the electronics will always wait for the proper half frame.

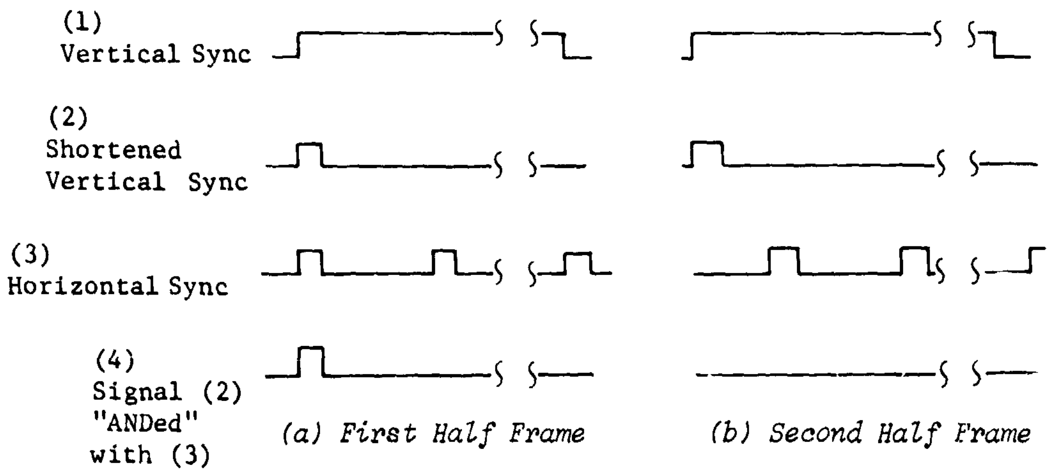


Figure VI-7 Timing of Signals Used to Identify First Field

The request for flash must also be synchronized with the start of a frame. There are five signals associated with the flash request. Four of these are the flash lamp numbers 0 to 3 (0 indicates "no flash"), and the fifth is a "flash request." These five signals come from the microprocessor, which receives them from the simulation computer. The flash request is delayed by a singleshot and then used to latch the flash lamp numbers into the "flash latch." Only one lamp number is active (high) at a time. The outputs of the flash latch are sent to four two-input AND gates. The other input to the gates comes from the "ANDed" start-of-frame signal. The outputs from the AND gates go to drivers, which send the signals to the high-voltage flash circuit for the flash lamps. The AND gate outputs also go to two OR gates and an "exclusive OR" gate. The output of the exclusive OR sets the circuitry of counters and accumulators into action by triggering a singleshot whose output sets a count latch and clears the flash latch.

4. Middle Synchronization

The column counter and the line counter must be synchronized with the camera. This synchronization requires two logic signals sent from the camera, the column clock and the line clock. These signals increment the column counter and decrement the line counter, respectively. Because these signals are active even when they are not needed, the load signals CCNTLD and LCNTLD are used to control them. CCNTLD, the signal resulting from "ANDing" the count latch and the horizontal sync, presets the column counter at the start of each line and disables the counter when a flash is not in progress. LCNTLD, the the count latch signal "ANDed" with the vertical sync, presets the line counter at the start of each frame and disables the counter when a flash is not in progress.

A comparator compares the video signal and a reference voltage. The output of the comparator is a logic signal that is zero when the video signal is above the reference voltage. This signal is inverted and disables the reset on a pixel latch. The set signal to set the latch comes from the column clock. The output of the pixel latch is put into an inverter with a capacitor and an AND gate. The configuration of latch, inverter, capacitor, and AND gate prevents multiple counts or missed counts of thresholded pixels. The output of the AND gate is "ANDed" with the output of the count latch to produce a load pulse. This pulse increments the pixel counter and loads the column accumulator and line accumulator with the current values in the column and line counters, respectively. This occurs every time the video contains a pixel with voltage that exceeds the reference value.

5. End Sync

After half a frame, the vertical sync occurs, and a number of events occur:

1) The count latch is cleared, which causes

2) The signal DATACLK to clock the final data into the output registers of the accumulators and pixel count register;

3) "DATA VALID" is set;

4) After a delay through a singleshot, the signal RESET ALU becomes low.

The simulation computer waits for the data valid signal after issuing the request for a flash. When the data are valid, the count of pixels is read. The computer then issues a NEXT command to the microprocessor, which passes it on to the multiplexer address counter. The counter increments and the outputs of the counter are used to select the next inputs for the multiplexer. Then the data in the column accumulator are read by the computer. NEXT is issued again, and the line data are read. NEXT is issued one more time to:

1) Clear the accumulators;

2) Clear the pixel counter;

3) Clear "DATA VALID."

Finally, CMUX is issued to reset the multiplexer address counter for the next sequence.

Throughout the flash sequence, the signal "GOTIT" is toggled for each request from the computer. This signal is produced by the computer and is fed back to the computer to establish bidirectional handshaking between the video electronics and the computer.

6. Comparator

The comparator, an LM319, compares the video signal to a reference voltage from digital-to-analog converter channel 8 of the simulation computer. Because the voltage has a +/-10 volt range and the video signal cannot exceed one volt, the reference voltage is divided by two resistors to produce a range of +/-1 volt at the comparator.

7. Counters

The three counters are 74LS169's. The column counter counts from -103 to +93. The line counter counts from 122 down to -121. Only eight bits are required to count in these ranges, so two 74LS169's were used for each counter. The pixel counter must be able to count every pixel if all are above the reference voltage. Because there are 188 pixels in a line and 244 lines, 45,872 is the maximum count for this counter. Five 74LS169's were used for the pixel counter to allow for possible future use of both video fields.

8. Accumulators

The counts in the column and line counters must be accumulated every time the comparator detects an "on" pixel. Because the counts may be either positive or negative, the greatest possible total would accumulate if half the columns and half the lines were counted. This is a count of 122 columns and 94 lines. Counting every pixel for this case yields a total possible count of 1,089,460 for the column accumulator and 110,564 for the line accumulator, or 22 bits and 21 bits, respectively, including the sign bit. To meet this requirement, six AM2517 four-bit arithmetic/logic units (ALU) were combined with three 74LS377 registers to make a 24-bit adder-accumulator. The signal "RESET ALU" controls the ALU function: if a reset is to be performed, the function is "F=0"; otherwise the function is "A+B". The data from the counters are added into the A inputs with the sign bit extended. The output of the 74LS377 registers is fed into the B inputs and into the inputs of the data multiplexers.

9. Multiplexers

A four-input multiplexer, comprising twelve 74LS153 dual four-input multiplexer chips, selects the data to be sent to the computer. Table VI-2 shows the select codes that control the multiplexer.

Table VI-2 Multiplexer Data Addressing

Control Code	Data Sent to Simulation Computer
00	Pixel Count
01	Column Accumulator
10	Line Accumulator
11	(Unused)

A 74LS169 counter is used to select the multiplexer. It is controlled by the NEXT and CMUX outputs of the microprocessor. The NEXT data line clocks the counter, and the CMUX line either enables the counter to count or loads a zero to reset it.

10. Drivers

The data from the multiplexer are sent to twelve 8830 dual differential drivers that send the data to the computer. A thirteenth 8830 is used to send DATA VALID and GOTIT to the computer.

11. High-Voltage Flash Box

The flash lamps used are xenon bulbs, which require an anode voltage between 200 and 400 volts. The circuit to supply this voltage and trigger the lamps is shown in Appendix B. For safety, the circuit is isolated from the power line with an isolation transformer. A voltage doubler and peak detector convert the isolated 115 volts ac to 300 volts dc, the anode voltage for the lamps.

A voltage divider is used to derive 170 volts dc from the 300 volts to charge a 0.25 μf capacitor. A silicon controlled rectifier (SCR) is wired across the capacitor and a trigger coil, which has a 30:1 turns ratio. When the flash request is sent, the SCR shorts the capacitor and coil, producing a high trigger voltage to flash the lamp.

VII. SIMULATOR CONTROL

During the simulation, the 6-DOF simulator adjusts the position and attitude of the simulation television camera so that it will have the same view of the target as a camera mounted on a real chase vehicle. The simulator servos are commanded to new positions and are allowed to settle before each flash of the docking aid lights. Because the camera makes no observations between flashes, it was not necessary to simulate continuous motion.

The main subroutine for simulator control is POINT. The subroutine calls two others to calculate the required translational and rotational position of the camera. A third subroutine converts these commands from units of meters and radians to servo command voltages. This routine also verifies that the requested position can be reached and calculates how long the servos will take to settle at the new position. It then sends the voltages to the servos through digital-to-analog converters.

A. SUBROUTINE SIMXLT COMPUTES TRANSLATIONAL POSITION

Subroutine SIMXLT determines where the camera should be positioned by evaluating the formula

$$[37] \qquad \underline{c} = \underline{\ell}_{ts} + A_{st}\left(A_t\left[s\underline{x} + A_c^t(s\underline{h}_c - \underline{\ell}_{lens})\right] - s\underline{h}_t\right) ,$$

where

$\underline{c}$, represented in the program as SIMXYZ, is the position of the camera in the simulator's coordinate system;

$\underline{\ell}_{ts}$, represented in the program as LTS, is the position of the docking aid on the target model, expressed in the simulator coordinate system;

A_{st}, represented in the program as AST, is the direction cosine matrix that describes the simulator's attitude with respect to the target model;

A_t, represented in the program as AT, is the direction cosine matrix that describes the target's attitude with respect to the "truth" coordinate system;

s, represented in the program as SCALE, is the model scale (0.01, 0.1, or 1.0);

ORIGINAL PAGE IS
OF POOR QUALITY

$\underline{x}$, represented in the program as the first three elements of the array STATE, is the chase vehicle's position in the "truth" coordinate system;

A_c^t, represented in the program as TRNACV, is the transpose of the direction cosine matrix that describes the chase vehicle's attitude with respect to the "truth" coordinate system;

$\underline{\ell}_{lens}$, represented in the program as LLENS, is the offset from the center of the simulator's gimbal set to the effective center of the camera lens;

$\underline{h}_c$ and $\underline{h}_t$, represented in the program as HC and HT, are the positions of the camera and docking aid in the coordinate systems of the chase vehicle and the target, respectively.

The values of s, A_{st}, $\underline{\ell}_{ts}$, and $\underline{\ell}_{lens}$ are constant within a simulation phase, but change between phases. The variables A_t, A_c^t, and $\underline{x}$ change as the simulated spacecraft move. The vectors $\underline{h}_c$ and $\underline{h}_t$ depend only on the design of the two spacecraft and do not change.

Figure VII-1 illustrates the physical interpretation of Eq 37.

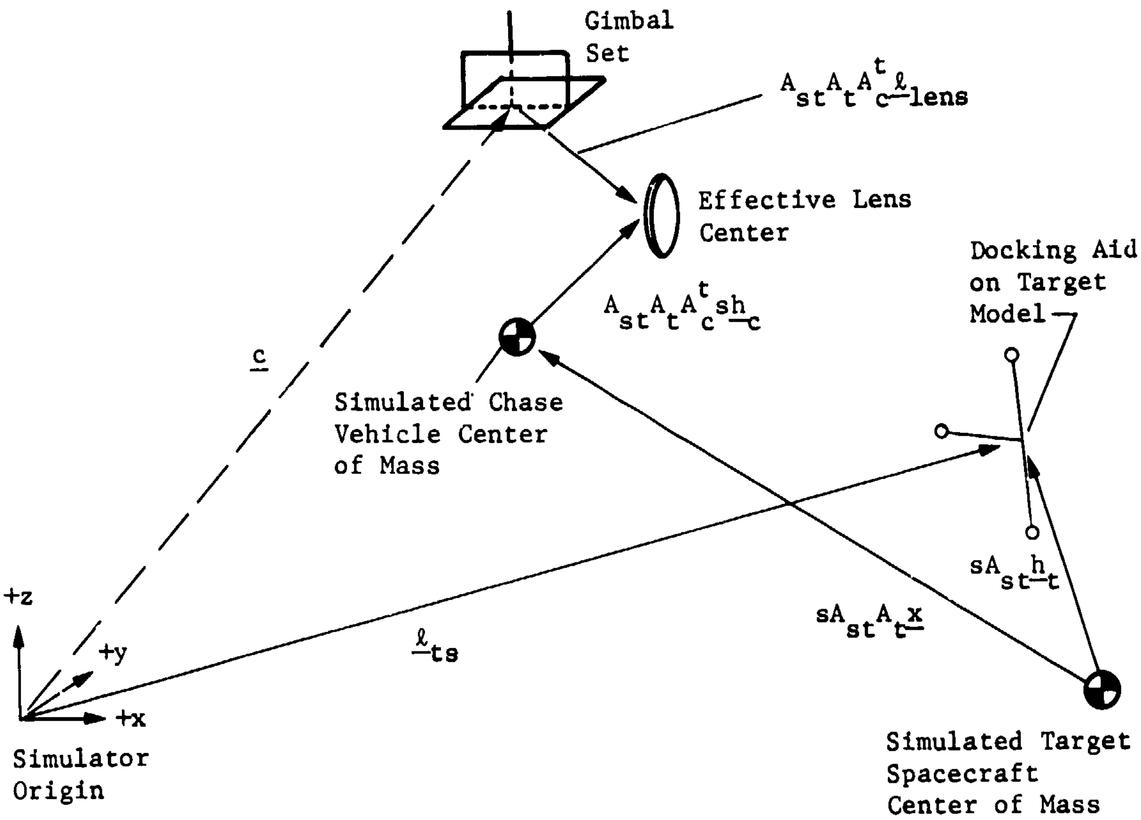


Figure VII-1 Physical Interpretation of Equation 37

B. SUBROUTINE SIMROT COMPUTES GIMBAL ANGLES

Subroutine SIMROT computes the attitude of the chase vehicle with respect to the target spacecraft and then finds a set of gimbal angles that will give the camera the same attitude with respect to the target model. However, the problem is more complicated than it might appear. First, the target model is not aligned with the simulator coordinate system. We pointed the docking axis of the model slightly upward in order to use the operating range of the simulator to best advantage. Second, the gimbal set is pitched down 45 degrees when all gimbal angles are zero. Mounting the gimbal set this way allowed us to operate with the small target models against the wall of the room and the large model on the floor without encountering singularities or servo limits in normal operation.

To account for these factors, the subroutine calculates

[38] $$A = A_{cv}A_t^t A_{ts}'$$

where

A, represented in the program as A, is a direction cosine matrix that describes the attitude of the camera with the required gimbal angles relative to its attitude with zero gimbal angles;

A_{cv}, represented in the program as ACV, is a direction cosine matrix that describes the chase vehicle attitude with respect to the "truth" coordinate system;

A_t^t, represented in the program as TRNAT, is the transpose of the direction cosine matrix that describes target spacecraft attitude with respect to the "truth" coordinate system;

A_{ts}', represented in the program as ATSP, is the direction cosine matrix that describes the attitude of the target model with respect to the camera's zero-gimbal-angle attitude.

Because the gimbal set provides a yaw-pitch-roll sequence, it can produce the attitude change specified by the matrix A as long as A can match the general solution of a yaw-pitch-roll direction cosine matrix element for element with realizable values of the gimbal angles. The general yaw-pitch-roll solution is

[39] $$A = \begin{bmatrix} cPcY & cPsY & -sP \\ sRsPcY - cRsY & cRcY + sRsPsY & sRcP \\ sRsY + cRsPcY & cRsPsY - sRcY & cRcP \end{bmatrix}$$

where Y, P, and R are the yaw, pitch, and roll gimbal angles, and c and s are abbreviations for sin and cos. Ordinarily,

[40] $Y = \mathrm{ATAN2}\ (a_{12}, a_{11});$

[41] $R = \mathrm{ATAN2}\ (a_{23}, a_{33});$

and either

[42] $$P = \mathrm{ATAN2}\left(-a_{13}, \frac{a_{11}}{\cos Y}\right),$$

or

[43] $$P = \mathrm{ATAN2}\left(-a_{13}, \frac{a_{12}}{\sin Y}\right),$$

where ATAN2 is the FORTRAN two-argument arc tangent function. However, alternate formulas must be used if there is a singularity. For example, if a_{11} and a_{12} are both zero, Eq 40 cannot be evaluated. Similarly, Eq 41 cannot be used if a_{23} and a_{33} are both zero, and Eq 42 cannot be used if cos(Y) = 0 or both a_{11} and a_{13} are zero.

When the problem is simply that sin(Y) or cos(Y) is zero, the subroutine has no problem, because cos(Y) can never equal zero at the same time sin(Y) equals zero. The subroutine selects either Eq 42 or Eq 43 depending on whether a_{11} or a_{12} has the larger absolute value.

The problem is more complex when there is a singularity, because there is no unique solution. To select among the infinite number of possible solutions, the subroutine adds the constraint that the new gimbal angles should match the current angles as closely as possible. The choice of this rule was arbitrary, but it does reduce the time required for the servos to settle to their new positions.

C. SUBROUTINE SERVO SENDS COMMANDS TO THE SIMULATOR

Subroutine SERVO receives the translational commands in meters and the rotational commands in radians and checks to see if any command exceeds the range over which the servos can operate. The program halts any time a servo is not able to position the camera where it should be.

If all the commands are legitimate, the subroutine uses empirical formulas to calculate the command voltages for the servos. Using a table of servo slew rates, it computes the time it will take for all the servos to settle to their new positions. It adds this interval to the time read from the computer's real-time clock and stores the result in the common block SRVOTM so that the subroutine that flashes the lights can determine when it is safe to do so.

Finally, the subroutine transmits the new servo commands through six digital-to-analog converter ports on the simulation computer.

APPENDIX A--PROGRAM LISTING

The program listing in this appendix is provided to document the simulation methods used to analyze the three-light video guidance system and to run the physical simulation. It was written to run on a Prime 550 computer under the PRIMOS operating system, but 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, which are not shown in the listing. The routines include ASIN and ACOS, which compute the inverse trigonometric functions arc sine and arc cosine. The function RANFN is a random number generator that computes normally distributed random values with a specified mean and standard deviation. The routines DINA, DINB, and DINC are digital input port interface routines, and the routine INITDI initializes the digital input ports. In addition, the matrix arithmetic routines MADD (addition), MSUB (subtraction), MMLT (multiplication), MINV (matrix inversion), MSCL (multiplication by a scalar), MIDN (setting an array equal 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, and DELE$A) are from the Prime library APPLIB.

3) Run time is approximately one-sixth of real time if the computer is dedicated to one user. The servo settling time is the primary factor affecting speed.

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 text. Removing this character will provide a printout at the operator's terminal for monitoring the progress of the simulation.

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

```
OK, [illegible]
OK, [illegible] PHYSSIM -DCLVAR
0000 [illegible] MAIN   FTN-REV17 43
0000 [illegible] OPEN   FTN-REV17 43
0000 [illegible] INIPAR FTN-REV17 43
0000 [illegible] INISIM FTN-REV17 43
0000 [illegible] ELECIN FTN-REV17 43
0000 [illegible] DOCK   FTN-REV17 43
0000 [illegible] FLASHM FTN-REV17 43
0000 [illegible] ESTRPY FTN-REV17 43
0000 [illegible] POSIT  FTN-REV17 43
0000 [illegible] D      FTN-REV17 43
0000 [illegible] V      FTN-REV17 43
0000 [illegible] N      FTN-REV17 43
0000 [illegible] P      FTN-REV17 43
0000 [illegible]        FTN-REV17 43
0000 [illegible] V      FTN-REV17 43
0000 [illegible] AN     FTN-REV17 43
0000 [illegible] A      FTN-REV17 43
0000 [illegible] V      FTN-REV17 43
0000 [illegible] S      FTN-REV17 43
0000 [illegible] L      FTN-REV17 43
0000 [illegible] T      FTN-REV17 43
0000 [illegible] W      FTN-REV17 43
0000 [illegible]        FTN-REV17 43
0000 [illegible] 1      FTN-REV17 43
0000 [illegible] 1      FTN-REV17 43
0000 [illegible] 2      FTN-REV17 43
0000 [illegible] 3      FTN-REV17 43
0000 [illegible] R      FTN-REV17 43
0000 [illegible]        FTN-REV17 43
0000 [illegible]        FTN-REV17 43
0000 [illegible] DIRMAT FTN-REV17 43
0000 [illegible] STPRIM FTN-REV17 43
0000 [illegible] FORCF  FTN-REV17 43
0000 [illegible] MPRIMF FTN-REV17 43
0000 [illegible] LINACL FTN-REV17 43
0000 [illegible] TORQUE FTN-REV17 43
0000 [illegible] LPRIME FTN-REV17 43
0000 [illegible] MAKROT FTN-REV17 43
0000 [illegible] QPRIME FTN-REV17 43
0000 [illegible] SGR1   FTN-REV17 43
0000 [illegible] ANGVEC FTN-REV17 43
0000 [illegible] TRGATT FTN-REV17 43
0000 [illegible] POINT  FTN-REV17 43
0000 [illegible] SIMXLT FTN-REV17 43
0000 [illegible] SIMROT FTN-REV17 43
0000 [illegible] SERVO  FTN-REV17 43
0000 [illegible] VREF   FTN-REV17 43
0000 [illegible] FLASH  FTN-REV17 43
0000 [illegible] GOTIT  FTN-REV17 43
0000 [illegible] DINAA  FTN-REV17 43
0000 [illegible] DINBA  FTN-REV17 43
0000 [illegible] PROPTR FTN-REV17 43
0000 [illegible] COMPK1 FTN-REV17 43
0000 [illegible] COMPK2 FTN-REV17 43
0000 [illegible] COMPK3 FTN-REV17 43
0000 [illegible] COMPK4 FTN-REV17 43
0000 [illegible] ESECS  FTN-REV17 43
0000 [illegible] DINCA  FTN-REV17 43

OK, LOAD
$ LO B_MSFC PHYSSIM
$ LO 'TIETZ>WORKING NERD>B_RANF
$ LO 'TIETZ>WORKING NERD>B_ASIN
$ LI APPLIB
$ LI MATHLB
$ LI LINK
$ LI
LOAD COMPLETE
$ SA *PHYSSIM
$ QU

OK  CO -TTY
OK, R *PHYSSIM
ENTER PHASE
```

```
   1  --> MAX RANGE, SMALL MODEL
   2  --> MED RANGE, MED MODEL
   3  --> MIN RANGE, LARGE MODEL
1
ENTER PATHNAME FOR OUTPUT
DOK15P1
 ENTER THREE ROTATION ANGLES TO SPECIFY INITIAL TARGET
 ATTITUDE FOR A YAW-PITCH-ROLL SEQUENCE---ANGLES MUST BE
 IN DEGREES
0
0
0

 ENTER TUMBLE AXIS-- ENTER 1, 2, OR 3 FOR YAW, PITCH,
 OR ROLL, RESPECTIVELY
1

 ENTER TUMBLE RATE IN DEGREES PER HOUR
500
 INPUT 16-DIGIT RANDOM INTEGER
6472340981234769
HIT CARRIAGE RETURN WHEN READY FOR SIMULATION
OK,
```

ORIGINAL PAGE IS OF POOR QUALITY

SIMULATION PROGRAM LISTING


```
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 W R T  '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
```

SIMULATION PROGRAM LISTING


```
C
C            ROUTINE  PAGE ROUTINE-NBR
C
C            <MAIN>     3    000
C            ACCEL     40    481 1
C            ANGVEC    62    904
C            ATITUD    24    440
C            CNTLAW    39    481
C            COMPG     29    451
C            COMPK1    79    909 1
C            COMPK2    80    909 2
C            COMPK3    81    909 3
C            COMPK4    82    909 4
C            DINAA     76    908 2
C            DINBA     77    908 3
C            DINCA     84    911
C            DIRMAT    52    901
C            DOCK      15    400
C            ELECIN    14    310
C            ESECS     83    910
C            ESTCOV    30    452
C            ESTRPY    21    420
C            FINDCV    25    441
C            FIRTHR    49    480
C            FLASH     72    908
C            FLASHM    19    410
C            FORCE     54    902 1
C            GOTIT     74    908 1
C            IMU       50    490
C            INCORP    28    450
C            INIPAR     6    200
C            INISIM    11    300
C            KALGAN    31    453
C            LINACL    56    902 3
C            LPRIME    58    902 5
C            MAKROT    59    902 6
C            MPRIME    55    902 2
C            OPEN       4    100
C            POINT     65    906
C            POSIT     22    430
C            PROPES    34    460
C            PROPTR    78    909
C            QPRIME    60    902 7
C            QUATRN    26    442
C            RPY       51    4A0
C            SELECT    42    482
C            SERVO     69    906 3
C            SETGOL    36    470
C            SIMROT    67    906 2
C            SIMXLT    66    906 1
C            SGR1      61    903
C            STPRIM    53    902
C            TABLE1    43    482 1
C            TABLE2    45    482 2
C            TABLE3    47    482 3
C            THRUST    38    480
C            TORQUE    57    902 4
C            TRGATT    63    905
C            UPDCOV    33    455
C            UPDSTA    32    454
C            VREF      71    907
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C     PROGRAM MSFC PHYSSIM - PHYSICAL SIMULATION OF RENDEZVOUS AND DOCKING
C        OPERATION USING THE VIDEO GUIDANCE LABORATORY SIMULATION AT
C        MARTIN MARIETTA DENVER AEROSPACE'S ACS DEVELOPMENT LABORATORY
C        BUILDING     THIS PROGRAM IS WRITTEN FOR A PRIME 550 COMPUTER
C        AND USES SEVERAL SUBROUTINES THAT ARE INSTALLATION DEPENDENT
C
C     CALLS
C        CLOS$A, DOCK, EXIT, INIPAR, INISIM, OPEN, YSNO$A
C
      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
      INTEGER PHASE
C        SIMULATION PHASE (1 --, LONG RANGE      .3 --, CLOSEUP )
      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$A
C        FUNCTION TO GET YES/NO ANSWER FROM USER AT TERMINAL
C
      COMMON/SIMUL/TERMNL, OUT
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      OUT=6
      TERMNL=1
C     (* LOOP UNTIL USER WANTS NO MORE SIMULATIONS *)
   10 CONTINUE
C        (* 100 - OPEN DATA FILE *)
         CALL OPEN(FILERR, TERMNL, OUT, PHASE, ESTATE, P, T, STATE)
C        (* 200 - INITIALIZE PROBLEM PARAMETERS *)
         CALL INIPAR(T, STATE, P, ESTATE, F, PHASE)
C        (* 300 - INITIALIZE SIMULATOR *)
         CALL INISIM(T, STATE, PHASE)
      DOCKED=.FALSE.
C     (* LOOP WHILE NOT DOCKED *)
   20 IF (DOCKED) GO TO 30
C        (* 400 - SIMULATE OBSERVATION & INTERVAL BETWEEN OBSERVATIONS *)
            CALL DOCK(P, ESTATE, STATE, T, DOCKED, F, PHASE)
         GO TO 20
   30 CONTINUE
C     (* CLOSE DATA FILE *)
         CALL CLOS$A(OUT-4)
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, PHASE, ESTATE, P, T, STATE)
C     (* 100 - OPEN FILES FOR OUTPUT   WARNING   HIGHLY INSTALLATION DEPENDENT *)
C
C     CALLS
C        DELE$A, EXST$A, TSRC$$
C     CALLED BY
C        (MAIN)
C
C     INPUT
C        TERMNL, OUT
C     OUTPUT
C        FILERR, TRNATO, ITUMBL, TUMRAT, PHASE, ESTATE, P, T, STATE
C
      INTEGER CHRPOS(2)
C        CHARACTER POSITION/COUNT CODES FOR SYSTEM ROUTINE TSRC$$
      INTEGER CODE
C        ERROR CODE RETURNED BY TSRC$$ (=0 IF NO ERROR)
      REAL ESTATE(6)
C        STATE ESTIMATE
      INTEGER I
C        DO LOOP INDEX
      INTEGER ITUMBL
C        CODE INDICATING WHICH AXIS THE TARGET TUMBLES ABOUT
      INTEGER MSG(20), MSGLEN
C        MESSAGE TEXT AND NUMBER OF CHARACTERS IN THE TEXT
      INTEGER OLDPHS, PHASE, PRVPHS
C        SIMULATION PHASES   LAST ONE ON DISK, CURRENT, AND WHAT SHOULD
C        BE THE LAST ONE ON THE DISK AT THIS POINT
      INTEGER OUT, TERMNL
C        FORTRAN LOGICAL UNIT NUMBER FOR OUTPUT
      REAL P(6,6)
C        COVARIANCE ASSOCIATED WITH ESTATE
      INTEGER PATH(16)
C        ARRAY OF ASCII DEFINING PATH NAME
      REAL STATE(14)
C        TRUE STATE OF CHASE VEHICLE
      REAL T
C        SIMULATION TIME, SECONDS
      REAL TADUT(3,3)
C        TARGET ATTITUDE FROM DISK
      INTEGER TERMNL
C        FORTRAN LOGICAL UNIT NUMBER OF USER TERMINAL
      REAL TRNATO(3,3)
C        TRANSPOSE OF INITIAL TARGET ATTITUDE DIRECTION COSINE MATRIX
      REAL TUMRAT
C        TARGET SPACECRAFT TUMBLE RATE, RADIANS PER SECOND
      INTEGER TYPE
C        TYPE CODE RETURNED BY TSRC$$ (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
      COMMON/ATINFO/TRNATO, ITUMBL, TUMRAT
C
      DATA MSG/'*** FILE ALREADY EXISTS   OK TO OVERWRITE'/
      DATA MSGLEN/40/
```

```
C
      CHRPOS(2)=32
      TYPE=0
      FILERR=.FALSE.
   10 CONTINUE
         CALL CLOS$A(OUT-4)
         WRITE(TERMNL,901)
         READ(TERMNL,902,ERR=10) PHASE
         IF(PHASE.LT.1.OR.PHASE.GT.3) GO TO 10
         WRITE(TERMNL,903)
         READ(TERMNL,904) (PATH(I),I=1,16)
         IF(PHASE.EQ.1) GO TO 60
            IF(.NOT.EXST$A(PATH,32)) GO TO 50
               CHRPOS(1)=0
               CALL TSRC$$(K$RDWR+K$NSAM,PATH,OUT-4,CHRPOS,TYPE,CODE)
               IF(CODE.NE.0) GO TO 80
               READ(OUT,ERR=40,END=40) T,STATE,TRNATO,OLDPHS,ESTATE,P,
     A            TUMRAT,ITUMBL
   20          CONTINUE
                  READ(OUT,ERR=40,END=30) T,STATE,TADOUT,OLDPHS,
     A               ESTATE,P
                  GO TO 20
   30          CONTINUE
               PRVPHS=PHASE-1
               IF(OLDPHS.EQ.PRVPHS) RETURN
               WRITE(TERMNL,907) OLDPHS,PRVPHS,PHASE
               GO TO 10
   40       CONTINUE
            WRITE(TERMNL,905)
            GO TO 10
   50       CONTINUE
               PRVPHS=PHASE-1
               WRITE(TERMNL,906) PRVPHS,PHASE
               GO TO 10
   60    CONTINUE
            IF(.NOT.EXST$A(PATH,32)) GO TO 70
               IF(.NOT.YSNO$A(MSG,MSGLEN,-1)) GO TO 10
               CALL DELE$A(PATH,32)
   70       CONTINUE
            CHRPOS(1)=0
            CALL TSRC$$(K$WRIT+K$NSAM,PATH,OUT-4,CHRPOS,TYPE,CODE)
            IF(CODE.NE.0) GO TO 80
            RETURN
C
   80 CONTINUE
      FILERR=.TRUE.
      RETURN
C
  901 FORMAT('ENTER PHASE'/'    1 --> MAX RANGE, SMALL MODEL'/
     A  '    2 --> MED RANGE, MED MODEL'/'    3 --> MIN RANGE, LARGE'
     B  ' MODEL')
  902 FORMAT(I5)
  903 FORMAT('ENTER PATHNAME FOR OUTPUT')
  904 FORMAT(16A2)
  905 FORMAT('THE FILE IS EMPTY OR UNREADABLE.  I CANNOT DO THIS '
     A  'PHASE WITHOUT A GOOD FILE FROM THE PREVIOUS PHASE')
  906 FORMAT('THE FILE DOES NOT EXIST.  I MUST HAVE A FILE WITH DATA'
     A  ' FROM PHASE',I2/'IF I AM TO DO PHASE',I2,'.')
  907 FORMAT('THE LAST ITEM ON THE FILE IS FROM PHASE',I2,'.'/
     A  'THE FILE SHOULD END IN DATA FROM PHASE',I2,' IF I AM'
     B  ' TO DO PHASE',I2,'.')
C
      END
```



```
C
      SUBROUTINE INIPAR(T,STATE,P,ESTATE,F,PHASE)
C*    200 - INITIALIZE PROBLEM PARAMETERS *
C
C     CALLS
C        SQRT, RANEN
C     CALLED BY
C        MAIN
C
C     INPUT
C        PHASE
C     OUTPUT
C        AW1, AW2, AW3, DCKRAD, ESTATE, FOCLEN, FULDIS, FI, INERCV, MEMPTY, P,
C        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 AW1(14),AW2(3,14),AW3(3,14)
C        MATRICES SPECIFYING CHASE VEHICLE
C           AW1 GIVES FUEL USE PER NEWTON-SECOND OF THRUST FOR EACH THRUSTER
C           AW2 GIVES FORCE DIRECTION FOR EACH THRUSTER
C           AW3 GIVES TORQUE/NEWTON & TORQUE DIRECTION FOR EACH THRUSTER
      REAL DCKRAD
C        RADIUS FROM TARGET SPACECRAFT WITHIN WHICH THE SPACECRAFT ARE
C        CONSIDERED DOCKED
      REAL ESTATE(6)
C        ESTIMATE OF STATE VECTOR
      REAL FOCLEN, FL(3)
C        LENS FOCAL LENGTH IN METERS & INIT. VALUES FOR 3 PHASES
      REAL FULDIS(3,3)
C        CHASE VEHICLE FUEL-DISTRIBUTION INERTIA TENSOR
      REAL F(6)
C        FORCES FROM THRUSTERS AFTER SELECTION
      REAL FI(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 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
      INTEGER PHASE
C        SIMULATION PHASE (1 --> LONG RANGE, ..., 3 --> CLOSE UP)
      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 RADTBL(3)
C        SEPARATION BETWEEN DOCKING FIXTURES THAT FLAGS END OF A SIMU-
C        LATION PHASE
      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
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
      COMMON/OPTSYS/FOCLEN,A,B
      COMMON/VEHIC/AK1,AK2,AK3,DOKRAD,F1
      COMMON/SIMUL/TERMNL,OUT
      COMMON/HNGOFF/HC,HT,HDC,HDT
      COMMON/R/F/30
      COMMON/A:INFO/TRNATO,ITUMBL,TUMRAT
C
      DATA FLO/ 025,  010,  010/
      DATA RADTBL/35 ,  85,0 4/
```



```
C
      IF(PHASE NE 1) GO TO 80
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)*
     A         SIN(PHI)
            TRNATO(2,2)=SIN(PHI)*SIN(THETA)*SIN(PSI)+COS(THETA)*
     A         COS(PHI)
            TRNATO(3,2)=SIN(THETA)*COS(PSI)
            TRNATO(1,3)=COS(THETA)*SIN(PSI)*COS(PHI)+SIN(THETA)*
     A         SIN(PHI)
            TRNATO(2,3)=SIN(PHI)*COS(THETA)*SIN(PSI)-SIN(THETA)*
     A         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
         T=0 0
         DO 60 I=1,6
            DO 60 J=1,6
               P(I,J)=0
   60    CONTINUE
         P(1,1)=400
         P(2,2)=100
         P(3,3)=100
         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
   80 CONTINUE
      A=1
      B=1
      DOKRAD=RADTBL(PHASE)
      FOCLEN=FLO(PHASE)
      F1(1)=360
      F1(2)=80
      F1(3)=80
      F1(4)=80
      F1(5)=80
      F1(6)=20
      F1(7)=20
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
      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
      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.
      GO(1)=STATE(10)
      GO(2)=STATE(11)
      GO(3)=STATE(12)
      GO(4)=STATE(13)
      HC(1)=2 0
      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
```



```
      IF(PHASE EQ 1) WRITE(OUT) T,STATE,TRNATO,PHASE,ESTATE,P,
     A   TUMRAT,ITUMBL
      RETURN
C
  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

```
  (ARGUMENTS (SIMINIT STATE PHASE)
   PROCEDURE INITIALIZE SIMULATOR *)

 CALLS
    POINT, ELECT
 CALLED BY
    MAIN

 INPUT
    STATE PHASE
 OUTPUT
    LTS, ATSP, AST, LLENS, ROLMIN, ROLMAX, YAWMIN, YAWMAX, SCALE, SIMYPR

 REAL AST(3,3), ATSPO(3,3,3)
    DIRECTION COSINE MATRIX GIVING ATTITUDE OF SIMULATOR WITH RESPECT
    TO TARGET MODEL, AND INITIAL VALUES FOR THIS MATRIX   (IN ASTO,
    THE THIRD SUBSCRIPT DENOTES SIMULATION PHASE NUMBER (1,2, OR 3) )
 REAL ATSP(3,3), ATSPO(3,3,3)
    DIRECTION COSINE MATRIX GIVING ATTITUDE OF TARGET MODEL WITH
    RESPECT TO THE CAMERA S ZERO-GIMBAL-ANGLE ATTITUDE   INCLUDES
    EFFECTS OF MISALIGNMENT BETWEEN CAMERA AND SIMULATOR WHEN
    GIMBAL SET IS ZEROED
 INTEGER
    DO LOOP INDICES
 REAL LLENS(3), LLENSO(3,3)
    OFFSET OF EFFECTIVE LENS CENTER FROM INTERSECTION OF GIMBAL
    AXES   VECTOR IS IN COORDINATE SYSTEM ALIGNED WITH LENS AND
    CENTERED AT THE AXIS INTERSECTION
 REAL LTS(3), LTSO(3,3)
    POSITION OF DOCKING AID ON TARGET MODEL IN SIMULATOR COORDI-
    NATE SYSTEM
 LOGICAL NEWRUN
    FLAGS FACT THAT THIS IS A NEW RUN AND THERE SHOULD BE NO DELAY FOR
    SERVO SETTLING TIME
 REAL PCHMIN, PCHMAX, PCHMNO, PCHMXO
    LIMITS OF TRAVEL OF PITCH GIMBAL AND INITIAL VALUES
 REAL OLDXYZ(3), OLDYPR(3)
    FORMER VALUES OF SIMULATOR TRANSLATION AND ROTATION POSITIONS
 INTEGER PHASE
    SIMULATION PHASE  1 --> SMALL MODEL, MAX RANGE      3 --> LARGE
    MODEL, CLOSE RANGE,
 REAL ROLMIN, ROLMAX, ROLMN, ROLMX
    LIMITS (IN RADIANS   OF SERVO FREEDOM ON ROLL GIMBAL AND INITIAL
    VALUES FOR THESE LIMITS
 REAL SCALE, SCALEO(3)
    SCALE FOR SIMULATION PHASE (1/100, 1/10  OR 1)
 REAL SIMYPR(3)
    SIMULATOR GIMBAL SET YAW, PITCH AND ROLL COMMANDS (RADIANS)
 REAL STATE(14)
    TRUE CHASE VEHICLE STATE
 REAL T
    ELAPSED TIME, SECONDS
 INTEGER   WSH
    WORD TO ACCEPT THROW-AWAY TEXT FROM TERMINAL   USED ONLY FOR
    SYNCHRONIZATION OF SIMULATION WITH OPERATIONAL CONSTRAINTS
 DOUBLE PRECISION TSETL
    TIME AT WHICH SERVOS ON SIMULATOR SHOULD BE SETTLED
 REAL XMIN, XMAX YMIN, YMAX, ZMIN, ZMAX, XMINO, XMAXO, YMINO, YMAXO, ZMINO,
A   ZMAXO
    LIMITS OF TRAVEL OF X,Y AND Z SERVOS ON SIMULATOR AND INITIAL
    VALUES
 REAL YAWMIN, YAWMAX, YAWNO, YAWXO
    LIMITS (IN RADIANS) OF SERVO FREEDOM ON YAW GIMBAL AND INITIAL
    VALUES FOR THESE LIMITS

 COMMON/SIMCAL/AST, LTS, SCALE, LLENS, ROLMIN, ROLMAX, YAWMIN, YAWMAX,
A   ATSP, SIMYPR, PCHMIN, PCHMAX, XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX
 COMMON/SRVCTM/OLDXYZ, OLDYPR, TSETL, NEWRUN

 DATA XMINO, XMAXO  19.4 75/
 DATA YMINO, YMAXO/- 92, 92/
 DATA ZMINO, ZMAXO/- 61.2 38/
 DATA PCHMNO PCHMXO/-1 605703 1 570796/
 DATA ASTO/ 9940469, 0174784, - 1075414, - 0156977, 9997258, 0173775,
```



```
A     0178157, - 0155810, 9940488,
B     9352946, - 0920360, - 3416918, 0979302, 9951933, 0 ,
C     3400494, - 0334620, 9398121,
D   0 , 0 , -1 , 1 , 0 , 0 ,
E   0 , -1 , 0 /
 DATA ATSPO/ 7789408, - 0233877, - 6903013, 0174784, 9997258,
A    - 015581, 6268542, 0011878, 7154965,
B    902966, 0692471, - 4240964, - 092036, 9951933, - 033462,
C    4197407, 0692471, 9049990,
D    707107, 707107, 0 , 0 , 0 , -1 ,
E   - 707107, 707107, 0 /
 DATA LTSO/4 93, - 388, 201.4 15, - 34, 28.2 77, -0 085, -0 48/
 DATA SCALEO/ 01, 1, 1 /
 DATA LLENSO/ 052, 0 , 0 , 09, 0 , 0 , 09, 0 , 0 /
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
   10 CONTINUE
      DO 30 I=1,3
         DO 20 J=1,3
            AST(I,J)=ASTO(I,J,PHASE)
            ATSP(I,J)=ATSPO(I,J,PHASE)
   20    CONTINUE
         LTS(I)=LTSO(I,PHASE)
         LLENS(I)=LLENSO(I,PHASE)
         SIMYPR(I)=0
         OLDXYZ(I)=0
         OLDYPR(I)=0
   30 CONTINUE
      TSETL=0 D0
      NEWRUN=.TRUE.
      SCALE=SCALEO(PHASE)
      ROLMIN=-3.665191
      ROLMAX=3.839724
      YAWMIN=-1.570796
      YAWMAX=1.623156
      XMIN=XMINO
      YMIN=YMINO
      ZMIN=ZMINO
      PCHMIN=PCHMNO
      XMAX=XMAXO
      YMAX=YMAXO
      ZMAX=ZMAXO
      PCHMAX=PCHMXO
C     (* 906 - POSITION SIMULATOR *)
         CALL POINT(T,STATE)
      WRITE(1,901)
      READ(1,902) TRASH
C     (* 310 - INITIALIZE VIDEO PROCESSING ELECTRONICS *)
         CALL ELECIN
      RETURN
C
  901 FORMAT('HIT CARRIAGE RETURN WHEN READY FOR SIMULATION')
  902 FORMAT(A2)
C
      END
```



```
C
      SUBROUTINE ELECIN
C     (* 310 - INITIALIZE ELECTRONICS AND COMPUTER INPUT PORT *)
C
C     CALLS
C        INITDI,FLASH
C     CALLED BY
C        INISIM
C
C     INPUT
C        (NONE)
C     OUTPUT
C        (NONE)
C
      INTEGER E
C        ERROR FLAG
      INTEGER*4 P,V,H
C        BIT BUCKETS FOR GARBAGE DATA FROM DUMMY FLASH
C
C     (* INITIALIZE DIGITAL INPUT PORTS ON COMPUTER *)
         CALL INITDI
C     (* 908 - DUMMY FLASH TO CLEAR GARBAGE *)
         CALL FLASH(C,P,V,H,E)
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

SIMULATION PROGRAM LISTING

PAGE 15

```
      SUBROUTINE DOCK(P,ESTATE,STATE,T,DOCKED,F,PHASE,
     *  [illegible] - RUN SIMULATION FOR ONE MEASUREMENT INTERVAL *)

      CALLS
         SETUP, INCORP, SQRT, FLASHM, PROPT[illegible], [illegible]SIT, ATITUD, PROPES, THRUST,
         [illegible], MMLT, MSUB, MADD, ESTRF[illegible], [illegible]T, ATAN2
      CALLED BY
         MAIN

      INPUT
         P, ESTATE, DOKRAD, STATE, [illegible] F, INER(V
      OUTPUT
         STATE, T, P, DOCKED, ESTATE, F

      REAL AC(3,3), ACVT(3,3)
         DIRECTION COSINE MATRIX DESCRIBING CURRENT CHASE VEHICLE ATTITUDE
         (RELATIVE TO PRIMARY REFERENCE FRAME), AND TRANSPOSE OF ACV
      REAL A[illegible]
         MEASURED DIRECTION COSINE MATRIX OF TARGET SPACECRAFT
      REAL ATTRATE(3)
         MEASURED ATTITUDE RATES ABOUT CHASE VEHICLE AXES
      REAL [illegible]
         [illegible] OF THOSE FLASHES THAT COULD BE SEEN (IF SOME COULD NOT
         BE [illegible]
      REAL [illegible]
         MEASURED CHASE VEHICLE POSITION IN PRIMARY REFERENCE FRAME
      REAL DEL[illegible]
         TIME BY WHICH STRATEGY LOGIC WANTS CHASE VEHICLE IN GOAL 'BOX'
      REAL DOKRAD
         RADIUS FROM TARGET AT WHICH CHASE VEHICLE IS CONSIDERED DOCKED
      REAL D[illegible]
         CHANGE IN CENTER-LAMP IMAGE COORDINATES BETWEEN FLASHES
      INTEGER [illegible]5)
         ERROR CODES RETURNED FROM 'FLASHM'
      REAL ESTATE(6)
         ESTIMATE OF STATE VECTOR
      REAL F(18)
         FORCES FROM CHASE VEHICLE THRUSTERS AFTER SELECTION
      REAL FOCLEN
         LENS FOCAL LENGTH
      REAL GYL, GXH, GYL, GYH, GZL, GZH
         X,Y,Z UPPER,LOWER LIMITS OF GOAL 'BOX'
      REAL HDC(3), HDT(3)
         DOCKING FIXTURES' POSITIONS IN RESPECTIVE SPACECRAFT COORDINATE
         SYSTEMS
      INTEGER I
         DO LOOP INDEX
      INTEGER NUV
         NUMBER OF LIGHT FLASHES THAT COULD BE SEEN
      INTEGER OUT
         FORTRAN UNIT NUMBER FOR OUTPUT FILE
      REAL P(6,6)
         COVARIANCE OF STATE ESTIMATE
      INTEGER PHASE
         SIMULATION PHASE (1 -- LONG RANGE, [illegible] 3 -- CLOSEUP)
      REAL R1(3), R2(3), R3(3), R4(3)
         INTERMEDIATE RESULTS WITH NO SIMPLE PHYSICAL INTERPRETATION
      REAL RELPOS(3)
         RELATIVE CAMERA COORDINATES IN CURRENT DOCKING AID FRAME (CENTERED
         AT CENTER LIGHT
      REAL R[illegible]
         MEASURED RANGE, CAMERA TO TARGET DOCKING AID CENTER LIGHT
      REAL RPYERR(3)
         MEASURED ROLL, PITCH, AND YAW ERRORS IN RADIANS
      REAL STATE(14)
         CHASE VEHICLE STATE VECTOR
      REAL T
         ELAPSED TIME
      REAL TAOUT(3,3)
         TRANSPOSE OF TARGET SPACECRAFT DIRECTION COSINE MATRIX
         (WITH RESPECT TO TRUTH AXES) FOR OUTPUT ONLY
      REAL TRUACV(3,3), TRUACT(3,3)
         CHASE VEHICLE ATTITUDE DIRECTION COSINE MATRIX (WITH RESPECT
         TO TRUTH COORDINATE SYSTEM), DERIVED FROM TRUE STATE
```


SIMULATION PROGRAM LISTING

PAGE 16

```
      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)

      COMMON/SIMUL/IDUMMY,OUT
      COMMON/OPTSYS/FOCLEN,DUMMY4(2)
      COMMON/HNGOFF/DUMMY3(6),HDC,HDT
      COMMON/VEHIC/DUMMY1(98),DOKRAD,DUMMY2(14)
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
C     (* TAKE MEASUREMENT *)
C        (* 410 - DO LIGHT-FLASHING SEQUENCE *)
         CALL FLASHM(U,V,T,STATE,F,VALID,E)
C        (* 490 - TAKE IMU MEASUREMENT *)
         CALL IMU(STATE,ACV,ACVT,ATRATE)
C        (* 909 - PROPAGATE TRUE STATE TO TIME OF THRUSTER COMMAND CHANGE *)
         CALL PROPTR( 5333333,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)+ 75*DU
           V(1)=V(1)+ 75*DV
           U(3)=U(3)+ 5*DU
           V(3)=V(3)+ 5*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,PHASE)
         GO TO 50
   10 CONTINUE
         CU=0
         CV=0
         NUV=0
         DO 20 I=1,4
           IF(E(I) NE 0) GO TO 20
              CU=CU+U(I)
              CV=CV+V(I)
              NUV=NUV+1
   20    CONTINUE
         IF(NUV EQ 0) GO TO 30
           RPYERR(1)=0
           RPYERR(2)=ATAN2(CV/NUV,FOCLEN)
           RPYERR(3)=ATAN2(CU/NUV,FOCLEN)
           GO TO 50
   30    CONTINUE
C          (* 420 - ESTIMATE ATTITUDE ERROR FROM STATE *)
              CALL ESTRPY(ESTATE,ACV,RPYERR)
   50 CONTINUE
C     (* WRITE PARAMETERS TO DISK FILE *)
        CALL TRGATT(TAOUT,T)
        WRITE(OUT) T,STATE,TAOUT,PHASE,ESTATE,P
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     (* 909 - PROPAGATE TRUE STATE TO TIME OF FIRST FLASH OF NEXT SEQUENCE *)
        CALL PROPTR(0 5333333,F,STATE,T)
C     (* 460 - PROP  STATE ESTIMATE TO TIME OF LAST FLASH OF NEXT SEQUENCE *)
        CALL PROPES(P,ESTATE,F,0 7,ACVT)
C     (* SEE IF CHASE VEHICLE HAS DOCKED YET *)
        CALL TRGATT(TAOUT,T)
        CALL DIRMAT(STATE(10),TRUAC,TRUACT)
        CALL MMLT(R1,TRUACT,HDC,3,3,1)
        CALL MMLT(R2,TAOUT,HDT,3,3,1)
        CALL MADD(R3,R1,STATE,3,1)
        CALL MSUB(R4,R3,R2,3,1)
        DOCKED=(SQRT(R4(1)**2+R4(2)**2+R4(3)**2) LE DOKRAD)  OR
     A     (T GT 280 )
C     (* WRITE PARAMETERS TO DISK FILE *)
        WRITE(OUT) T,STATE,TAOUT,PHASE,ESTATE,P
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 ')
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
[illegible] FLASHM [illegible] STATE [illegible] VALID E)
[illegible] THE LAMPS IN SEQUENCE AND RETURN CENTROID DATA [illegible]

[illegible] PARAMETER
[illegible]

INPUT
  [illegible] STATE [illegible]
OUTPUT
  [illegible] VALID STATE [illegible]

REAL [illegible]
  [illegible] OF PIXELS THAT ARE [illegible] (ABOVE THRESHOLD VOLTAGE)
REAL [illegible]
  [illegible] COLUMN NUMBERS FOR ALL [illegible] PIXELS
REAL [illegible]
  [illegible] ROW NUMBERS FOR ALL [illegible] PIXELS
REAL [illegible]
  HORIZONTAL COMPONENT OF IMAGE S CENTER OF BRIGHTNESS
REAL [illegible]
  VERTICAL COMPONENT OF IMAGE S CENTER OF BRIGHTNESS
INTEGER [illegible]
  PIXEL COUNTS
INTEGER [illegible] VERT 4
  [illegible] COLUMN NUMBERS FOR [illegible] PIXELS
INTEGER [illegible] HORZ 4
  [illegible] ROW NUMBERS FOR [illegible] PIXELS
INTEGER [illegible]
  ERROR CODES FOR EACH FLASH  A VALUE OF ZERO MEANS THAT
  NO ERROR OCCURED [illegible] ONE IF DATA VALID DID NOT OCCUR
  [illegible] IF LIGHT WAS NOT [illegible] OR 3  E(I) IS THREE WHEN
  [illegible] PIXEL COUNT WAS ZERO
INTEGER [illegible] POFSET
  DARK FRAME VALUE OF PIXEL  COUNTS  USED TO SUBTACT OUT BACKGROUND
  [illegible]
INTEGER [illegible] HOFSET
  DARK FRAME VALUE USED TO SUBTRACT BACKGROUND NOISE FROM
  HORIZONTAL ACCUMULATOR
INTEGER [illegible] VOFSET
  DARK FRAME VALUE USED TO SUBTRACT BACKGROUND NOISE FROM
  VERTICAL ACCUMULATOR
INTEGER [illegible] PMIN
  MINIMUM OF THE FOUR PIXEL COUNTS
INTEGER [illegible]
  DO-LOOP INDEX
LOGICAL ERROR
  TRUE IF ANY ERROR OCCURED - (SEE [illegible] ABOVE)
LOGICAL VALID
  TRUE IF FULL FLASH SEQUENCE WAS USEABLE
INTEGER NULL,RIGHT,LEFT,CENTER
  LAMP NUMBERS FOR NO FLASH  FLASH RIGHT LAMP, ETC
REAL F [illegible]
  FORCES FROM CHASE VEHICLE THRUSTERS AFTER SELECTION
REAL STATE(14)
  CHASE VEHICLE STATE VECTOR
REAL [illegible]
  ELAPSED TIME

DATA NULL RIGHT CENTER LEFT 0 1 2 3/
```



```
(* 908 - FLASH CENTER LIGHT AND SET PMIN *)
   CALL FLASH(CENTER,PIX(4),VERT(4),HORZ(4),E(1))
   PMIN=PIX(4)
(* 909 - PROPAGATE TRUE STATE 1/30 SEC *)
   CALL PROPTR( 03333333,F,STATE,T)
(* 908 - FLASH LEFT LIGHT AND UPDATE PMIN *)
   CALL FLASH(LEFT,PIX(1),VERT(1),HORZ(1),E(2))
   PMIN=AMIN1(PMIN,PIX(1))
(* 909 - PROPAGATE TRUE STATE 1/30 SEC *)
   CALL PROPTR( 03333333,F,STATE,T)
(* 908 - FLASH RIGHT LIGHT AND UPDATE PMIN *)
   CALL FLASH(RIGHT,PIX(3),VERT(3),HORZ(3),E(3))
   PMIN=AMIN1(PMIN,PIX(3))
(* 909 - PROPAGATE TRUE STATE 1/30 SEC *)
   CALL PROPTR( 03333333,F,STATE,T)
(* 908 - GET DATA WITH NO LIGHTS FLASHING (BACKGROUND CLUTTER) *)
   CALL FLASH(NULL,POFSET,VOFSET,HOFSET,E(5))
(* 909 - PROPAGATE TRUE STATE 1/30 SEC *)
   CALL PROPTR( 03333333 F,STATE,T)
(* 908 - FLASH CENTER LIGHT AGAIN AND UPDATE PMIN *)
   CALL FLASH(CENTER,PIX(2),VERT(2),HORZ(2),E(4))
   PMIN=AMIN1(PMIN,PIX(2))
(* 909 - PROPAGATE TRUE STATE 1/30 SEC *)
   CALL PROPTR( 03333333,F,STATE,T)
   ERROR=(E(5) NE 0)  AND  (E(5) NE 3)
   DO 10  I 1 4
(*  CONVERT PIXEL COUNTS TO FLOATING POINT *)
      PIX(I)=PIX(I)-POFSET
      PF(I)=AMAX1(1 0,FLOAT(PIX(I)))
(*  REMOVE BACKGROUND CLUTTER *)
      HORZ(I)=HORZ(I)-HOFSET
      HF(I)=FLOAT(HORZ(I))
      VERT(I)=VERT(I)-VOFSET
      VF(I)=FLOAT(VERT(I))
(*  COMPUTE CENTER OF BRIGHTNESS FROM HARDWARE-SUPPLIED DATA *)
      U(I)=(HF(I)/PF(I)-8 )*60 E-6
      V(I)=(VF(I)/PF(I)-2 )*35 65E-6
      ERROR = ERROR  OR  E(I) NE 0
10 CONTINUE
(* REMOVL CLUTTER FROM PMIN AND CONVERT TO FLOATING-POINT *)
   PMIN PMIN-POFSET
   VALID = (  NOT  ERROR )  AND  ( PMIN GT 0 )
   RETURN

   END
```

ORIGINAL PAGE IS OF POOR QUALITY

```
C
      SUBROUTINE ESTRPY(ESTATE,ACV,RPYERR)
C     (* 420 - ESTIMATE ATTITUDE ERROR FROM STATE ESTIMATE *
C
C     CALLS
C        ASIN,MSCL,MMLT
C     CALLED BY
C        DOCK
C
C     INPUT
C        ESTATE,ACV
C     OUTPUT
C        RPYERR
C
      REAL ACV(3,3)
C        CHASE VEHICLE DIRECTION COSINE MATRIX (WITH RESPECT
C        TO PRIMARY REFERENCE FRAME)
      REAL EMAG
C        ESTIMATED RANGE (ABSOLUTE VALUE OF ESTATE)
      REAL ESTATE(6)
C        ESTIMATED STATE
      REAL R1(3),R(3)
C        INTERMEDIATE RESULTS
      REAL RPYERR(3)
C        ERRORS IN ROLL, PITCH AND YAW, RESPECTIVELY (RADIANS)
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      RPYERR(1)=0
C     (* COMPUTE R=ACV*(-ESTATE)/ABS(ESTATE), IN CV COORD  SYS *)
         EMAG=SQRT(ESTATE(1)**2+ESTATE(2)**2+ESTATE(3)**2)
         CALL MSCL(R1,ESTATE,3,1,-1 /EMAG)
         CALL MMLT(R,ACV,R1,3,3,1)
      RPYERR(2)=ATAN2(R(3),R(1))
      RPYERR(3)=ATAN2(-R(2),R(1))
      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, AND 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/FOCLEN,A,B
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
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)
      IF(AP NE 0  OR BP NE 0 ) GO TO 40
         RHO=800
         RELPOS(1)=-800
         RELPOS(2)=0
         RELPOS(3)=0
         RETURN
   40 IF(AP NE 0 ) GO TO 50
         RHO=B*FOCLEN/BP
         YP=RHO
         XP=0
         ZP=0
         GO TO 70
   50 IF(H NE 0 ) GO TO 60
         ZP=0
         D=BP*A/(B*AP)
         XP=-1 /SQR1(1 +D**2)
         YP=D*XP
         RHO=A*XP*FOCLEN/AP
         IF((U(1)-U(2))**2+(V(1)-V(2))**2 LT
    A       (U(3)-U(2))**2+(V(3)-V(2))**2) YP=-YP
         GO TO 70
   60 CONTINUE
         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((U(3)-U(1))*(V(2)-VC)  GT
    A       (V(3)-V(1))*(U(2)-UC) )   ZP=-ZP
         IF((U(1)-U(2))**2+(V(1)-V(2))**2 LT
    A       (U(3)-U(2))**2+(V(3)-V(2))**2) YP=-YP
         RHO=A*SQR1(XP**2+ZP**2)*FOCLEN/AP
   70 CONTINUE
      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,D
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/OPTICS/DUMMY1(2),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 PRIMARY 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 MATRIX 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/OPTS/S/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,SQRT,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 TO 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 RDTLOS
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      (* 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)
          PPDMAG=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(PPDMAG 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(PPDMAG, DOTPRD)
C             (* COMPUTE QUATERNION COMPONENTS *)
                CALL MSCL(QT1,V1,3,1,SIN(PHI*.5)/PPDMAG)
                QT1(4)=COS(PHI*.5)
   30   CONTINUE
          IF(V(3) EQ V(1) AND V(3) EQ V(1)) RETURN
C       (* CORRECT FOR ROTATION ABOUT LINE OF SIGHT *)
C         (* 791 - CONVERT QT1 TO DIRECTION COSINE MATRIX *)
              CALL DIRMAT(QT1,AT,TRNAT)
C          (* FIND ROTATION OF LINE 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(4)*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

        END
```



```
C
      SUBROUTINE INCORP(CVPOS,P,ESTATE,PHASE)
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, PHASE
C     OUTPUT
C        P, ESTATE
C
      REAL CVPOS(3)
C        MEASURED CHASE VEHICLE POSITON IN PRIMARY REFERENCE FRAME
      INTEGER ERR
C        ERROR CODE (=0 IF OKAY)
      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
      INTEGER I,J
C        DO LOOP INDICES
      REAL P(6,6),PA(3,3),PAI(3,3)
C        COVARIANCE OF STATE ESTIMATE, ITS UPPER LEFT 3X3 SUBMATRIX, AND
C        THE INVERSE OF THAT SUBMATRIX
      INTEGER PHASE
C        SIMULATION PHASE (1,2 OR 3)
      REAL R(3,3)
C        MEASUREMENT COVARIANCE
      REAL T1(3),E(3),C,SCRACH(4,6)
C        TEMPORARY VARIABLES
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* CHECK FOR RIDICULOUS VALUE *)
         DO 10 I=1,3
            DO 10 J=1,3
               PA(I,J)=P(I,J)
   10    CONTINUE
         CALL MINV(PAI,PA,3,SCRACH,4,6,ERR)
         IF(ERR NE 0) GO TO 30
         CALL MSUB(E,CVPOS,ESTATE,3,1)
         CALL MMLT(T1,PAI,E,3,3,1)
         CALL MMLT(C,E,T1,1,3,1)
         IF(C LE 16 ) GO TO 20
           RETURN
   20    CONTINUE
C     (* 451 - COMPUTE JACOBIAN G *)
         CALL COMPG(ESTATE,G)
C     (* 452 - ESTIMATE MEASUREMENT COVARIANCE *)
         CALL ESTCOV(ESTATE,R,PHASE)
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
   30 CONTINUE
      WRITE(1,901)
      STOP
C
  901 FORMAT('P IS SINGULAR IN ''INCORP''')
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,PHASE)
C     (* 452 - ESTIMATE MEASUREMENT COVARIANCE *)
C
C     CALLS
C        <NONE>
C     CALLED BY
C        INCORP
C
C     INPUT
C        ESTATE,PHASE
C     OUTPUT
C        R
C
      REAL ESTATE(6)
C        ESTIMATE OF STATE VECTOR
      INTEGER I,J
C        DO LOOP INDICES
      INTEGER PHASE
C        SIMULATION PHASE (1,2 OR 3)
      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=40 E-9*(ESTATE(1)**2+ESTATE(2)**2+ESTATE(3)**2)**2+ 01
      IF(PHASE NE 1) VARNCE=6 *VARNCE
      R(1,1)=VARNCE
      R(2,2)=VARNCE
      R(3,3)=VARNCE
      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[illegible]
C        PARTIAL OF MEASUREMENT WITH RESPECT TO STATE
      REAL GT[illegible]
C        TRANSPOSE OF G
      INTEGER [illegible]
C        DO LOOP INDICES
      REAL KGAIN[illegible]
C        KALMAN GAIN MATRIX
      REAL P[illegible]
C        [illegible] ESTIMATE COVARIANCE
      REAL R[illegible]
C        MEASUREMENT COVARIANCE
      REAL SCRACH[illegible]
C        SCRATCH AREA FOR ROUTINE MINV
      REAL TEMP1([illegible]),TEMP2(3,[illegible]),TEMP3(3,3)
C        INTERMEDIATE RESULTS WITH NO PROBLEM-RELATED SIGNIFICANCE
      INTEGER TERMNL
C        FORTRAN UNIT NUMBER FOR TERMINAL
C
      COMMON /SMLL/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(TEMP2,R,TEMP2,3,3)
      CALL MINV(TEMP3,TEMP2,3,SCRACH,3,6,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 FAILED 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)
```

```
C    * [illegible] FORCE [illegible] *
      CALL [illegible] AR[illegible] 3, 4, 1)
      CALL [illegible] ACV, FDOD, 3, 3 [illegible]
C    * ESTIMATE ACCELERATION *
      CALL [illegible] (FORCE, 3, 1, 1, AVCHAS)
C    * ESTIMATE ACCELERATION VARIANCE *
      AMAG[illegible]=ACCEL(1)**2+ACCEL(2)**2+ACCEL(3)**
      AVAR [illegible] 05*AMAG[illegible]
C    * FIND POSITION CHANGE *
      DO [illegible] I=1,3
         [illegible]STATE(I) = ESTATE(I+3) + .5*ACCEL(I) *STEP)*STEP
   [illegible] CONTINUE
C    * FIND VELOCITY CHANGE *
      DO 2[illegible] I=4,6
         [illegible]STATE(I) = [illegible]
   [illegible] CONTINUE
C    * FIND COVARIANCE CHANGE *
      DO [illegible]
         [illegible]
   [illegible] CONTINUE
      [illegible] AVAR
      [illegible] AVAR
      [illegible] AVAR
      DO [illegible]
         [illegible]
   [illegible] CONTINUE
      [illegible]
         [illegible]
            [illegible] DPCI [illegible]
   [illegible] CONTINUE
C    * COMPUTE NEW STATE AND COVARIANCE *
      DO [illegible]
         [illegible] STATE(I)=ESTATE(I)+DSTATE(I)
         [illegible]
   [illegible] CONTINUE
      RETURN
      END
```



```
      SUBROUTINE SETGOL(ESTATE, P, T, ACV, AT, GXL, GXH, GY_, GYH, GZL, GZH,
     A    DEDLIN, VALID)
C    * 470 - SET GOAL *)
C
C    CALLS
C       SQRT, MAX1, MTRN, MMLT, MSUB
C    CALLED BY
C       DOCK
C
C    INPUT
C       ESTATE, P, DOKRAD, T, ACV, AT
C    OUTPUT
C       GXL, GXH, GY_, 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/LINGOFF/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 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

```
      SUBROUTINE CNTLAW(ROTCMD, XLTCMD, ESTATE, ACV, TLIM, GXL, GXH, GYL, GYH,
     A    GZL, GZH, ATRATE, RPYERR)
      (* 481 - CONTROL LAW *)

      CALLS
         MMLT, ACCEL
      CALLED BY
         THRUST

      INPUT
         ESTATE, ACV, TLIM, GXL,     , GZH, ATRATE, RPYERR
      OUTPUT
         ROTCMD, XLTCMD

      REAL ACV(3,3)
         MEASURED CHASE VEHICLE ATTITUDE IN PRIMARY REFERENCE FRAME
      REAL ATRATE(3)
         MEASURED CHASE VEHICLE ATTITUDE RATES
      REAL ESTATE(6
         STATE ESTIMATE
      REAL GXL, GXH, GYL, GYH, GZL, GZH
         LOWER AND UPPER LIMITS OF GOAL 'BOX'
      REAL ROTCMD(3), XLTCMD(3)
         ROTATIONAL AND TRANSLATIONAL COMMANDS FROM CONTROL LAW
      REAL RPYERR(3
         MEASURED ATTITUDE ERROR IN ROLL, PITCH  AND YAW
      REAL TLIM
         TIME LIMIT
      REAL VBOD(3)
         VELOCITY IN CHASE VEHICLE BODY COORDINATE SYSTEM
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      (* CONVERT VELOCITY TO BODY COORD  SYS  *)
         CALL MMLT(VBOD ACV, ESTATE(4), 3, 3, 1)
      (* 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

      END
```



```
      FUNCTION ACCEL(XDOT, XMIN, XMAX, TLIM, DT, AMIN, AMAX)
      (* 481 1 - ESTIMATE ACCELERATION FOR ONE AXIS *)

      CALLS
         (NONE)
      CALLED BY
         CNTLAW

      INPUT
         X, XDOT, XMIN, XMAX, TLIM, DT
      OUTPUT
         (FUNCTION VALUE ONLY)

      REAL AMIN, AMAX
           MIN AND MAX RELATIVE THRUST
      REAL DT
           TIME STEP BETWEEN DECISIONS
      REAL K1, K2, K3, K4
           CONSTANTS BASED ON MIN, MAX POSSIBLE ACCELERATION
      REAL TLIM
           TIME LIMIT - SECONDS UNTIL DEADLINE
      REAL XDOT
           VELOCITY IN THIS AXIS
      REAL XMIN, XMAX
           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(XLOT 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     (* 4.2 - 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

```
      SUBROUTINE TABLE(INDEX1,E)
C     FIND COMBINATION OF TABLE ENTRIES THAT SATISFY THRUSTER
C     SELECTION REQUIREMENTS
[illegible]
C     COMMAND ENTRIES FROM THRUSTER SELECT LOGIC
C     ENTER TABLE [illegible]
C     SELECTION BASED ON INDEX
      GO TO (10,20,30,40,50,60,70,80,90,100,110,120,130,140,150,160,170,
     1 180,190,200,210,220,230,240,250,260),INDEX1
[illegible]
```


ORIGINAL PAGE IS
OF POOR QUALITY

```
 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
      END
```

```
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(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
  260 E(3)=1 0
      E(9)=1 0
  300 RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
      SUBROUTINE TABLES(INDEX...)
C     ... FIND COMBINATION OF TABLE ENTRIES THAT SATISFY ... 
C     SEL...   REQUIREMENTS
C
C     CALLED ...
C        SE...
C
C     INPUT
C
C
C     ... ENTRIES FROM THRUSTER SELECT LOGIC
C     ... INDEX
C
C     ... BASED ON INDEX ...
      ... 20 30 40 50 60 70 80 90 100 110 120 130 140 150 160 170
      ... 190 200 210 220 230 240 250 ...
```



```
      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 200
  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

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     (* 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
```

```
      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     (* 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     (* 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,DSTATE(7))
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,G
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)
C     (* 902 2 -  COMPUTE TIME DERIVATIVE OF SPACECRAFT MASS *)
C
C     CALLS
C        MMLT
C     CALLED BY
C        STPRIN
C
C     INPUT
C        AKI,F
C     OUTPUT
C        DMDT
C
      REAL AKI(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/AKI,DUMMY(97)
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* COMPUTE DMDT = AKI*F *)
         CALL MMLT(DMDT,AKI,F,1,14,1)
C     (* CHANGE SIGN - MASS LOSS IS NEGATIVE *)
         DMDT=-DMDT
      RETURN
C
      END
```



```
C
      SUBROUTINE LINACL(T,M,NTFORC,CURPOS,ACCEL)
C     (* 902 3 -  COMPUTE LINEAR ACCELERATIONS IN X, Y, AND Z DIRECTIONS *)
C
C     CALLS
C        SIN,COS
C     CALLED BY
C        STPRIN
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 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
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
      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,DLDT)
C     (* 902 5 -  COMPUTE TIME DERIVATIVE OF ANGULAR MOMENTUM *)
C
C     CALLS
C        NONE
C     CALLED BY
C        STPRIM
C
C     INPUT
C        N
C     OUTPUT
C        DLDT
C
      REAL DLDT(3)
C        TIME DERIVATIVE OF ANGULAR MOMENTUM VECTOR
      INTEGER I
C        DO LOOP INDEX
      REAL N(3)
C        TORQUE ON C V  ABOUT ITS CENTER OF MASS IN 'TRUTH' COORDINATE SYSTEM
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      DO 10 I=1,3
         DLDT(I)=N(I)
   10 CONTINUE
      RETURN
C
      END
```

ORIGINAL PAGE IS OF POOR QUALITY

```
C
      SUBROUTINE MAKEO[illegible] ([illegible] OMEGA [illegible])
C     [illegible] FORM ROTATION MATRIX OMEGA FROM ANGULAR VELOCITY VECTOR *)
C
C     CALLS
C        [illegible]
C     CALLED [illegible]
C        [illegible]
C
C     INPUT
C        [illegible]
C     OUTPUT
C        OM[illegible]
C
      REAL [illegible]
C        DIRECTION COSINE MATRIX GIVING [illegible] ATTITUDE
      REAL [illegible]
C        [illegible] VELOCITY IN [illegible] COORDINATE SYSTEM
      INTEGER [illegible]
C        [illegible]
      REAL [illegible](4,4)
C        ROTATION MATRIX OMEGA TO BE APPLIED TO QUATERNION
C        TO FORM TIME DERIVATIVE OF QUATERNION
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      DO 1[illegible] [illegible] 4
         [illegible]
   10    [illegible]
      OMEGA([illegible])=BODVEL(3)
      OMEGA(1,[illegible])=-BODVEL(2)
      OMEGA([illegible])=-BODVEL(1)
      OMEGA(2,[illegible])=-BODVEL(3)
      OMEGA(2,3)=BODVEL(1)
      OMEGA(2,4)=BODVEL(2)
      OMEGA([illegible])=BODVEL(2)
      OMEGA(3,[illegible])=-BODVEL(1)
      OMEGA(3,4)=BODVEL(3)
      OMEGA([illegible])=-BODVEL([illegible])
      OMEGA(4,[illegible])=-BODVEL(2)
      OMEGA(4,[illegible])=-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        STPRIN
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

```
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
```



```
C
      SUBROUTINE ANGVEL(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  TETHR  AXE
      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 ANGVEL')
C
      END
```

```
      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 ITUMB
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 TRNAT0(3,3)
C        INITIAL VALUE OF TRNAT
      REAL TUMRAT
C        TUMBLE RATE (IN RAD/SEC)
C
      COMMON/ATINFO/TRNAT0,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,TRNAT0,CHGATT,3,3,3)
      RETURN
```



```
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE POINT(T,STATE)
C     (* 906 - CONTROL SIMULATOR SO THAT CAMERA SEES WHAT REAL CAMERA WOULD *)
C
C     CALLS
C        DIRMAT, SERVO, SIMROT, SIMXLT, TRGATT, VREF
C     CALLED BY
C        PROPTR, INISIM
C
C     INPUT
C        T, STATE
C     OUTPUT
C        SIMYPR
C
      REAL ACV(3,3), TRNACV(3,3)
C        CHASE VEHICLE ATTITUDE DIRECTION COSINE MAT. X (ATTITUDE WITH
C        RESPECT TO 'TRUTH' FRAME), AND TRANSPOSE OF THIS MATRIX
      REAL SIMXYZ(3), SIMYPR(3)
C        SIMULATOR POSITION (Z,Y,Z IN METERS) AND ATTITUDE (YAW,ROLL,PITCH
C        IN RADIANS) COMMANDS
      REAL STATE(14)
C        'TRUE' STATE OF CHASE VEHICLE IN 'TRUTH' FRAME
      REAL T
C        TIME SINCE START OF RENDEZVOUS, SECONDS
      REAL TRNAT(3,3)
C        TRANSPOSE OF DIRECTION COSINE MATRIX OF TARGET (ATTITUDE WITH RESPECT
C        TO 'TRUTH' FRAME)
C
      COMMON/SIMCAL/DUMMY(12), SCALE, DUMMY2(16), SIMYPR, DUMMY1(8)
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* 905 - FIND TARGET ATTITUDE AS FUNCTION OF TIME *)
         CALL TRGATT(TRNAT,T)
C     (* 901 - FIND CHASE VEHICLE ATTITUDE FROM STATE VECTOR *)
         CALL DIRMAT(STATE(10),ACV,TRNACV)
C     (* 906 1 - COMPUTE SIMULATOR TRANSLATIONAL COMMANDS *)
         CALL SIMXLT(TRNAT,STATE,TRNACV,SIMXYZ)
C     (* 906 2 - COMPUTE SIMULATOR ROTATIONAL COMMANDS *)
         CALL SIMROT(SIMYPR,ACV,TRNAT)
C     (* 906 3 - COMMAND SERVOS ON SIMULATOR *)
         CALL SERVO(SIMXYZ,SIMYPR)
C     (* 907 - SET CAMERA THRESHOLD VOLTAGE *)
         CALL VREF(2 875-SCALE*SQRT(STATE(1)**2+STATE(2)**2+
     A      STATE(3)**2)*0 25)
      RETURN
C
      END
```



```
C
      SUBROUTINE SIMXLT(TRNAT,STATE,TRNACV,SIMXYZ)
C     (* 906 1 - COMPUTE SIMULATOR TRANSLATIONAL COMMANDS *)
C
C     CALLS
C        MADD, MMLT, MSCL, MSUB, MTRN
C     CALLED BY
C        POINT
C
C     INPUT
C        STATE, TRNACV, TRNAT
C     OUTPUT
C        SIMXYZ
C
      REAL AT(3,3), TRNAT(3,3)
C        TARGET ATTITUDE DIRECTION COSINE MATRIX AND ITS TRANSPOSE
      REAL AST(3,3)
C        DIRECTION COSINE MATRIX GIVING ATTITUDE OF SIMULATOR WITH
C        RESPECT TO THE TARGET MODEL USED FOR THIS SIMULATION PHASE
      REAL HC(3),HT(3)
C        HANGOFFS OF CAMERA AND DOCKING AID WITH RESPECT TO THEIR SPACECRAFT
C        CENTERS OF MASS
      REAL LLENS(3)
C        OFFSET OF EFFECTIVE LENS CENTER FROM INTERSECTION OF GIMBAL AXES
C        VECTOR IS IN COORDINATE SYSTEM ALIGNED WITH LENS AND CENTERED AT
C        THE AXIS INTERSECTION
      REAL LTS(3)
C        POSITION OF DOCKING AID ON TARGET MODEL IN SIMULATOR COORDINATE SYSTEM
      REAL SCALE
C        SCALE FOR THIS SIMULATION PHASE (MODEL SIZE/TRUE SIZE)
      REAL SIMXYZ(3)
C        COMMAND TO SIMULATOR, IN METERS, IN SIMULATOR COORD SYS
      REAL STATE(3)
C        FIRST 3 ELEMENTS OF STATE VECTOR (=POSITION IN 'TRUTH' COORD SYS )
      REAL TRNACV(3,3)
C        TRANSPOSE OF DIRECTION COSINE MATRIX GIVING CHASE VEHICLE ATTITUDE
      REAL VTEMP(3), VTEMP1(3), VTEMP2(3), VTEMP3(3), VTEMP4(3), VTEMP5(3),
     A   VTEMP6(3), VTEMP7(3), VTEMP8(3)
C        TEMPORARY STORAGE FOR INTERMEDIATE RESULTS
C
      COMMON/SIMCAL/AST, LTS, SCALE, LLENS, DUMMY1(24)
      COMMON/HNGOFF/HC, HT, DUMMY(6)
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     (* COMPUTE SIMXYZ=
C        LTS+AST*(AT*(SCALE*STATE+TRNACV*(SCALE*HC-LLENS))-SCALE*HT)   *)
         CALL MSCL(VTEMP,HC,3,1,SCALE)
         CALL MSUB(VTEMP1,VTEMP,LLENS,3,1)
         CALL MMLT(VTEMP2,TRNACV,VTEMP1,3,3,1)
         CALL MSCL(VTEMP3,STATE,3,1,SCALE)
         CALL MADD(VTEMP4,VTEMP3,VTEMP2,3,1)
         CALL MTRN(AT,TRNAT,3)
         CALL MMLT(VTEMP5,AT,VTEMP4,3,3,1)
         CALL MSCL(VTEMP6,HT,3,1,SCALE)
         CALL MSUB(VTEMP7,VTEMP5,VTEMP6,3,1)
         CALL MMLT(VTEMP8,AST,VTEMP7,3,3,1)
         CALL MADD(SIMXYZ,LTS,VTEMP8,3,1)
      RETURN
C
      END
```

```
      SUBROUTINE SIMROT (DUMMY, ACV, R JAT)
C     (* YOU  - COMPUTE SIMULATOR ROTATIONAL COMMANDS *)
C
C     CALLS
C        MMLT ATAN2
C     CALLED BY
C        POINT
C
C     INPUT
C        ACV, TRNAT
C     OUTPUT
C        SIMYPR
C
C     REAL A(3,3)
C        DIRECTION COSINE MATRIX GIVING NET ATTITUDE CHANGE CAUSED BY
C        NON ZERO GIMBAL ANGLES
C     REAL ACV(3,3)
C        CHASE VEHICLE ATTITUDE DIRECTION COSINE MATRIX   ATTITUDE IS
C        WITH RESPECT TO (TRUTH) COORDINATE SYSTEM
C     REAL ANGNET
C        NET ROTATION ANGLE - SUM (OR DIFFERENCE) OF YAW AND ROLL   USED
C        AT SINGULARITY WHERE ROLL AND YAW AXES ALIGN
C     REAL ATSP(3,3)
C        DIRECTION COSINE MATRIX GIVING ATTITUDE OF TARGET MODEL WITH
C        RESPECT TO THE CAMERA'S ZERO-GIMBAL-ANGLE ATTITUDE   INCLUDES
C        EFFECTS OF MISALIGNMENT BETWEEN CAMERA AND SIMULATOR WHEN GIMBAL
C        SET IS ZEROED
C     REAL D
C        INTERMEDIATE RESULT   ANGLE CHANGE, RADIANS
C     REAL R
C        NEW ROLL GIMBAL COMMAND, RADIANS
C     REAL ROLMIN, ROLMAX
C        LIMITS (IN RADIANS) OF ROLL GIMBAL FREEDOM
C     REAL SIMYPR(3)
C        YAW, PITCH, ROLL COMMANDS TO GIMBAL SET   RADIANS
C     REAL TEMP(3,3)
C        INTERMEDIATE RESULT
C     REAL TRNAT(3,3)
C        TRANSPOSE OF DIRECTION COSINE MATRIX THAT GIVES TARGET ATTITUDE
C        IN (TRUTH) COORDINATE SYSTEM
C     REAL U, V
C        DUMMY ARGUMENTS IN STATEMENT FUNCTION DEFINITION
C     REAL Y
C        NEW YAW GIMBAL COMMAND, RADIANS
C     REAL YAWMIN, YAWMAX
C        LIMITS (IN RADIANS) OF YAW GIMBAL FREEDOM
C     LOGICAL ZERO
C        STATEMENT FUNCTION TO CHECK FOR VALUES CLOSE TO ZERO
C
      COMMON /SMCAL/DUMMY(15), ROLMIN, ROLMAX, YAWMIN, YAWMAX, ATSP,
     A   DUMMY(11)
C
C     STATEMENT FUNCTION
      ZERO(U,V)=ABS(U) LT 1 E-5  AND  ABS(V) LT 1 E-5
```



```
C
C     (* COMPUTE A=ACV*TRNAT*ATSP *)
         CALL MMLT(TEMP, ACV, TRNAT, 3, 3, 3)
         CALL MMLT(A, TEMP, ATSP, 3, 3, 3)
C     (* CHECK FOR SINGULARITY *)
      IF(ZERO(A(1,1), A(1,2)) OR ZERO(A(2,3), A(3,3))) GO TO 20
C        (* NO SINGULARITY, FIND GIMBAL ANGLES *)
            SIMYPR(1)=ATAN2(A(1,2), A(1,1))
            SIMYPR(3)=ATAN2(A(2,3), A(3,3))
            IF(ABS(A(1,1)) LE ABS(A(1,2))) GO TO 10
               SIMYPR(2)=ATAN2(-A(1,3), A(1,1)/COS(SIMYPR(1)))
               RETURN
   10       CONTINUE
               SIMYPR(2)=ATAN2(-A(1,3), A(1,2)/SIN(SIMYPR(1)))
               RETURN
   20 CONTINUE
C        (* SINGULARITY, PICK SOLUTION WITH LEAST CHANGE IN ANGLES *)
            ANGNET=ATAN2(-A(2,1), A(2,2))
            IF(A(1,3) GT 0 ) GO TO 50
               SIMYPR(2)=0 7853982
               D=ANGNET-SIMYPR(1)+SIMYPR(3)
               Y=SIMYPR(1)+ 5*D
               R=SIMYPR(3)- 5*D
               IF(R GT ROLMAX OR R LT ROLMIN) GO TO 40
                  IF(Y GT YAWMAX OR Y LT YAWMIN) GO TO 30
                     SIMYPR(1)=Y
                     SIMYPR(3)=R
                     RETURN
   30             CONTINUE
                     SIMYPR(3)=SIMYPR(3)-D
                     RETURN
   40          CONTINUE
                  SIMYPR(1)=SIMYPR(1)+D
                  RETURN
   50       CONTINUE
               SIMYPR(2)=-0 7853982
               D=ANGNET-SIMYPR(1)-SIMYPR(3)
               Y=SIMYPR(1)+ 5*
               R=SIMYPR(3)+ 5*
               IF(R GT ROLMAX OR R LT ROLMIN) GO TO 70
                  IF(Y GT YAWMAX OR Y LT YAWMIN) GO TO 60
                     SIMYPR(1)=Y
                     SIMYPR(3)=R
                     RETURN
   60             CONTINUE
                     SIMYPR(3)=SIMYPR(3)+D
                     RETURN
   70          CONTINUE
                  SIMYPR(1)=SIMYPR(1)+D
                  RETURN
C
      END
```

ORIGINAL PAGE IS OF POOR QUALITY

```
C
      SUBROUTINE SERVO(SIMXYZ,SIMYPR)
C     (* 906 3 - SEND ANALOG CONTROL VOLTAGE TO SERVOS ON SIMULATOR *)
C
C     CALLS
C        CLOS$A,EXIT
C     CALLED BY
C        POINT
C
C     INPUT
C        SIMXYZ,SIMYPR
C     OUTPUT
C        <NONE>
C
      REAL SIMXYZ(3),SIMYPR(3)
C        SIMULATOR POSITION (Z,Y,Z IN METERS) AND ATTITUDE (YAW,ROLL,PITCH
C        IN RADIANS) COMMANDS
      REAL OLDXYZ(3),OLDYPR(3)
C        PREVIOUS VALUES OF SIMXYZ,SIMYPR FOR TIME DELAY CALCULATION
      LOGICAL GOOD
C        TRUE IF SIMULATOR CAN GET TO POSITION COMMANDED
      INTEGER I
C        DO LOOP INDEX
      LOGICAL NEWRUN
C        FLAGS FACT THAT THIS IS A NEW RUN AND THERE SHOULD BE NO DELAY
C        FOR SERVO SETTLING TIME
      INTEGER OUT,TERMNL
C        FORTRAN LOGICAL UNIT OF OUTPUT FILE AND TERMINAL
      INTEGER PORT(6)
C        D/A PORT NUMBERS FOR X,Y,Z,YAW,PITCH AND ROLL, RESPECTIVELY
      REAL ROLMIN,ROLMAX,YAWMIN,YAWMAX,PCHMIN,PCHMAX,XMIN,XMAX,YMIN,
     A    YMAX,ZMIN,ZMAX
      REAL SPDX,SPDY,SPDZ,SPDYAW,SPDPIT,SPDROL,DELMIN
C        SIMULATOR AXIS SLEWRATES (M/SEC OR RAD/SEC) & MIN  RESPONSE DELAY
      DOUBLE PRECISION TSETL
C        TIME BY WHICH SIMULATOR SERVOS SHOULD BE SETTLED
      REAL VOLTS(6)
C        VOLTAGE TO SEND TO, RESPECTIVELY, X,Y,Z,YAW,PITCH,ROLL
C        LIMITS OF SERVO TRAVEL FOR GIMBALS AND TRANSLATIONAL SERVOS
      REAL YPR(3)
C        SIMYPR EXPRESSED IN DEGREES INSTEAD OF RADIANS
C
      COMMON/SIMUL/TERMNL,OUT
      COMMON/SRVOTM/OLDXYZ,OLDYPR,TSETL,NEWRUN
      COMMON/SIMCAL/DUMMY(16),ROLMIN,ROLMAX,YAWMIN,YAWMAX,DUMMY1(12),
     A    PCHMIN,PCHMAX,XMIN,XMAX,YMIN,YMAX,ZMIN,ZMAX
C
      DATA PORT/1,2,3,6,4,7/
      DATA SPDX,SPDY,SPDZ,SPDYAW,SPDPIT,SPDROL,DELMIN/
     A     091, 087, 079, 003, 003, 003, 4/
```



```
C
      DO 10 I 1,3
         YPR(I)=SIMYPR(I)*57 29578
   10 CONTINUE
      GOOD= TRUE
      IF(SIMYPR(1) GE YAWMIN  AND  SIMYPR(1) LE YAWMAX) GO TO 20
         WRITE(TERMNL,902)
         GOOD= FALSE
   20 IF(SIMYPR(2) GE PCHMIN  AND  SIMYPR(2) LE PCHMAX) GO TO 30
         WRITE(TERMNL,903)
         GOOD= FALSE
   30 IF(SIMYPR(3) GE ROLMIN  AND  SIMYPR(3) LE ROLMAX) GO TO 40
         WRITE(TERMNL,904)
         GOOD= FALSE
   40 IF(SIMXYZ(1) GE XMIN    AND  SIMXYZ(1) LE XMAX  ) GO TO 50
         WRITE(TERMNL,905)
         GOOD= FALSE
   50 IF(SIMXYZ(2) GE YMIN    AND  SIMXYZ(2) LE YMAX  ) GO TO 60
         WRITE(TERMNL,906)
         GOOD= FALSE
   60 IF(SIMXYZ(3) GE ZMIN    AND  SIMXYZ(3) LE ZMAX  ) GO TO 70
         WRITE(TERMNL,907)
         GOOD= FALSE
   70 CONTINUE
      VOLTS(1)=( 005840617*SIMXYZ(1)+4 050095)*SIMXYZ(1)-10 64318
      VOLTS(2)=( 1528103*SIMXYZ(2)-3 776146)*SIMXYZ(2)
      VOLTS(3)=(- 04265157*SIMXYZ(3)-6 279022)*SIMXYZ(3)+5 545861
      VOLTS(4)=( 06429689*SIMYPR(1)-6 2652)*SIMYPR(1)
      VOLTS(5)=(- 04356859*SIMYPR(2)-6 29776)*SIMYPR(2)
      VOLTS(6)=( 001013212*SIMYPR(3)+2 638789)*SIMYPR(3)
C     (* 910 - GET TIME SINCE MIDNIGHT (SECONDS) *)
         CALL ESECS(TSETL)
C     (* FIGURE SETTLING TIME *)
         TSETL=TSETL + AMAX1(
     A         ABS(SIMXYZ(1)-OLDXYZ(1))/SPDX,
     B         ABS(SIMXYZ(2)-OLDXYZ(2))/SPDY,
     C         ABS(SIMXYZ(3)-OLDXYZ(3))/SPDZ,
     D         ABS(SIMYPR(1)-OLDYPR(1))/SPDYAW,
     E         ABS(SIMYPR(2)-OLDYPR(2))/SPDPIT,
     F         ABS(SIMYPR(3)-OLDYPR(3))/SPDROL) + DELMIN
       OLDXYZ(1)=SIMXYZ(1)
       OLDXYZ(2)=SIMXYZ(2)
       OLDXYZ(3)=SIMXYZ(3)
       OLDYPR(1)=SIMYPR(1)
       OLDYPR(2)=SIMYPR(2)
       OLDYPR(3)=SIMYPR(3)
       IF( NOT NEWRUN) GO TO 80
          NEWRUN= FALSE
          TSETL=0 D0
   80  CONTINUE
       IF(GOOD) GO TO 90
          WRITE(TERMNL,901) SIMXYZ,YPR
          WRITE(TERMNL,909) VOLTS
          WRITE(TERMNL,908)
          CALL CLOS$A(OUT-A)
          CALL EXIT
   90  CONTINUE
       DO 100 I = 1,6
          CALL DAOUT(INT(204 8*VOLTS(I)),PORT(I))
  100  CONTINUE
       RETURN
C
  901  FORMAT('SIMULATOR POSIT='/3G15 6/'SIMULATOR Y-P-R SEQ='/3G15 6)
  902  FORMAT('SIMULATOR YAW COMMAND EXCEEDS SERVO RANGE ')
  903  FORMAT('SIMULATOR PITCH COMMAND EXCEEDS SERVO RANGE ')
  904  FORMAT('SIMULATOR ROLL COMMAND EXCEEDS SERVO RANGE ')
  905  FORMAT('SIMULATOR X-AXIS COMMAND EXCEEDS SERVO RANGE')
  906  FORMAT('SIMULATOR Y-AXIS COMMAND EXCEEDS SERVO RANGE ')
  907  FORMAT('SIMULATOR Z-AXIS COMMAND EXCEEDS SERVO RANGE')
  908  FORMAT('*** SIMULATION ABORTED DUE TO OUT-OF-RANGE SIMULATOR ',
     A    'COMMAND ')
  909  FORMAT('VOLTAGES TO SIMULATOR=',6(F7 3))
C
       END
```

```
      SUBROUTINE VREF(VOUT)
C(* 907   SEND COMPARITOR REFERENCE VOLTAGE TO VIDEO PROCESSING ELEC-
C    TRONICS *)
C
C   CALLS
C     DAOUT
C   CALLED BY
C     POINT
C
C   INPUT
C     VOUTI
C   OUTPUT
C     NONE
C
      REAL VOUTI
C      VOLTAGE TO SEND TO ELECTRONICS FROM D/A CONVERTER
C- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C   * SEND INTEGER CORRESPONDING TO VOLTAGE TO D/A PORT 8 OF COMPUTER *)
      CALL DAOUT(INT(VOUTI*204.8),8)
      RETURN
C
      END
```


ORIGINAL PAGE IS OF POOR QUALITY

```
C
      SUBROUTINE FLASH(LIGHT,IIP,IIV,IIH,FERROR)
C(* 908   FLASH ONE LIGHT & RETURN RAW CENTROID DATA *)
C
C   CALLS
C     DINAA,DINBA,DINCA,TNOUA,GOTIT
C   CALLED BY
C     FLASHM,ELECIN
C
C   INPUT
C     LIGHT
C   OUTPUT
C     FERROR IIP,IIV,IIH
C
      INTEGER IGOTIT
C     FLAG TO INDICATE WHETHER ELECTROICS RECEIVED COMMAND
      INTEGER LIGHT
C     LAMP NUMBER OF LAMP TO FLASH  0, 1, 2, OR 3  (0 --> NO FLASH)
      INTEGER FERROR
C     ERROR FLAG    SET TO ONE IF 'DATA-VALID' SIGNAL DID NOT OCCUR,
C     SET TO TWO IF FLASH IS NOT 0, 1, 2, OR 3, SET TO THREE IF PIXEL
C     COUNT := 0, SET TO ZERO OTHERWISE
      INTEGER LIGHTT
C     VALUE PASSED TO THE ELECTRONICS TO FLASH A LIGHT
      INTEGER OKAY
C     'DATA-VALID' SIGNAL READ FROM ELECTRONICS
      INTEGER RESET
C     VALUE SENT TO ELECTRONICS TO RESET MUX BEFORE READING FIRST DATA
      INTEGER INEXT
C     VALUE SENT TO ELECTRONICS TO INCREMENT MUX
      INTEGER COUNT
C     AMOUNT OF DELAY TO GIVE ELECTRONICS TIME TO PROCESS DATA
C     (STARTS AT ZERO AND COUNTS UP. IF IT REACHES 20, FERROR
C     IS SET AND THE SUBROUTINE IS EXITED)
      INTEGER CLEAR
C     VALUE SENT TO ELECTRONICS TO CLEAR 'POOF'
      INTEGER PIXEL1,PIXEL2
C     RAW VALUES RETURNED FROM THE ELECTRONICS (COUNT OF PIXELS ABOVE
C     THRESHOLD VOLTAGE)
      INTEGER PN(2)
C     SAME AS PIXEL1, PIXEL2 (SEE EQUIVALENCE STATEMENTS)
      INTEGER VERT1,VERT2
C     RAW VERTICAL VALUES RETURNED FROM THE ELECTRONICS (SUM OF ROW NUMBERS
C     FOR PIXELS ABOVE THRESHOLD VOLTAGE)
      INTEGER VN(2)
C     SAME AS VERT1 AND VERT2 (SEE EQUIVALENCE STATEMENTS)
      INTEGER HORZ1,HORZ2
C     RAW HORIZONTAL VALUES RETURNED FROM ELECTRONICS (SUM OF COLUMN NUMBERS
C     FOR PIXELS ABOVE THRESHOLD VOLTAGE)
      INTEGER HN(2)
C      SAME AS HORZ1 AND HORZ2 (SEE EQUIVALENCE STATEMENTS)
      INTEGER*4 IP,IIP
C     PN(1) AND PN(2) CONCATENATED   IP RETURNED FROM VIDEO ELECTRONICS AS
C       I/O PORT A = PN(1) =                  I/O PORT B = PN(2) =
C              PIXEL1                                    PIXEL2
C     0000 OODD DDDD DDDD                  DDDD DDDD 0000 0000
C     \---------------- IP or IIP --------------------------/
C     WHERE D'S ARE DATA BITS   VALUE IS 0 TO 196607
      INTEGER*4 IV,IIV
C     VN(1) AND VN(2) CONCATENATED   IH RETURNED FROM VIDEO ELECTRONICS AS
C       I/O PORT A = VN(1) =                  I/O PORT B = VN(2) =
C             VERT1                                     VERT2
C     SDDD DDDD DDDD DDDD                  DDDD DDDD 0000 0000
C     \------------------- IV or IIV -----------------------/
C     WHERE S,D ARE DATA BITS   NEGATIVE NUMBERS ARE IN 2'S COMPLEMENT
C     NOTATION WITH S BEING THE SIGN BIT
      INTEGER*4 IH,IIH
C     HN(1) AND HN(2) CONCATENATED   (SEE IV, IIV ABOVE   DATA ORGANI-
C     ZATION OF HORZ1,HORZ1,IH,IIH,HN() PARALLELS VERT1,    ,VN()   )
      DOUBLE PRECISION TNOW,TSETL
C     TIME NOW AND WHEN SERVOS SHOULD HAVE STABILIZED
C
      EQUIVALENCE (PIXEL1,PN(1))
      EQUIVALENCE (PIXEL2,PN(2))
      EQUIVALENCE (IP,PN)
```

```
C
      EQUIVALENCE (VERT1,VN(1))
      EQUIVALENCE (VERT2,VN(2))
      EQUIVALENCE (IV,VN)
C
      EQUIVALENCE (HORZ1,HN(1))
      EQUIVALENCE (HORZ2,HN(2))
      EQUIVALENCE (IH,HN)
C
      COMMON/SRVOTH/DUMMY(6),TSETL,IDUMMY
C
      DATA INEXT/ 115720/
      DATA RESET/ 115623/
      DATA CLEAR/ 115500/
```



```
C
      FERROR=0
      COUNT=0
   10 CONTINUE
C      (* 910 - CHECK TIME *)
          CALL ESECS(TNOW)
       IF(TNOW LT TSETL) GO TO 10
C      (* RESET VIDEO PROCESSING ELECTRONICS *)
       CALL TNOUA(RESET,2)
       CALL TNOUA(CLEAR,2)
C      (* SET UP LIGHT DATA *)
       IF(LIGHT EQ 0) LIGHTT=( 115750)
       IF(LIGHT EQ 1) LIGHTT=( 115711)
       IF(LIGHT EQ 2) LIGHTT=( 115712)
       IF(LIGHT EQ 3) LIGHTT=( 115714)
      IF(LIGHT GE 0 AND LIGHT LE 3) GO TO 20
       FERROR=2
       GO TO 50
   20 CONTINUE
C         (* SET DATA AND CLEAR LIGHT *)
          CALL TNOUA(LIGHTT,2)
          CALL TNOUA(CLEAR,2)
C         (* WAIT FOR 'DATA-VALID' SIGNAL ('OKAY') TO BE SET NEGATIVE
   30     CONTINUE
             CALL DINCA(OKAY)
             COUNT=COUNT+1
C            (* QUIT IF WE'VE WAITED TOO LONG, OTHERWISE KEEP GOING *)
                IF(COUNT NE -1) GO TO 40
                   FERROR=1
                   GO TO 50
   40        CONTINUE
             IF(OKAY GE 0) GO TO 30
C         (* READ PIXEL COUNT *)
             CALL DINAA(PIXEL1)
             CALL DINBA(PIXEL2)
C         (* STRIP OFF JUNK BITS *)
             PN(1)=AND(PN(1)  0017777)
             PN(2)=AND(PN(2),  177400)
             IF(IP EQ 0) FERROR=3
C         (* INCREMENT MUX AND READ HORIZONTAL ACCCUMULATOR DATA *)
             CALL TNOUA(INEXT,2)
             IGOTIT=0
             CALL GOTIT(IGOTIT)
             CALL TNOUA(CLEAR,2)
             CALL GOTIT(IGOTIT)
             CALL DINAA(HORZ1)
             CALL DINBA(HORZ2)
C         (* STRIP OFF JUNK BITS *)
             HN(2)=AND(HN(2),  177400)
C         (* INCREMENT MUX AND READ VERTICAL ACCUMULATOR DATA *)
             CALL TNOUA(INEXT,2)
             CALL GOTIT(IGOTIT)
             CALL TNOUA(CLEAR,2)
             CALL GOTIT(IGOTIT)
             CALL DINAA(VERT1)
             CALL DINBA(VERT2)
C         (* STRIP OFF JUNK BITS *)
             VN(2)=AND(VN(2),  177400)
C         (* RESET MUX AND RETURN *)
             CALL TNOUA(INEXT,2)
             CALL TNOUA(CLEAR,2)
   50 CONTINUE
C      (* CLEAN UP & RETURN *)
          CALL TNOUA(RESET,2)
          CALL TNOUA(CLEAR,2)
          IIP=IP
          IIV=IV
          IIH=IH
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE GOTIT(IGOTIT)
C     (* 938 1 - WAIT FOR CHARACTER TO BE RECEIVED *)
C
C     CALLS
C        DINCA
C     CALLED BY
C        FLASH
C
C     INPUT
C        IGOTIT
C     OUTPUT
C        IGOTIT
C
      INTEGER IGOTIT
C        FLAG SAYING THE ELECTRONICS HAS RECEIVED A CHARACTER
      INTEGER ITEST
C        VALUE RECEIVED FROM ELECTRONICS (1 BIT IS FROM LAST CHARACTER SENT)
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      IF(IGOTIT EQ 0) GO TO 20
   10    CONTINUE
            CALL DINCA(ITEST)
            IF(AND(ITEST, 400) EQ IGOTIT) GO TO 10
         IGOTIT=0
         GO TO 40
   20 CONTINUE
   30    CONTINUE
            CALL DINCA(ITEST)
            IF(AND(ITEST, 400) EQ IGOTIT) GO TO 30
         IGOTIT= 400
   40 CONTINUE
      RETURN
```



```
C
      SUBROUTINE DINAA(ITEST)
C     (* 909 ? - DEBOUNCE DATA LINE *)
C
C     CALLS
C        DINA
C     CALLED BY
C        FLASH
C
C     INPUT
C        NONE
C     OUTPUT
C        ITEST
C
      INTEGER ITEST, JTEST
C        DATA FROM INPUT PORT
      INTEGER I
C        DO LOOP INDEX (NUMBER OF TIMES PORT REREAD)
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
   10 CONTINUE
         CALL DINA(ITEST)
         DO 20 I=1,50
            CALL DINA(JTEST)
            IF(JTEST NE ITEST) GO TO 10
   20    CONTINUE
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
      SUBROUTINE DINBA(ITEST)
C     (* 906 3 - DEBOUNCE DATA LINE *)
C
C     CALLS
C        DINB
C     CALLED BY
C        FLUSH
C
C     INPUT
C        NONE
C     OUTPUT
C        ITEST
C
      INTEGER ITEST,JTEST
C        DATA FROM INPUT PORT
      INTEGER I
C        DO LOOP INDEX (NUMBER OF TIMES PORT REREAD)
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
   10 CONTINUE
         CALL DINB(ITEST)
         ITEST=AND(ITEST, 177400)
         DO 20 I=1,50
            CALL DINB(JTEST)
            IF(AND(JTEST, 177400) NE ITEST) GO TO 10
   20    CONTINUE
      RETURN
C
      END
```



```
C
      SUBROUTINE PROPTR(DT,F,STATE,T)
C     (* 909 - 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,T0
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
      T0=T
   10 IF(TLEFT LE 0) GO TO 40
         STEP=AMIN1(STEPMX,TLEFT)
         TLEFT=TLEFT-STEP
C        (* 909 1 - COMPUTE K1 *)
            CALL COMPK1(F,K1,STATE,STEP,T)
C        (* 909 2 - COMPUTE K2 *)
            CALL COMPK2(F,K1,STATE,STEP,T,K2)
C        (* 909 3 - COMPUTE K3 *)
            CALL COMPK3(F,K2,STATE,STEP,T,K3)
C        (* 909 4 - 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
         T=T+STEP
         GO TO 10
   40 CONTINUE
      T=T0+DT
C     (* 909 5 - POINT SIMULATION CAMERA *)
         CALL POINT(T,STATE)
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
      SUBROUTINE COMPK1(F,K1,STATE,STEP,T)
C     (* 9.9.1 - 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     (* 909.2 - 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
```

```
C
      SUBROUTINE COMPK3(F,K2,STATE,STEP,T,K3)
C     (* 909 3 - 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     (* 909 4 - 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-2

```
C
      SUBROUTINE ESEC3(SECNDS)
C     (* 910 - RETURN SECONDS SINCE MIDNIGHT *)
C
C     CALLS
C        TIMDAT
C
C     CALLED BY
C        FLASH SERVO
C
C     INPUT
C        NONE
C     OUTPUT
C        SECNDS
C
      DOUBLE PRECISION SECNDS
C        ELAPSED SECONDS SINCE MIDNIGHT
      DOUBLE PRECISION S1,S2,S3
C        COMPONENTS OF SECNDS FROM WORDS IN ARRAY
      INTEGER ARRAY(11)
C        DATA FROM SYSTEM ROUTINE TIMDAT
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
      CALL TIMDAT(ARRAY,11)
      S1=FLOAT(ARRAY(4))*60
      S2=FLOAT(ARRAY(5))
      S3=FLOAT(ARRAY(6))/FLOAT(ARRAY(11))
      SECNDS=S1+S2+S3
      RETURN
C
      END
```

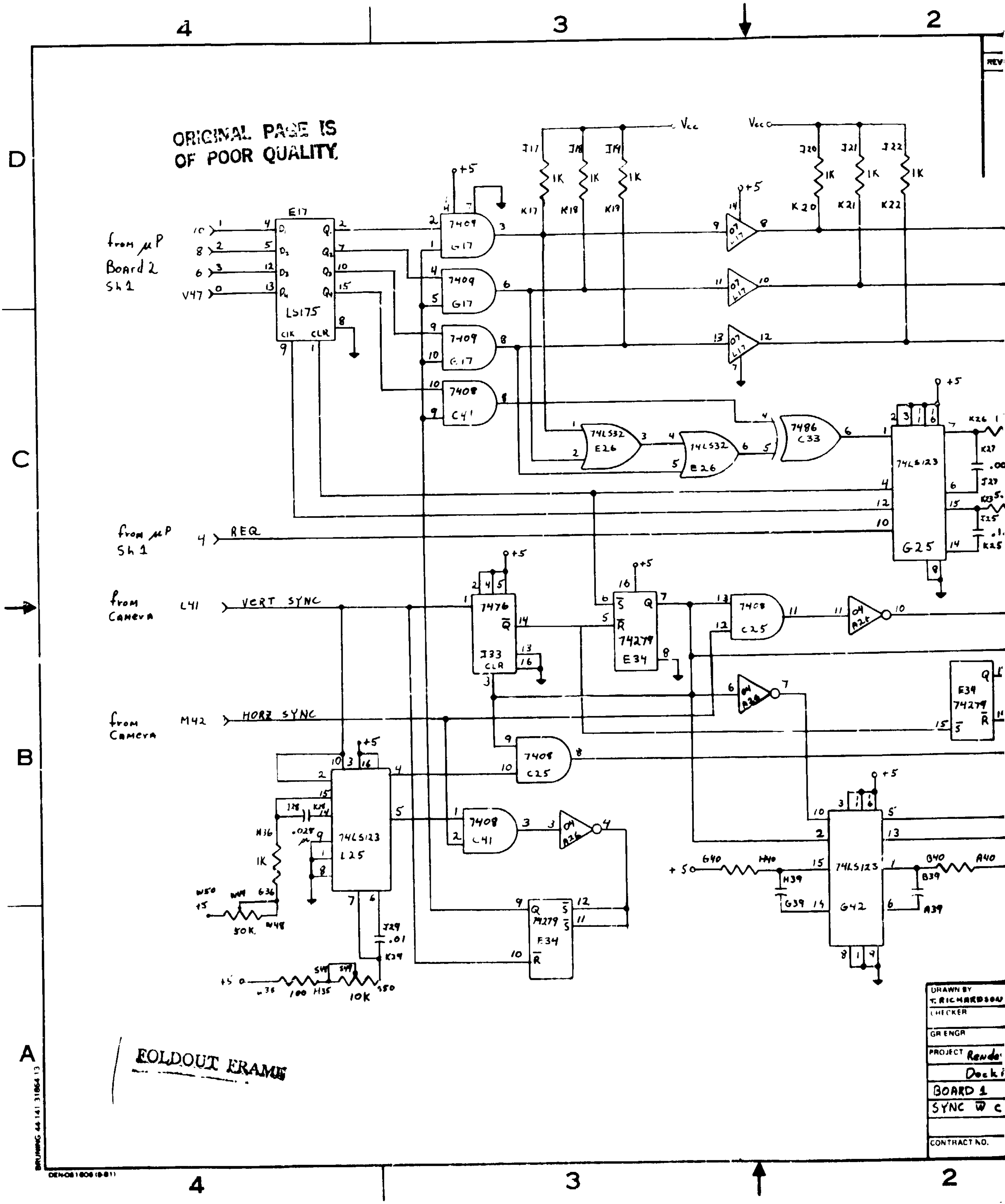


```
C
      SUBROUTINE DINCA(ITEST)
C     (* 911 - DEBOUNCE HANDSHAKE LINE *)
C
C     CALLS
C        DINC
C     CALLED BY
C        GOTIT,FLASH
C
C     INPUT
C        NONE
C     OUTPUT
C        ITEST
C
      INTEGER ITEST,JTEST
C        DATA FROM INPUT PORT
      INTEGER I
C        DO LOOP INDEX (NUMBER OF TIMES TO RETRY READING PORT)
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
   10 CONTINUE
         CALL DINC(ITEST)
         ITEST=AND(ITEST,100400)
         DO 20 I=1,50
            CALL DINC(JTEST)
            IF(AND(JTEST,100400) NE ITEST) GO TO 10
   20    CONTINUE
      RETURN
C
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

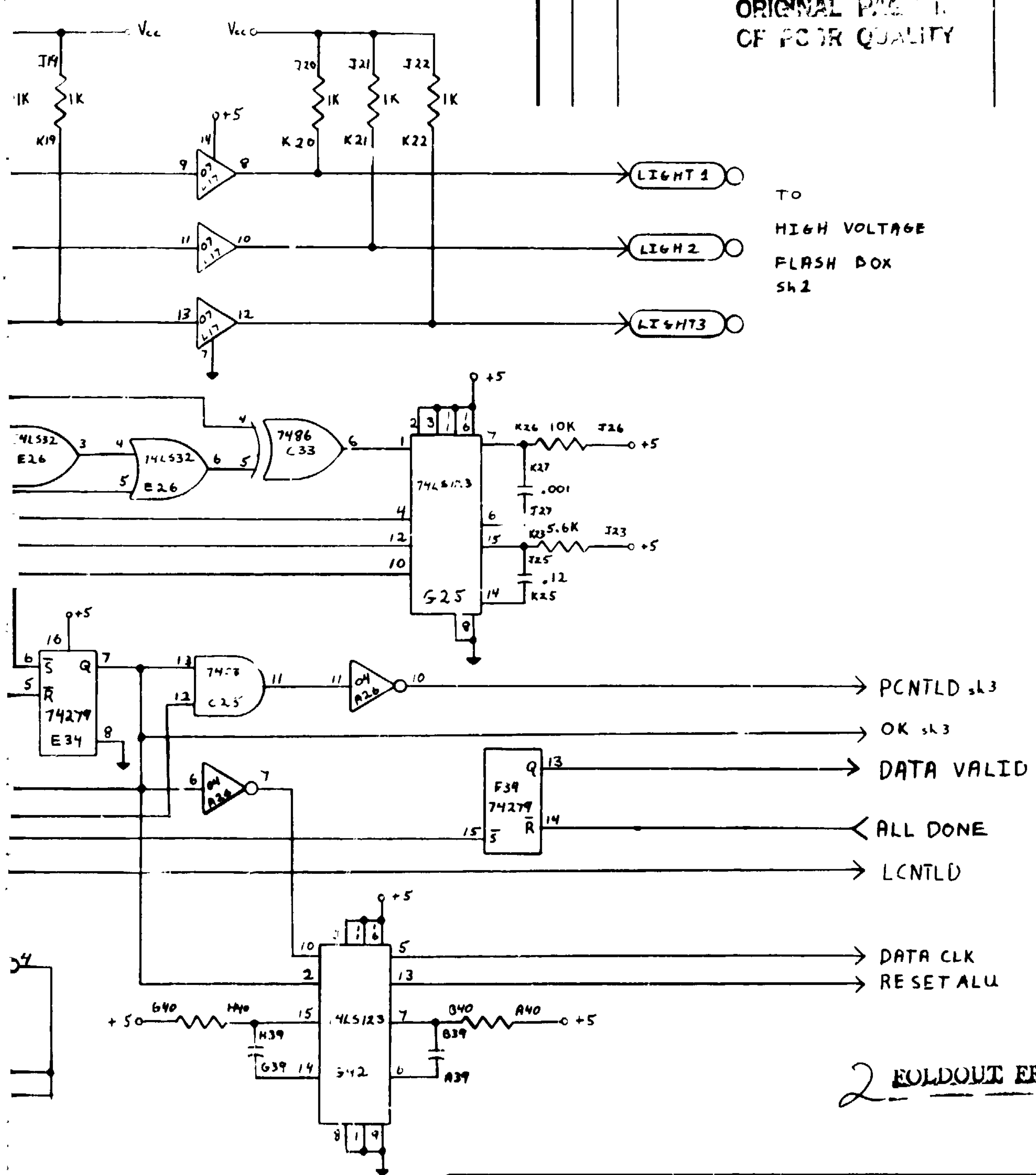
APPENDIX B--SCHEMATIC DIAGRAMS

The diagrams provided in this appendix are for the video processing electronics described in Chapter VI. Reference designators of the form E1, A12, C9, etc identify the column and row on the wire-wrap card where pin one of an integrated circuit is located.


4
3
2
D
C
B
A
from μP
Board 2
Sh 1
E17
LS175
7409
G17
7408
C41
7407
L17
Vcc
1K
74LS32
E26
7486
C33
74LS123
G25
from μP
Sh 1
REQ
from Camera
L41
VERT SYNC
7476
J33
74279
E34
7408
C25
04
A26
from Camera
M42
HORZ SYNC
74LS123
L25
50K
10K
G42
DRAWN BY
T. RICHARDSON
CHECKER
GR ENGR
PROJECT
BOARD 1
SYNC
CONTRACT NO.

REV	ZONE	DESCRIPTION	DATE	APPROVED

ORIGINAL PAGE IS OF POOR QUALITY

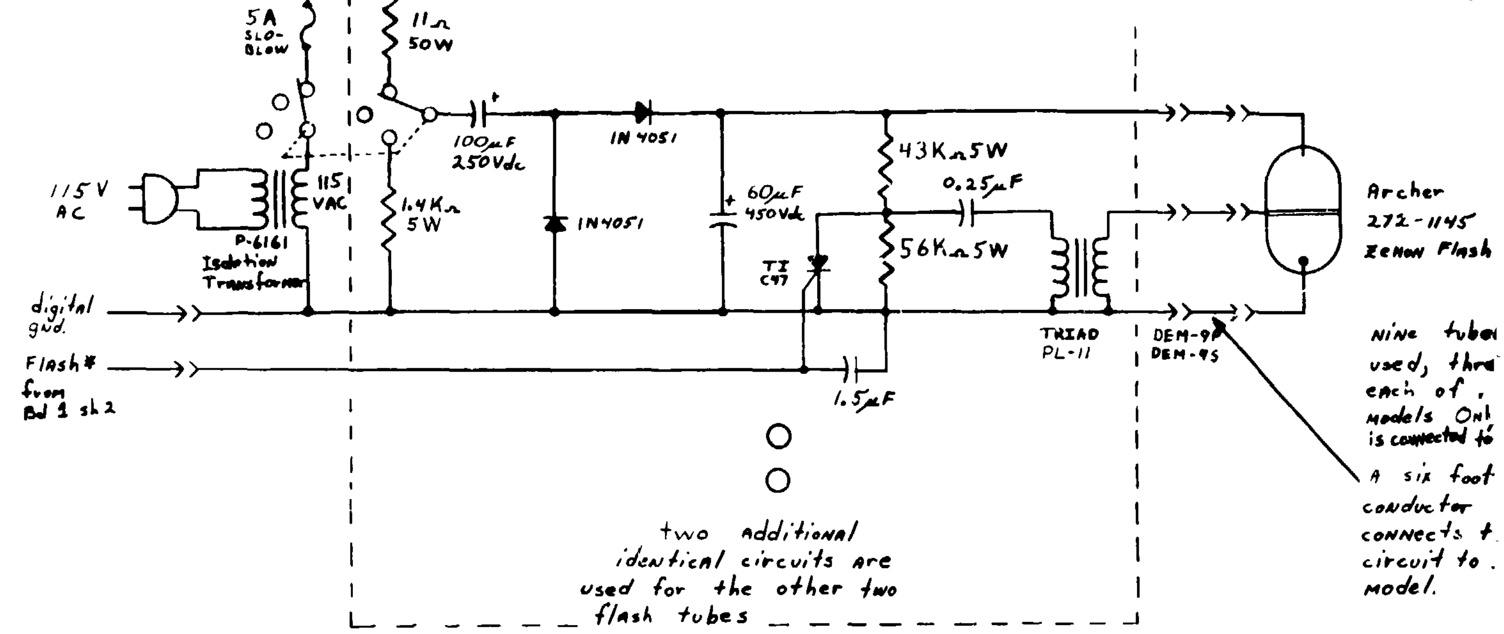


2 FOLDOUT FRAME

DRAWN BY T. RICHARDSON DEPT 0570 DATE 3/31/83	MARTIN MARIETTA CORPORATION DENVER AEROSPACE POST OFFICE BOX 179 DENVER COLORADO 80201
CHECKER	
GR ENGR	
PROJECT Rendezvous and Docking	
BOARD 1	
SYNC W CAMERA	SIZE C · FSCM NO 04236 · DRAWING NO B-2
CONTRACT NO	SCALE · SHEET 2 of 6

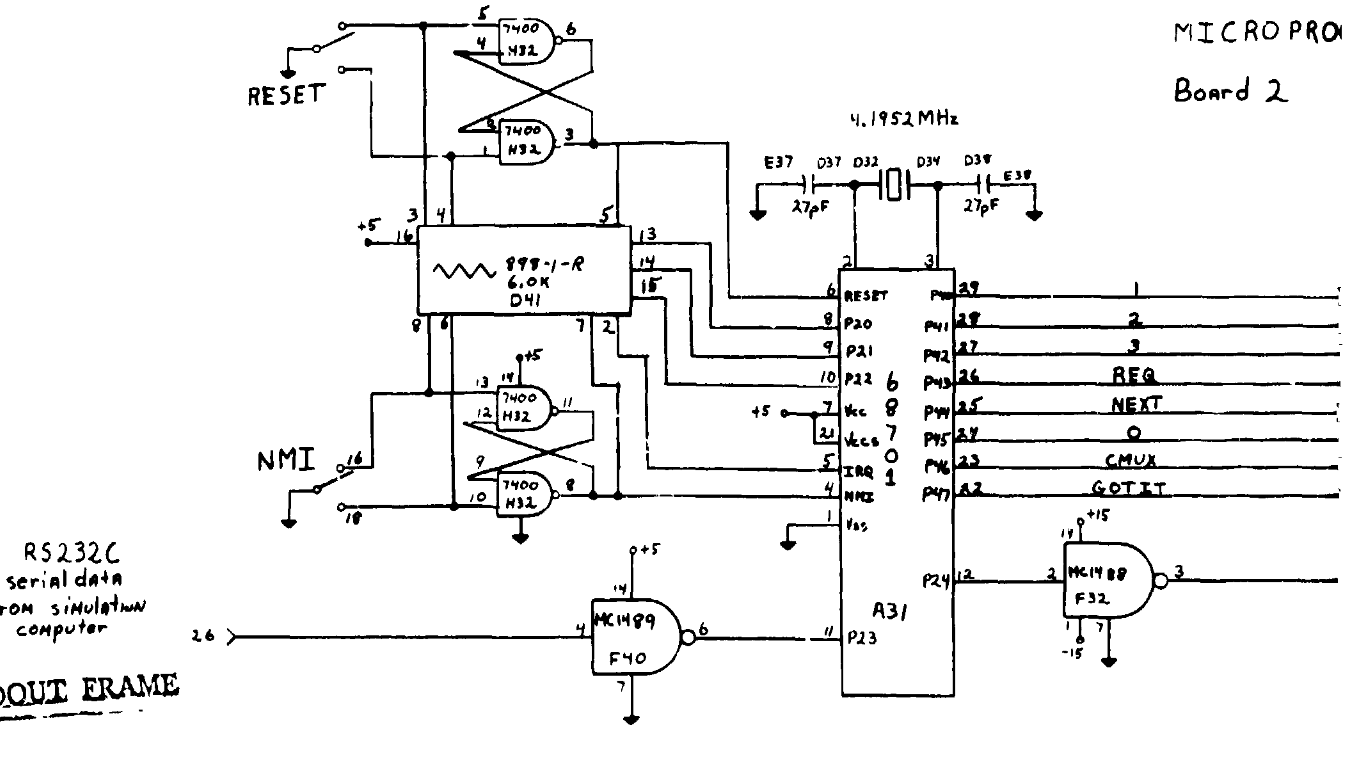
ORIGINAL PAGE IS
OF POOR QUALITY

FLASH CIRCUIT



MICROPRO

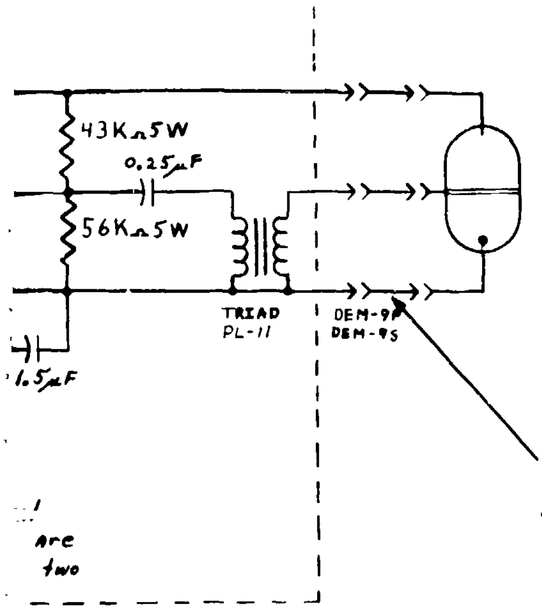
Board 2



FOLDOUT FRAME

REVISIONS				
SYM	ZONE	DESCRIPTION	DATE	APPROVED

CUIT



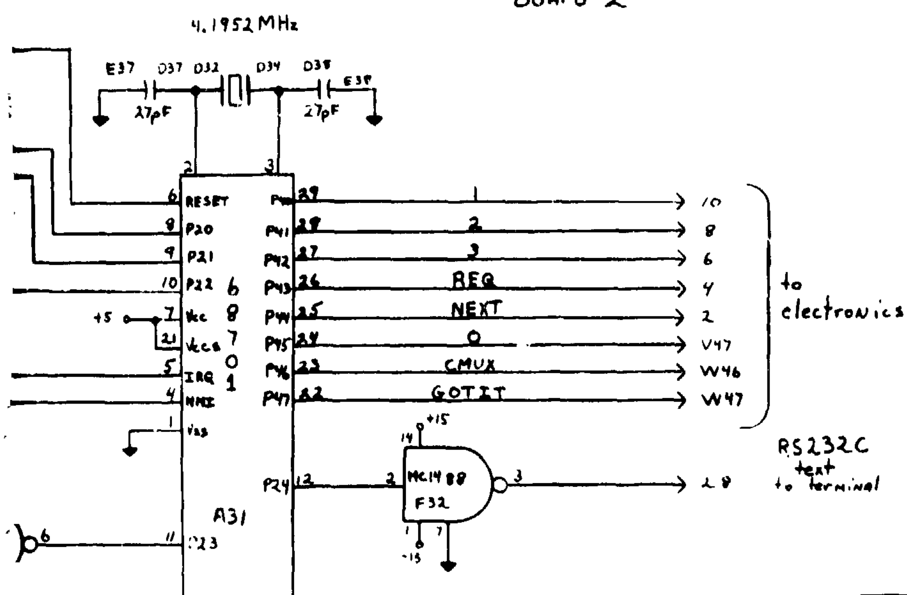
Nine tubes are used, three in each of three models. Only one model is connected to the electronics at a time.

A six foot, nine conductor cable connects the flash circuit to the model.

ORIGINAL PAGE IS
OF POOR QUALITY

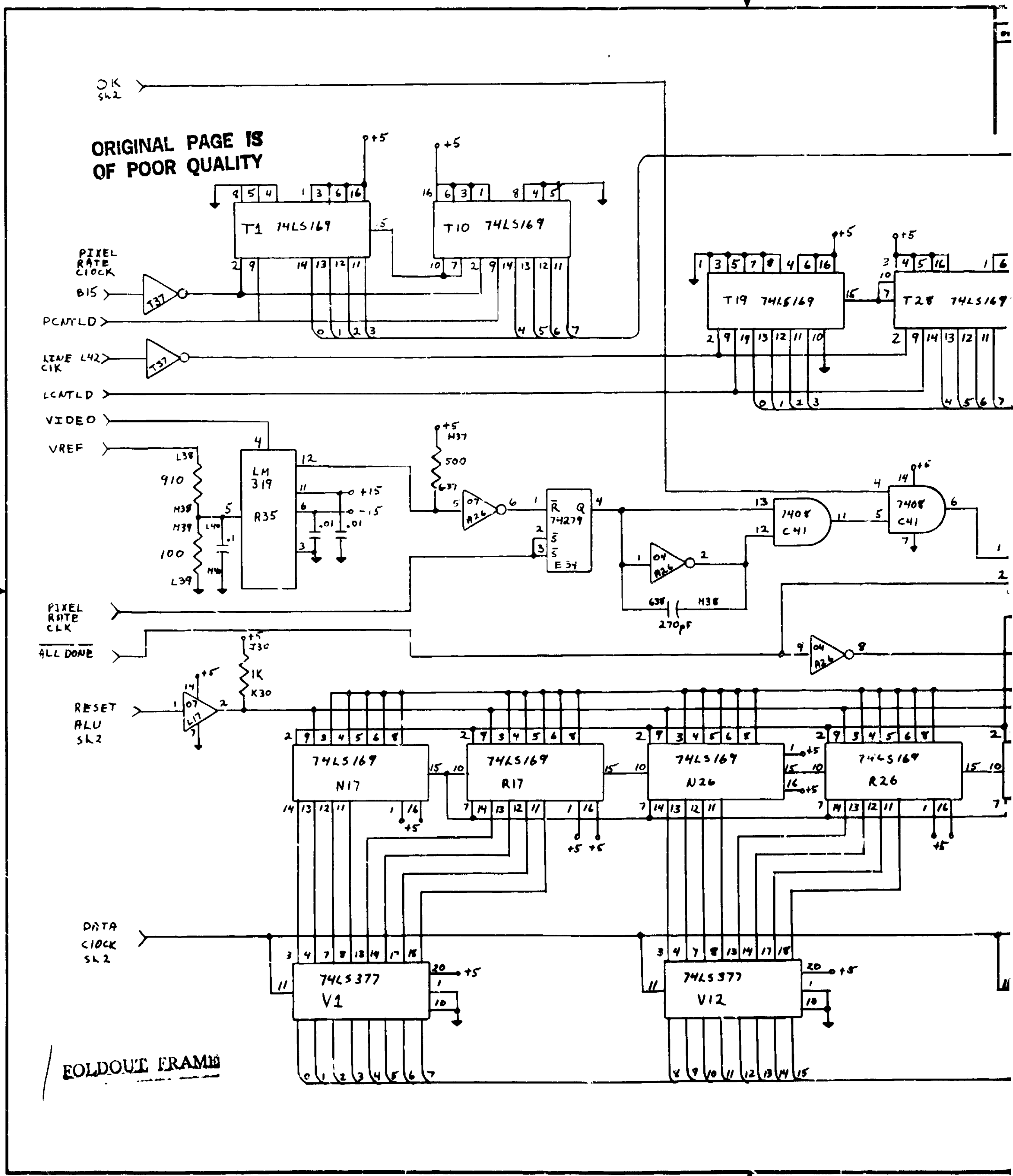
MICROPROCESSOR

Board 2



2 FOLDOUT FRAME

SIZE	CODE IDENT NO		
C	04236	RENDEZVOUS + DOCKING	B-3
SCALE		T. E. RICHARDSON	SHEET 1 of 6

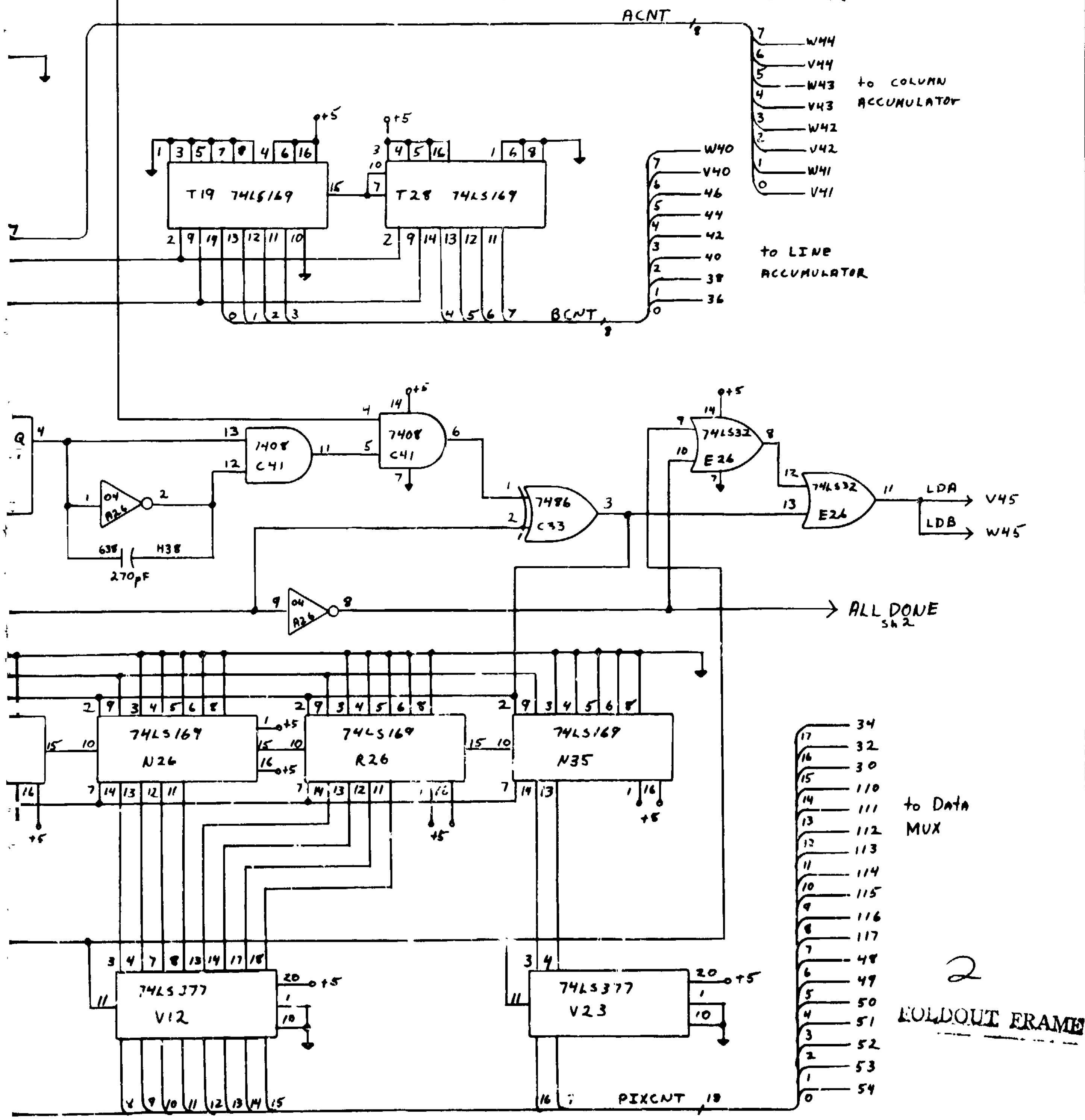


ORIGINAL PAGE IS OF POOR QUALITY

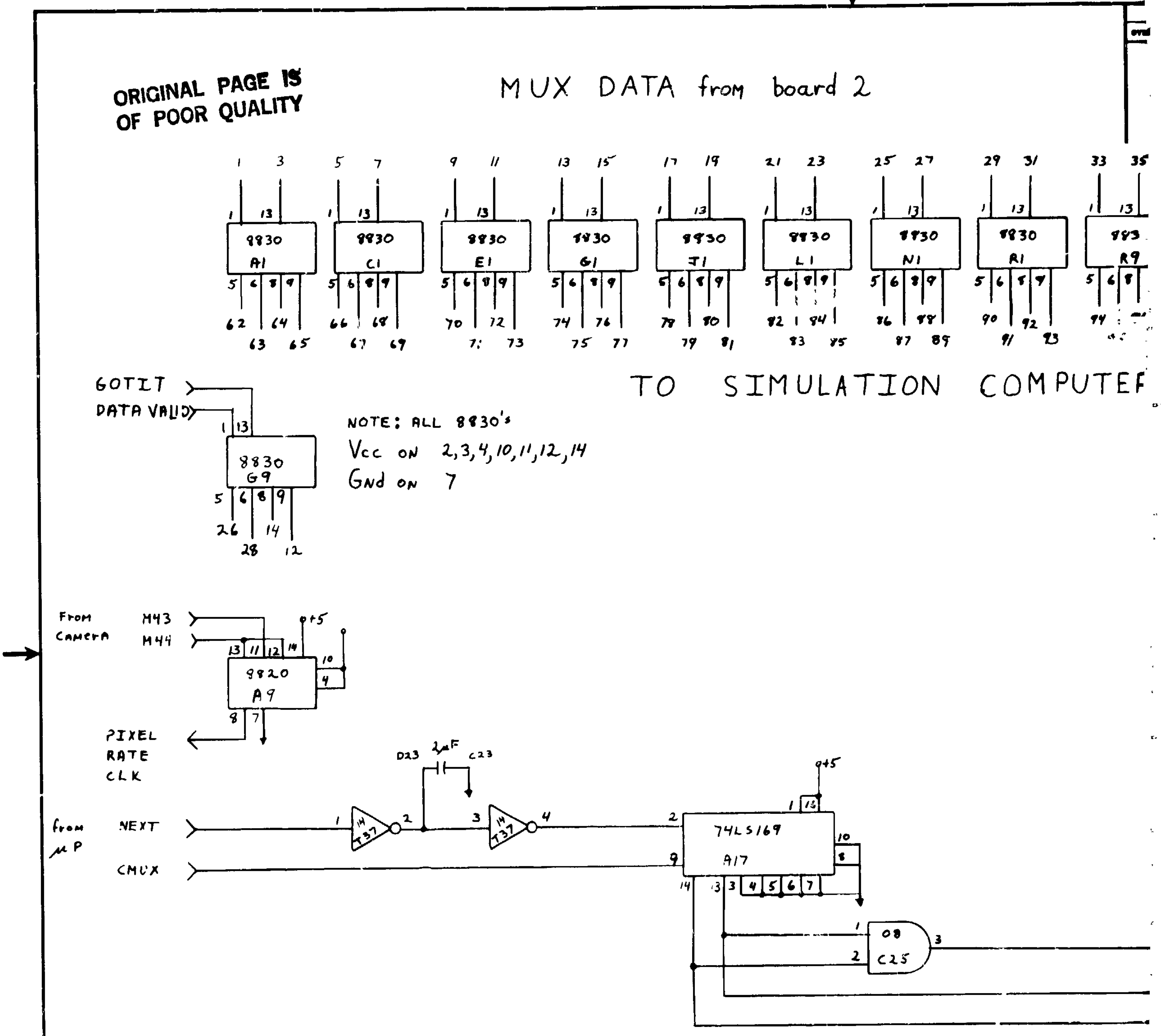
FOLDOUT FRAME

REVISIONS				
SYM	ZONE	DESCRIPTION	DATE	APPROVED

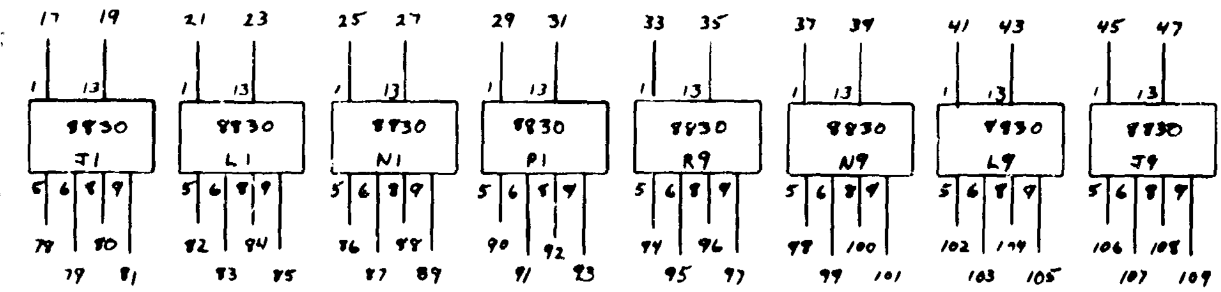
ORIGINAL PAGE IS OF POOR QUALITY



SIZE	CODE IDENT NO		
C	04236	RENDEZVOUS + DOCKING COUNTERS + COMPARATOR BOARD 2	B-4
SCALE		T.E. RICHARDSON	SHEET 3 of 6

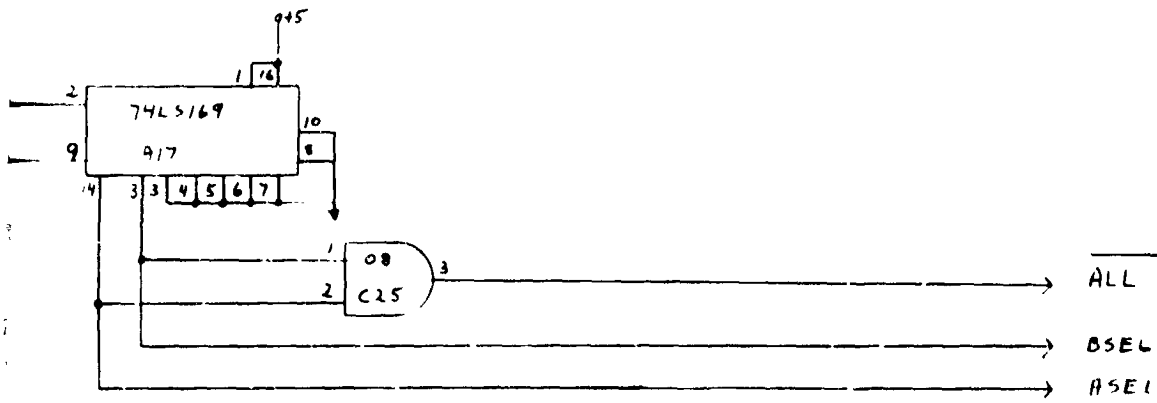

MUX DATA from board 2
8830 A1
8830 C1
8830 E1
8830 G1
8830 J1
8830 L1
8830 N1
8830 R1
TO SIMULATION COMPUTER
GOTIT
DATA VALID
8830 G9
NOTE: ALL 8830's
Vcc ON 2,3,4,10,11,12,14
Gnd ON 7
From Camera
M43
M44
+5
8820 A9
PIXEL RATE CLK
D23
2μF
C23
from μP
NEXT
CMUX
T37
74LS169
A17
08
C25

TA from board 2



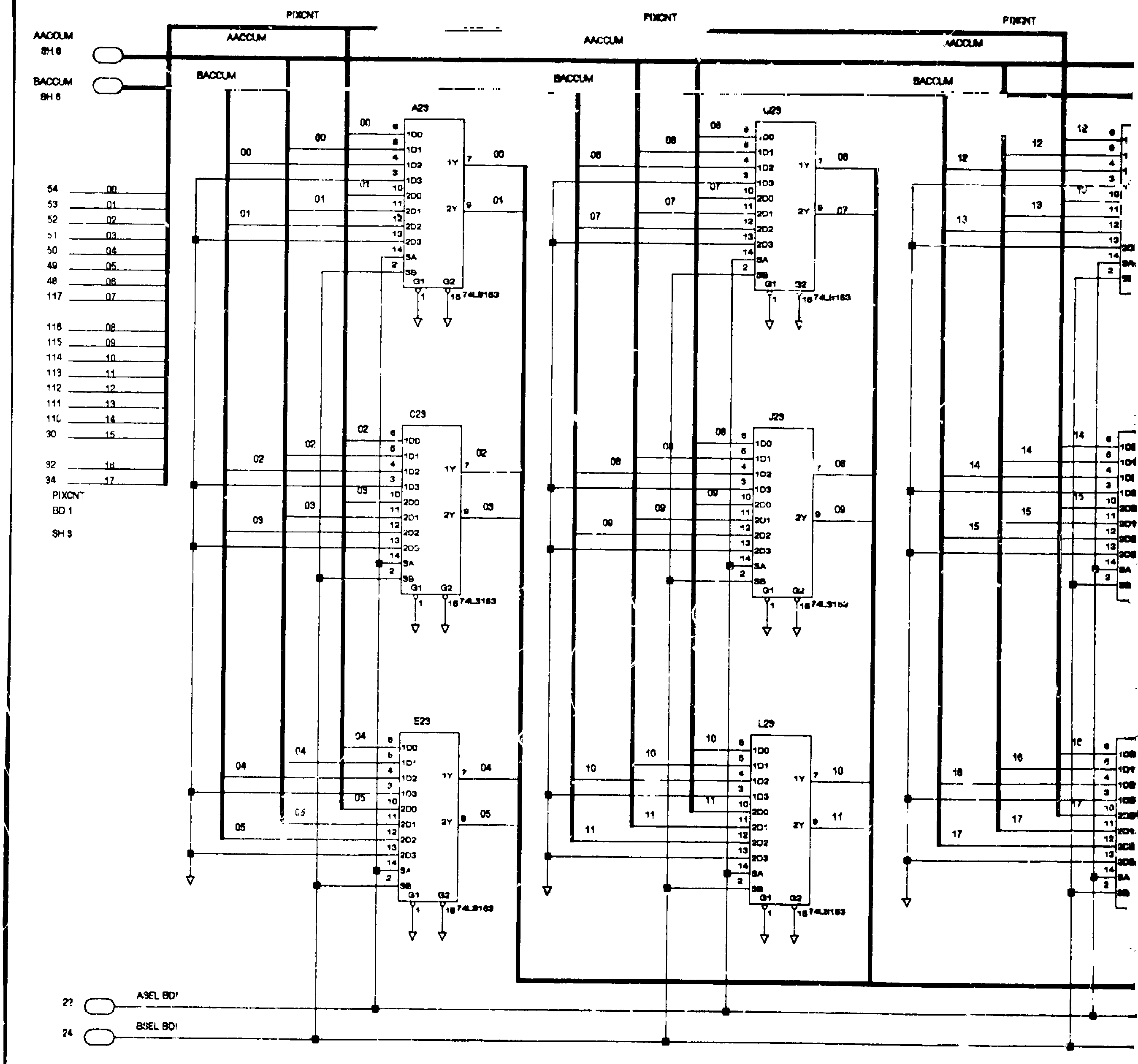
TO SIMULATION COMPUTER

FOLDOUT FRAME



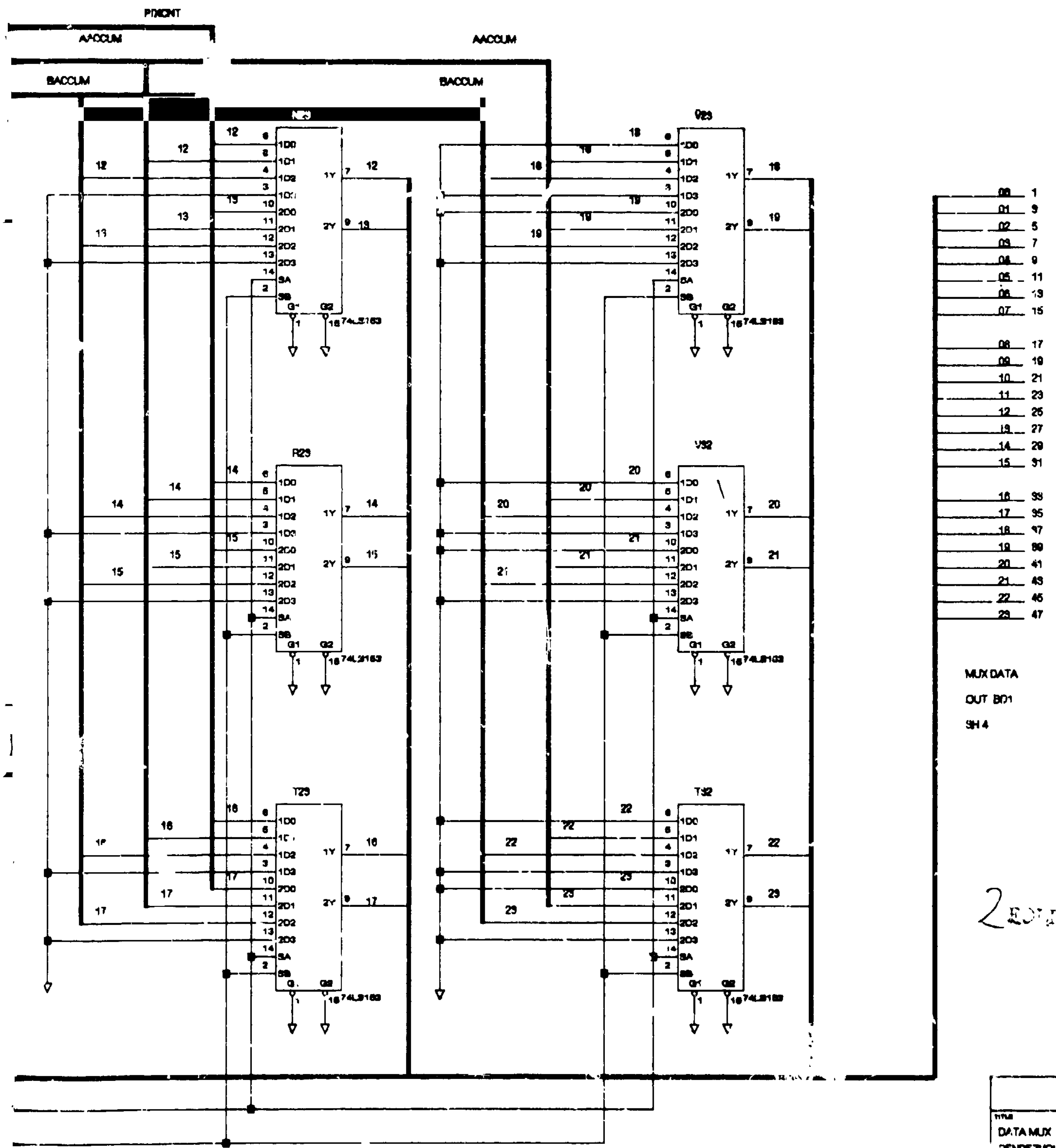
SIZE	CODE IDENT NO		
C	34236	RENDEZVOUS + DOCKING DRIVERS + MUX counter BOARD	B-5
SCALE		T.E. RICHARDSON	SHEET 4 of 6

ORIGINAL PAGE IS
OF POOR QUALITY



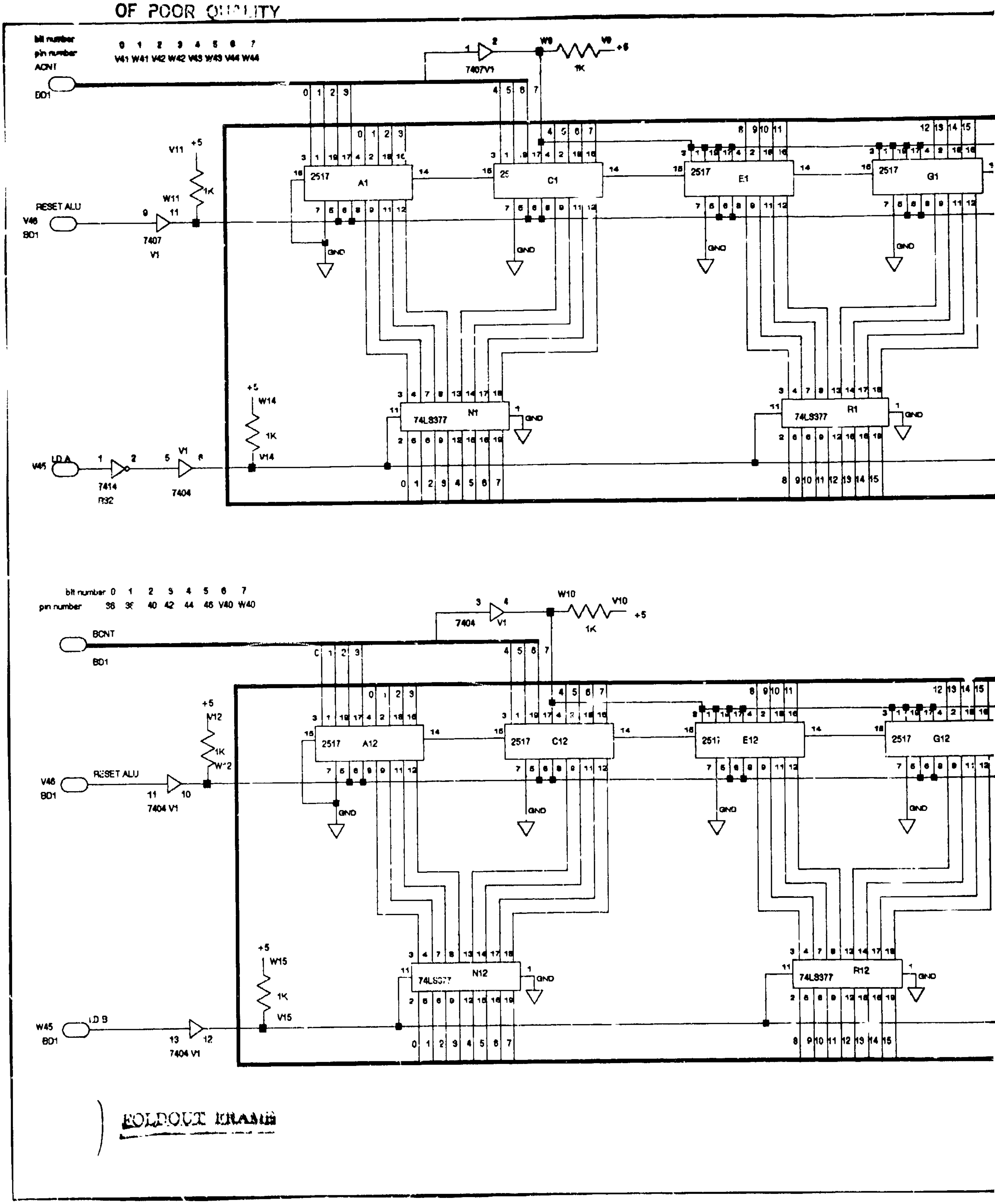
FOLDOUT FRAME

ORIGINAL [illegible]
OF POOR [illegible]



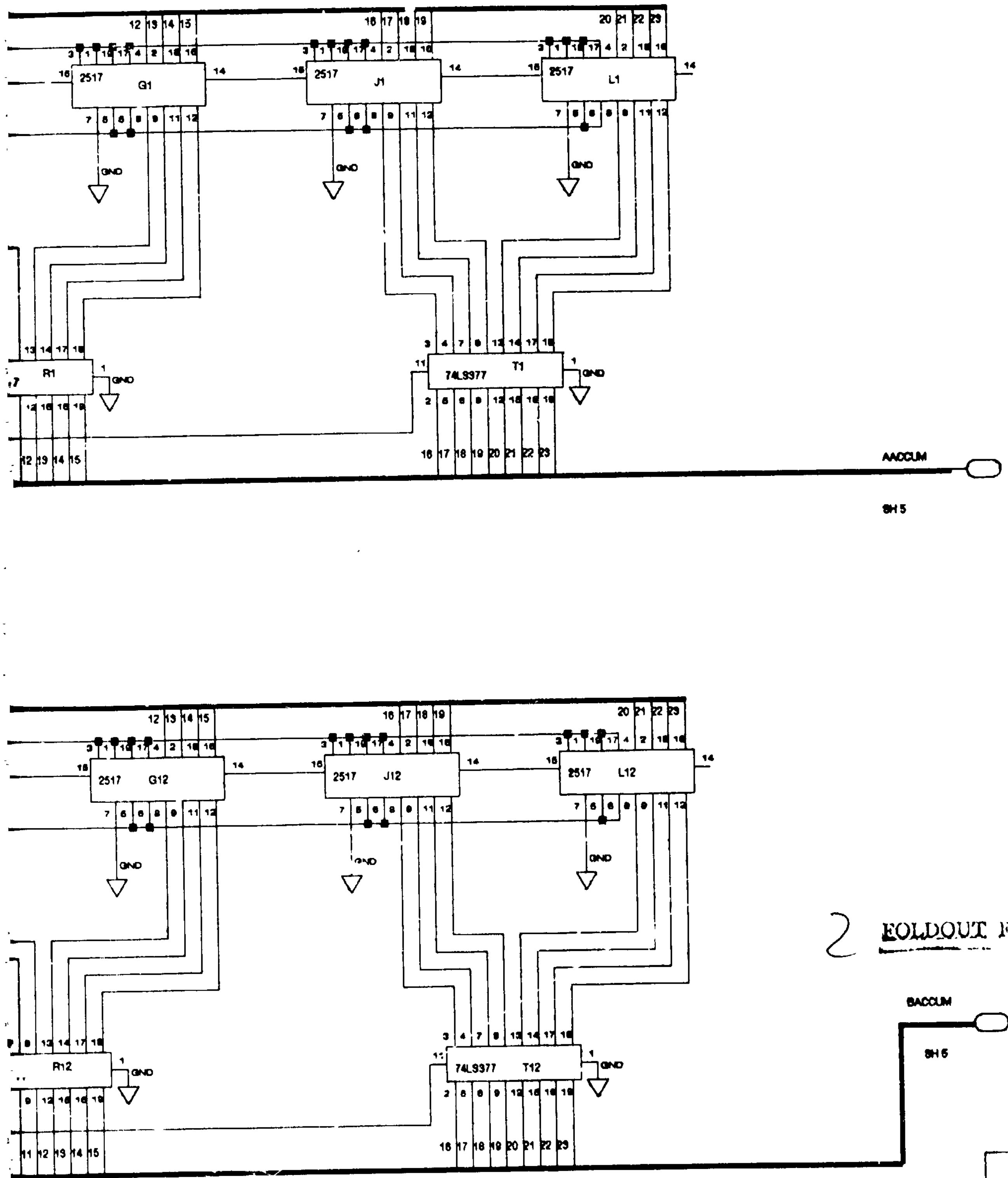
DATA MUX
RENDEZVOUS and DOCKING
BOARD 2
T. E. RICHARDSON

ORIGINAL PAGE IS
OF POOR QUALITY



FOLDOUT FRAME

ORIGINAL PAGE IS OF POOR QUALITY



FOLDOUT FRAME

B-7

TITLE	
A AND B ACCUMULATORS	
RENDEZVOUS AND DOCKING	
T. E. RICHARDSON	
BOARD 2	
SIZE D	DRAWING NO.

APPENDIX C--MICROPROCESSOR FIRMWARE

The program listing in this appendix is the program executed by the microprocessor on one of the video processing electronics circuit cards. The program flow chart is shown in Figure VI-6. The program's function is to intercept commands intended for the video processing electronics from the data stream between the computer and the operator's terminal. The commands are characters preceded by an ASCII "escape" character. When an escape character is received, the microprocessor sends the next character it receives to its parallel output port F4. All other characters are relayed to the terminal.