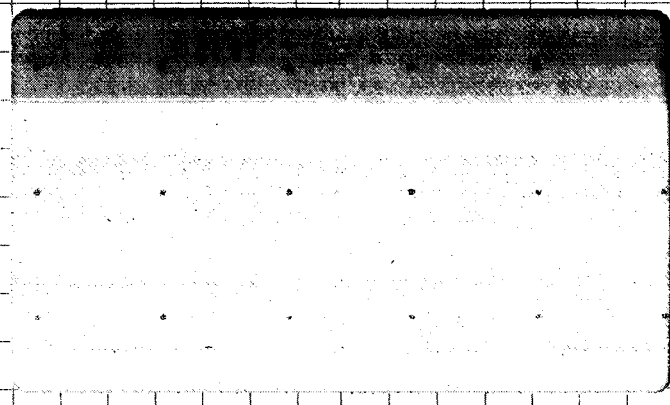


(NASA-CR-179147) CONTACT DYNAMICS MATH MODEL Interim Report (Control Dynamics Co.) 95 p Avail: NTIS HC A05/MF A01 CSCL 09B

N87-25801

Unclas
G3/61 0085374

Office Park South, Suite 304, 600 Boulevard South, Huntsville, Alabama 35802 (205) 882-2650

CONTROL DYNAMICS COMPANY

CONTACT DYNAMICS MATH MODEL

INTERIM REPORT
APRIL, 1986

SPONSORED BY:

GEORGE C. MARSHALL SPACE FLIGHT CENTER
MARSHALL SPACE FLIGHT CENTER, ALABAMA 35812

UNDER

CONTRACT NO. NAS8-36570

CONTRIBUTORS:

DR. JOHN R. GLAESE
PATRICK A. TOBBE

PREPARED BY:

CONTROL DYNAMICS COMPANY
OFFICE PARK SOUTH, SUITE 304
600 BOULEVARD SOUTH
HUNTSVILLE, ALABAMA 35802

TABLE OF CONTENTS

1.0 Introduction

The Space Station Mechanism Test Bed consists of a hydraulically driven, computer controlled six degree of freedom (DOF) motion system with which docking, berthing, and other mechanisms can be evaluated. Shown in Figure 1, the Test Bed facility simulates the docking of two orbiting vehicles, a target and chaser. The chase vehicle docking mechanism is attached to the six DOF motion system, while the target's mechanism is fixed to the ceiling of the facility. A force and moment sensor is mounted in the ceiling above the target docking fixture. Contact forces and moments due to hardware mechanism operation are measured in six degrees of freedom and provided to the simulation host computer to enable representation of orbital contact dynamics. The old contact dynamics simulation model in use represented a restricted case in which one body was considered much larger than the second and therefore unaffected by the docking forces and moments.

The purpose of this report is to describe the development of a generalized math model to eliminate the restrictive assumptions of the old model. This new model represents the relative motion between two rigid orbiting vehicles. These vehicles are acted on by forces and moments from vehicle contact, attitude control systems, and gravity. The resulting model allows motion in six degrees of freedom for each body, with no vehicle size limitation. The new computer simulation is modular to facilitate the addition or deletion of various parts of the model.

This report derives the translational and rotational equations of motion for the vehicles in the model. The method used to transform the forces and moments from the sensor location to the vehicles' centers of mass is also explained. The interface between the dynamics math model and the overall

CONTACT DYNAMICS USING 6 DOF MOTION SYSTEM

Figure 1

simulation consists of the relative position, velocity, and orientation between the vehicles, and the contact forces and moments. This interface is thoroughly discussed. Two math models of docking mechanisms, a simple translational spring and the RMS end effector, are presented along with simulation results. The translational spring model is used in an attempt to verify the simulation with compensated hardware in the loop results. Finally, recommendations are made at the end of the report to upgrade and improve the simulation.

2.0 Equations of Motion

Figure 2 depicts two rigid vehicles undergoing small deviations from a nominal circular orbit. The target vehicle is denoted by the number one and the chaser is labeled number two. The following coordinate frames and points are used by the simulation. Coordinate frame E is an inertial frame fixed at the center of the earth. In this frame, the X and Z axes are fixed in the equatorial plane with Z pointing at the vernal equinox. Y points to the North Pole. The local vertical frame, L, rotates on a circular path of radius Ro at a constant angular rate of ω_L. The Z axis of the local vertical frame lies along $\underset{\sim}{Ro}$, the position of the local vertical frame with respect to the inertial frame. The X axis lies in the plane of the orbit and is tangent to the orbit path, while Y is the orbit normal. The orientation of the L frame with respect to the E frame is defined through a 2-3-2 Euler angle sequence of the following angles. The first two angles, the ascending node angle and the angle of inclination, are constant. The third angle, the orbit angle, is equal to its initial value plus the product of ω_L and time. The V1 and V2 frames are fixed at the centers of mass of the target and chase vehicles, respectively, and located with respect to the local vertical by the position vectors $\underset{\sim}{R1}$ and $\underset{\sim}{R2}$. The D1 and D2 frames are docking port frames located at arbitrary positions with respect to the vehicles' centers of mass by the vectors $\underset{\sim}{RD1V1}$ and $\underset{\sim}{RD2V2}$. A sensor coordinate frame, S1, is located arbitrarily on the target vehicle with respect to V1 by the vector $\underset{\sim}{RS1V1}$.

The inertial angular velocities of the target and chase vehicles are labeled $\underset{\sim}{\omega_1}$ and $\underset{\sim}{\omega_2}$. m_1 and m_2 are the masses of the target and chase vehicles, respectively, while I1 and I2 are the inertia dyadics about the centers of mass.

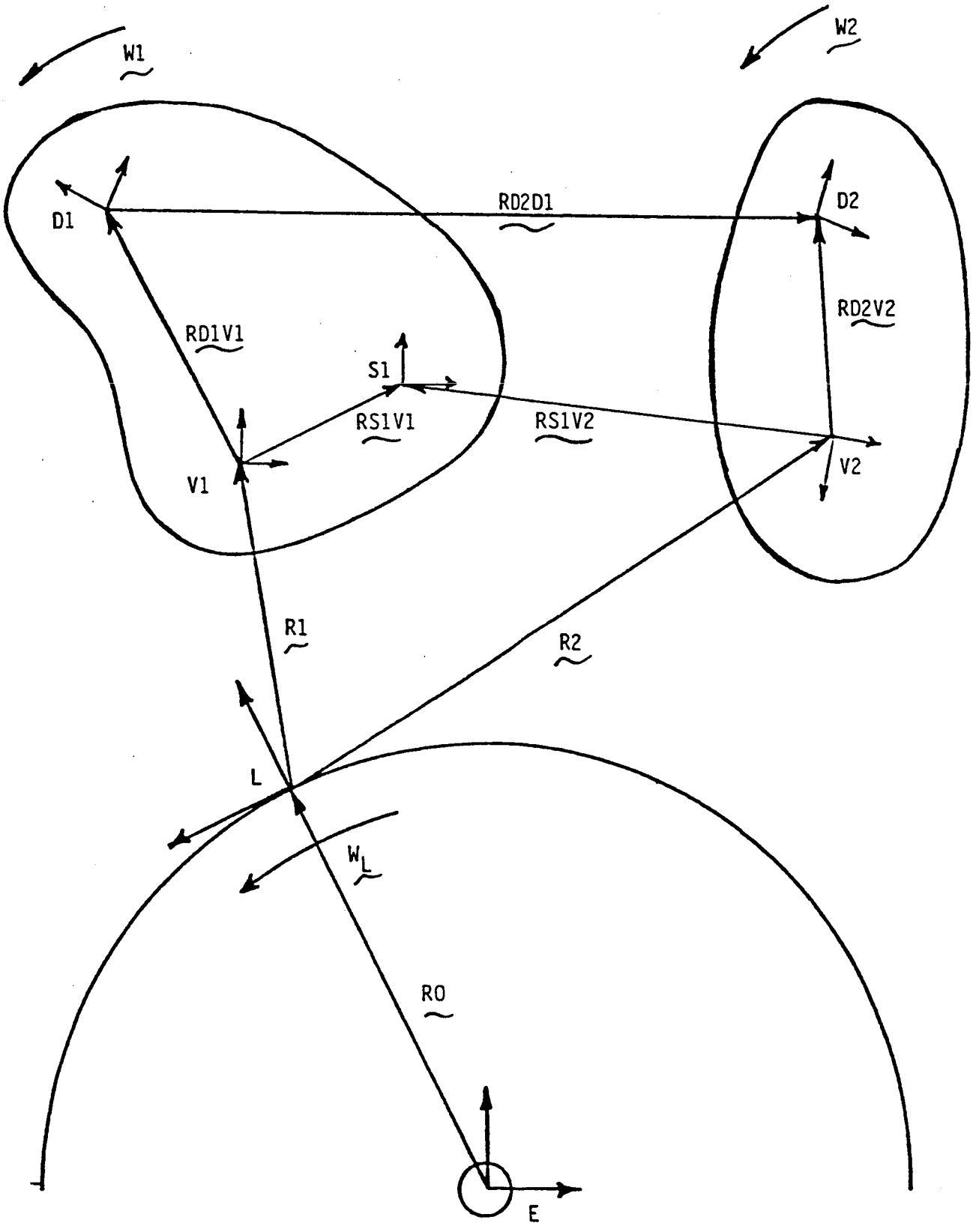


Figure 2

The translational equations of motion will now be derived for the target vehicle. Those of the chase vehicle will be identical. All forces, except gravity, such as vehicle contact and control forces, are included in the vector $\underset{\sim}{f}$. Referring to Figure 3, the coordinate frame U lies in the plane of the orbit such that the $\underset{\sim}{U3}$ axis is parallel to the Y axis of the local vertical frame. The position vector $\underset{\sim}{Ro}$ is therefore defined by equation (1).

$$\underset{\sim}{Ro} = Ro\cos\omega_L t\,\underset{\sim}{U1} + Ro\,\mathrm{SIN}\,\omega_L t\,\underset{\sim}{U2} \tag{1}$$

t = time

The constant angular rate of the local vertical frame, ω_L, is given by equation (2)

$$\omega_L = \left(\frac{GM}{Ro^3}\right)^{1/2} \tag{2}$$

G = Universal Constant of Gravitation

M = Mass of the Earth

Differentiating equation (1) twice with respect to time leads to equation (3).

$$\ddot{\underset{\sim}{Ro}} = -Ro\,\omega_L^2\left(\cos\omega_L t\,\underset{\sim}{U1} + \mathrm{SIN}\,\omega_L t\,\underset{\sim}{U2}\right) \tag{3}$$

$\underset{\sim}{R1}$ is the position vector of the target vehicle center of mass with respect to the local vertical. From Newton's third law, the sum of the forces acting on the vehicle must equal the time rate of change of the linear momentum.

$$m_1\left(\ddot{\underset{\sim}{Ro}} + \ddot{\underset{\sim}{R1}}\right) = \underset{\sim}{Fg} + \underset{\sim}{f} \tag{4}$$

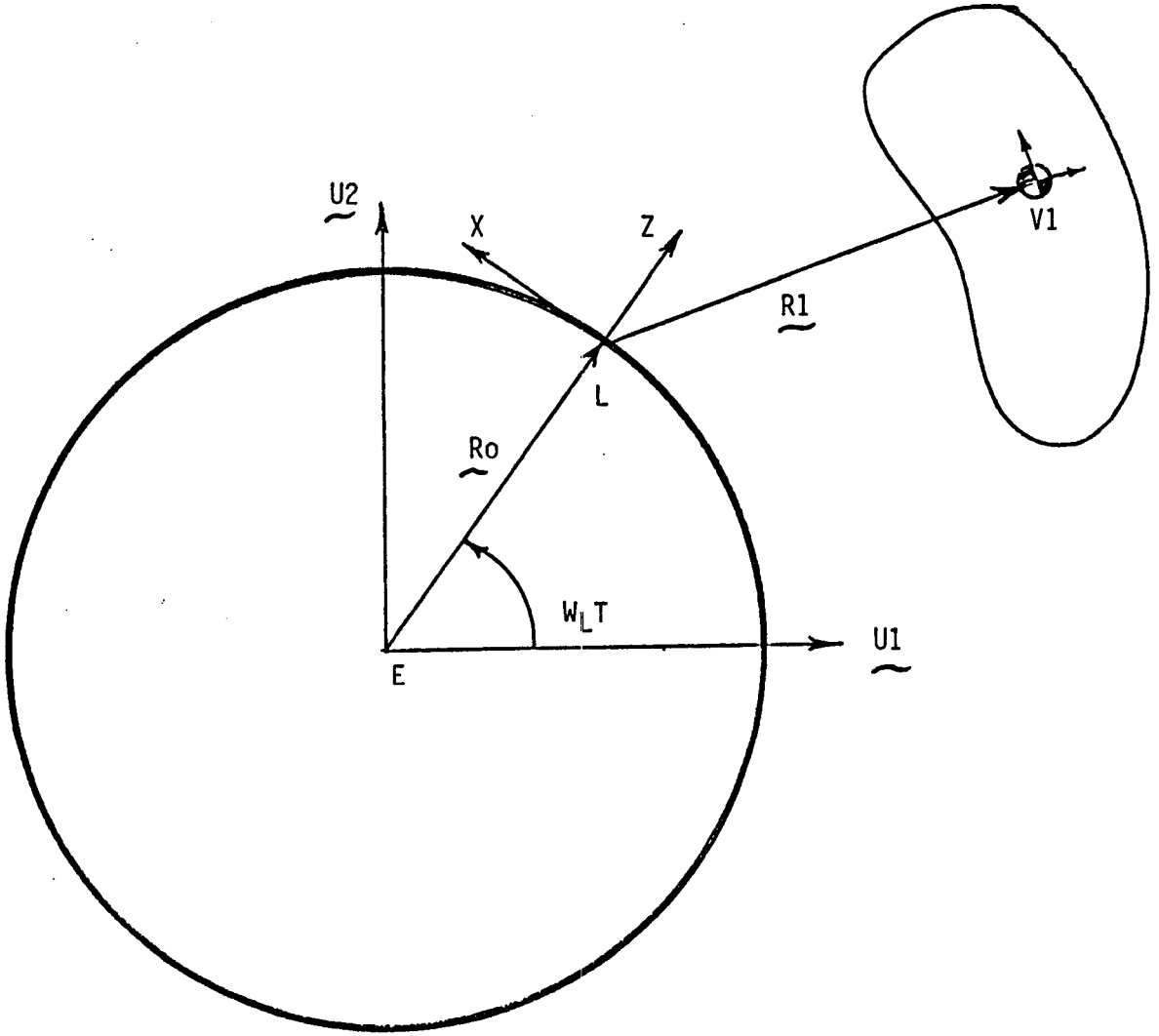


Figure 3

The force due to gravity is defined in equation (5)

$$\underset{\sim}{F_g} = \frac{-GMm_1}{|\underset{\sim}{R_o}+\underset{\sim}{R1}|^3}\left(\underset{\sim}{R_o}+\underset{\sim}{R1}\right) \tag{5}$$

If it is assumed that $|\underset{\sim}{R1}| << |\underset{\sim}{R_o}|$, then using the Binomial theorem and keeping only first order terms in $\underset{\sim}{R1}$, the gravitational force can be written as

$$\underset{\sim}{F_g} = -GMm_1\left(\underset{\sim}{R_o}+\underset{\sim}{R1}\right)\left(\frac{1}{R_o^3}\right)\left(1-\frac{3\underset{\sim}{R1}\cdot\underset{\sim}{R_o}}{R_o^2}\right) \tag{6}$$

Using equation (2), equation (6) can be written as in (7).

$$\underset{\sim}{F_g} = -m_1\omega_L^2\left(\underset{\sim}{R_o}+\underset{\sim}{R1}\right)\left(1-\frac{3\underset{\sim}{R1}\cdot\underset{\sim}{R_o}}{R_o^2}\right) \tag{7}$$

Substituting (7) into (4) yields equation (8).

$$\ddot{\underset{\sim}{R1}} = -\omega_L^2\,\underset{\sim}{R1}\left(1-\frac{3\underset{\sim}{R1}\cdot\underset{\sim}{R_o}}{R_o^2}\right)+\omega_L^2\,\underset{\sim}{R_o}\left(\frac{3\underset{\sim}{R1}\cdot\underset{\sim}{R_o}}{R_o^2}\right)+\frac{\underset{\sim}{f}}{m_1} \tag{8}$$

In order to express the components of (8) in the rotating local vertical frame, the following notation is now defined.

$$\dot{\underset{\sim}{R}} = \mathring{\underset{\sim}{R}} + \underset{\sim}{\omega_L}\times\underset{\sim}{R} \tag{9}$$

$\dot{\underset{\sim}{R}}$ is the time derivative of $\underset{\sim}{R}$ as seen by an observer fixed in inertial space. $\mathring{\underset{\sim}{R}}$ is the time derivative of $\underset{\sim}{R}$ as seen by an observer in the local vertical frame. Since $\underset{\sim}{\omega_L}$ is constant, (9) can be differentiated with respect to time to produce (10).

$$\ddot{\underset{\sim}{R}} = \ddot{\underset{\sim}{R}}_o + \underset{\sim}{\omega}_L \times (\underset{\sim}{\omega}_L \times \underset{\sim}{R}) + 2\,\underset{\sim}{\omega}_L \times \dot{\underset{\sim}{R}} \quad , \tag{10}$$

Using equations (8) and (10), and ignoring second order effects, the following equations are generated to govern translational motion of the vehicle.

$$\ddot{X}1 = -2\,\omega_L \dot{Z}1 + fx/m1 \tag{11}$$

$$\ddot{Y}1 = -\omega_L^2\, Y1 + fy/m1 \tag{12}$$

$$\ddot{Z}1 = 2\omega_L \dot{X}1 + 3\omega_L^2\, Z1 + fz/m1 \tag{13}$$

In these equations, X1, Y1, and Z1 define the components of $\underset{\sim}{R1}$ with respect to the local vertical frame. This formulation avoids the numerical problems of referencing the vehicle position from the center of the earth. This is especially useful for contact dynamics simulations which require the relative position between the vehicles.

The rotational equations of motion for the target and chase vehicles are the well known Newton - Euler equations shown in (14).

$$I \cdot \dot{\underset{\sim}{\omega}} = -\underset{\sim}{\omega} \times I \cdot \underset{\sim}{\omega} + \underset{\sim}{T} \tag{14}$$

$\underset{\sim}{\omega}$ = Angular Velocity of Vehicle

I = Inertia Dyadic about Vehicle Center of Mass

$\underset{\sim}{T}$ = Torques Acting about Vehicle Center of Mass

The torque vector consists of gravity gradient, contact, and attitude control system torques. Equation (14) is expressed in a vehicle fixed frame so that the inertia dyadic will remain constant. This equation does not limit the body axes to align with the principal moment of inertia axes. In other words, the inertia matrix does not have to be diagonal. The gravity gradient torque acting on the body is expressed in equation (15).

$$\underset{\sim}{T_g} = 3\omega_L^2 \frac{\underset{\sim}{R_o}}{R_o} \times I \cdot \frac{\underset{\sim}{R_o}}{R_o} \qquad (15)$$

The Newton - Euler equations are not restricted to small angle rotations of the vehicle.

3.0 Force/Moment Transformation

The math model assumes that a force/momemt sensor is located at some arbitrary position on the target vehicle which corresponds to its position on the ceiling fixture of the 6 DOF facility. However, since the sensor location is not at the center of mass of either vehicle, the measured values must be transferred to these points.

The sensor will measure the resultant forces and torques acting on the target vehicle due to hardware contact. If contact occurs at more than one point, the force vector measured will be the sum of these contact forces. The moment measured will be the resultant torque acting on the target vehicle about the sensor location due to the multiple contact forces and moments. The force and moment transformation method presented here is independent of the number of contact points.

It is assumed that the inertial forces acting on the moving parts of the docking mechanism are negligible. This assumption is valid when the moving mass of the docking mechanism is small compared to the mass of the vehicle. If this assumption is not violated, then the contact forces and moments are equal and opposite between the target and chase vehicles.

Referring to Figure 2, $\underset{\sim}{RS1V1}$ and $\underset{\sim}{RS1V2}$ are the position vectors from the target and chase vehicle centers of mass to the sensor. $\underset{\sim}{Fs}$ and $\underset{\sim}{Ms}$ are now defined as the force and moment values from the sensor. $\underset{\sim}{Ft}$, $\underset{\sim}{Mt}$, $\underset{\sim}{Fc}$ and $\underset{\sim}{Mc}$ are defined as the forces and moments acting at the target and chase vehicles' centers of mass. The forces and moments from the sensor can now be transferred to the vehicles' centers of mass through the following equations.

$$\underset{\sim}{F_t} = \underset{\sim}{F_s} \tag{16}$$

$$\underline{F_c} = -\underline{F_s} \tag{17}$$

$$\underline{M_t} = \underline{M_s} + \underline{RS1V1} \times \underline{F_s} \tag{18}$$

$$\underline{M_c} = -\underline{M_s} - \underline{RS1V2} \times \underline{F_s} \tag{19}$$

4.0 Relative Vehicle Motion

The purpose of the math model is to generate the relative motion between the target and chase vehicles when acted on by contact, gravity, and control system forces and torques. This data will be used to generate the commands for the six DOF motion system.

As seen in Figure 2, $\underset{\sim}{RD2D1}$ is the position of the chase docking port with respect to the target docking port. It is calculated through equation (20).

$$\underset{\sim}{RD2\,D1} = \underset{\sim}{R2} + \underset{\sim}{RD2V2} - \underset{\sim}{R1} - \underset{\sim}{RD1V1} \tag{20}$$

The relative vehicle velocity is defined as the velocity of the chaser docking port as seen by an observer on the target docking port. Equation (21) defines the relationship between the time derivative of a vector as seen by observers fixed in rotating coordinate frames.

$$\dot{\underset{\sim}{A}} = \overset{\circ}{\underset{\sim}{A}} + \underset{\sim}{\omega_{CB}} \times \underset{\sim}{A} \tag{21}$$

$\dot{\underset{\sim}{A}}$ = Time derivative of $\underset{\sim}{A}$ with respect to B frame.

$\overset{\circ}{\underset{\sim}{A}}$ = Time derivative of $\underset{\sim}{A}$ with respect to C frame.

$\underset{\sim}{\omega_{CB}}$ = Angular velocity of C frame with respect to B frame.

Equation (20) is differentiated with respect to time to produce (22)

$$\dot{\underset{\sim}{RD2D1}} = \dot{\underset{\sim}{R2}} + \dot{\underset{\sim}{RD2V2}} - \dot{\underset{\sim}{R1}} - \dot{\underset{\sim}{RD1V1}} \tag{22}$$

Using (21), the following relations are now presented.

$$\underset{\sim}{\dot{RD2D1}} = \underset{\sim}{\overset{\circ}{RD2D1}} + \underset{\sim}{\omega_1} \times \underset{\sim}{RD2D1} \tag{23}$$

$\underset{\sim}{\overset{\circ}{RD2D1}}$ = Time derivative of $\underset{\sim}{RD2D1}$ with respect to target docking port, D1, frame.

$$\underset{\sim}{\dot{R2}} = \underset{\sim}{\overset{\diamond}{R2}} + \underset{\sim}{\omega_L} \times \underset{\sim}{R2} \tag{24}$$

$\underset{\sim}{\overset{\diamond}{R2}}$ = Time derivative of $\underset{\sim}{R2}$ with respect to local vertical frame.

$$\underset{\sim}{\dot{R1}} = \underset{\sim}{\overset{\diamond}{R1}} + \underset{\sim}{\omega_L} \times \underset{\sim}{R1} \tag{25}$$

$\underset{\sim}{\overset{\diamond}{R1}}$ = Time derivative of $\underset{\sim}{R1}$ with respect to local vertical frame.

The position vectors $\underset{\sim}{RD1V1}$ and $\underset{\sim}{RD2V2}$ are fixed in the vehicle frames. Therefore, the time derivatives of these vectors are given as

$$\underset{\sim}{\dot{RD1V1}} = \underset{\sim}{\omega_1} \times \underset{\sim}{RD1V1} \tag{26}$$

$$\underset{\sim}{\dot{RD2V2}} = \underset{\sim}{\omega_2} \times \underset{\sim}{RD2V2} \tag{27}$$

Substituting (23) through (27) into (22) produces the desired result shown in equation (28).

$$\underset{\sim}{\overset{\circ}{RD2D1}} = \underset{\sim}{\overset{\diamond}{R2}} + \left(\underset{\sim}{\omega_L} - \underset{\sim}{\omega_1}\right) \times \underset{\sim}{R2} + \left(\underset{\sim}{\omega_2} - \underset{\sim}{\omega_1}\right) \times \underset{\sim}{RD2V2} - \underset{\sim}{\overset{\diamond}{R1}} - \left(\underset{\sim}{\omega_L} - \underset{\sim}{\omega_1}\right) \times \underset{\sim}{R1} \tag{28}$$

The angular velocity of the chase vehicle with respect to the target vehicle is given in equation (29)

$$\underset{\sim}{\omega}_{21} = \underset{\sim}{\omega}_2 - \underset{\sim}{\omega}_1 \tag{29}$$

5.0 Simulation Description

The computer code of the dynamics math model is written entirely in FORTRAN 77 and is modular in form. There is a driver routine called by the host simulation which controls the math model and acts as the interface to the host simulation.

Figure 4 depicts the program flow as directed by the driver in the dynamic loop. The first pass through the program initializes the necessary variables. The program input consists of raw force and moment sensor data and OMV control system stick commands. The force and moment sensor data is filtered and transferred to the vehicles' centers of mass. The stick commands are used to generate control forces and torques acting on the chase vehicle. These forces and moments are used in the equations of motion which are solved numerically for the relative position and velocity between the vehicles.

The following nomenclature used throughout the code is now presented. Position and translational velocity vectors begin with the letter R. The position vector RABC is defined as the vector from point B to point A expressed in C frame coordinates. RABCD is the time derivative of RABC. Angular velocity vectors begin with the letters OM. OMABC is defined as the angular velocity of A with respect to B in C frame coordinates. The transformation matrix [AB] transforms vectors from the B frame to A frame as shown in equation (30)

$$\underset{\sim}{R}^{A} = [AB]\,\underset{\sim}{R}^{B} \tag{30}$$

The labels used to describe the vehicles and geometry of Figures 2 and 3 are the same as those in the code.

DYNAMICS PROGRAM FLOW

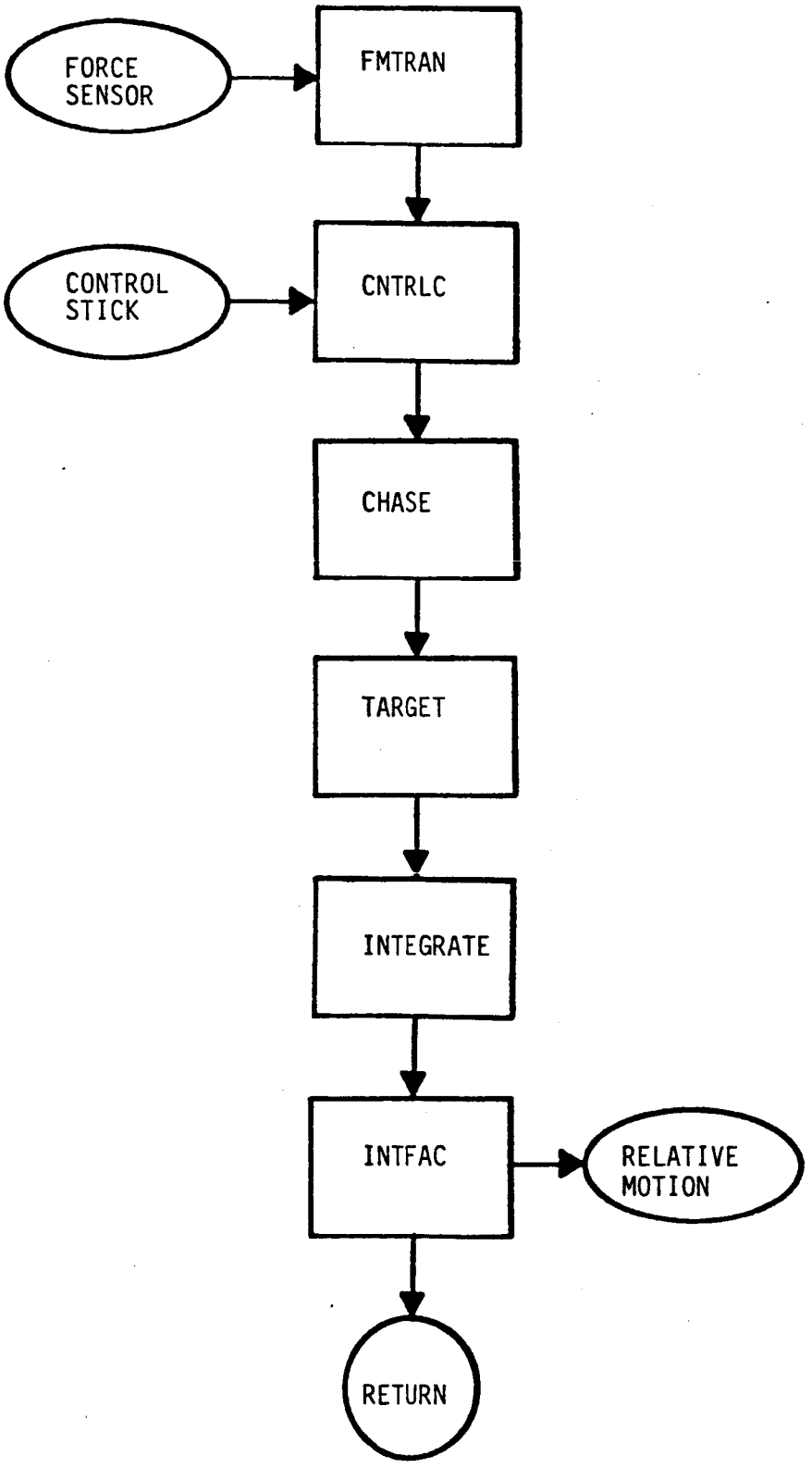


Figure 4

The program common blocks and subroutines will now be discussed. A detailed list of definitions of the global variables is included in Appendix 1. However, most variable definitions are given in comments throughout the code. A program listing of the math model is in Appendix 2.

There are four files which contain the common blocks and global variables used throughout the code. These files are simply included in the necessary subroutines of the program.

The file labeled cmmnbb.f is composed of all variables needed to interface the dynamics routine with the host simulation. The common block PASS consists of the variable JPASS, a program counter used in the OMV control system. The variable JPASS counts the number of passes through the dynamic loop. Common block INT1 contains the variable NNN. NNN may have four different integer values from -1 to 2. A value of -1 is used for the first initialization of the program, while a value of 0 is for program initialization between runs. In the dynamic loop, this variable has the value of 1, and a value of 2 is used for an end of run print of thruster on times. JPASS and NNN are generated in the host simulation to be used in the dynamic math model.

The common block CADIN is composed of the seven element array KDAT. The first 6 elements are X, Y, Z translation and roll, pitch, yaw rotation stick commands for the OMV control system. The seventh element is an attitude rate hold switch. The common block sensor consists of the ten element array KIN and the integer flags NOUSE, IFREEZ, IFLAG, and IPOT. The first 6 elements of KIN are X, Y, Z, and roll, yaw, pitch signals from the force and moment sensor. Element 7 is the freeze flag and element 8 is used to reset the offset forces. Element 9 is the IPOT flag. The dynamic model uses the KIN

array to calculate the contact forces and moments, and to generate the four integer flags. Common block TIME is composed of the variables T, DT, DT2, and TD. T is time. DT and DT2 are the simulation cycle and half cycle times. TD is a time delay used in the OMV control system. All of these variables are set in the host simulation to be used by the dynamics routine. The common block RELATIVE contains the output of the dynamics math model. The vector RD2D1D1 is the position vector from the target docking port to the chase docking port, in target docking port coordinates. RD2D1D1D is the velocity of the chaser docking port with respect to the target docking port, in target docking port coordinates. OMV2V1V2 is the angular velocity of the chase vehicle with respect to the target vehicle, in chase vehicle coordinates. D1D2 is the transformation matrix from the chaser docking port frame to the target docking port frame.

The common blocks of the file cmmn.f, STAT, EARTH, and TRANS, define the size of the state vectors and necessary orbital parameters. In particular, OMO is the constant angular velocity of the local vertical frame, and XLE is the transformation matrix from the E to L frame.

The files cmmn1.f and cmmn2.f contain the variables pertaining to the target and chase vehicles, respectively. Appendix 1 should be consulted for a listing of the variable definitions of these files. The equivalence statements in these files should be noted. Y1, Y2, YD1, and YD2 are defined as the state and state dot vectors for the target and chase vehicles. The state vectors are composed of the vehicle angular momentum vectors, quaternions, and translational velocity and position vectors.

The file cmmncon.f contains the variables associated with the NASA OMV control system.

The file dyn.f is the subroutine DYNAMIC. This subroutine is the driver for the math model as shown in Figure 4. This subroutine will be called at time T by the host simulation, with no argument list, in order to obtain the relative vehicle motion data at time T + DT. The subroutine uses the 3-1 integration scheme, with the previous state dot values stored in the arrays YDO1 and YDO2. The first pass through DYNAMIC initializes the state and state dot vectors, vehicle mass properties and geometry, control system parameters, and the force and moment sensor scaling factors. After this first pass, the routine begins at the top of the dynamic loop with the force and moment transformation routine. The state and state dot vectors are then calculated and integrated. The vehicle relative motion data contained in the common block RELATIVE is generated and returned to the host simulation.

The subroutines CHASE and TARGET are located in the files chase.f and target.f. These routines calculate the state dot vectors for both vehicles. The first three elements of the state dot vector are components of the time derivative of the angular momentum vector. This vector is generated using equation (14) with the chase vehicle having additional torques due to the control system. The next four elements of the state dot vector are the result of the quaternion differential equations in (31) and (32).

$$\dot{\underset{\sim}{q}} = \frac{1}{2}\left(q_4 \underset{\sim}{\omega} + \underset{\sim}{q} \times \underset{\sim}{\omega}\right) \tag{31}$$

$$\dot{q}_4 = -\frac{1}{2}\,\underset{\sim}{\omega} \cdot \underset{\sim}{q} \tag{32}$$

$\underset{\sim}{q}$ = Vector Part of Quaternion

q4 = Scalar Part of Quaternion

$\underset{\sim}{\omega}$ = Vehicle Angular Velocity Vector

The quaternions of the state vectors relate the vehicle fixed frames to inertial space. The next three terms of the state dot vectors are the vehicle accelerations with respect to the local vertical frame. These accelerations are calculated through equations (11) through (13). The force terms for the chase vehicle contain contributions from the OMV control system as well as the contact forces. The remaining terms of the state dot vector are the vehicle center of mass velocities with respect to the local vertical frame.

The file cntrlc.f contains the subroutines CNTRLC, QL1M1, and LOGIC which comprise the NASA OMV control system. Control system stick commands are input to the routines through the array KDAT. The resultant forces and moments about the chase vehicle center of mass are output in the V2 frame. These forces and moments are labeled FJX, FJY, FJZ, MX, MY, and MZ.

Subroutine FMTRAN, located in the file fmtran.f, takes in raw data from the force and moment sensor through the integer array KIN. This data is then scaled and filtered to produce the contact forces and moments acting on the target vehicle at the sensor location. From these values, the forces and moments acting at the target center of mass are calculated using equations (16) and (18), and those acting at the chase vehicle center of mass through (17) and (19).

File intfac.f contains the subroutine INTFAC. This subroutine generates the relative vehicle position, velocity, angular velocity, and orientation. This subroutine also calculates other transformation matrices used throughout the model. The relative vehicle position and velocity, RD2D1D1 and RD2D1D1D, are generated through equations (20) and (28). The relative angular velocity of the chase vehicle with respect to the target vehicle, OMV2V1V2, is computed using equation (29) with the resulting vector in the V2 frame.

The file libjg.f consists of various math library subroutines. Many of these routines handle routine matrix vector operations. There is a matrix inverse routine for 3 x 3 matrices and also routines which generate direction cosine matrices from quaternions and vice versa.

Subroutine STARTT of the file start.f defines the vehicles' mass properties and geometry. The orbital parameters and initial transformation matrices are also input. The effects of gravity can be nullified by setting the variable GMEK to zero. This variable is the product of the universal gravitational constant and the mass of the earth. This will fix the local vertical frame in space. The control system parameters are also set at the end of this subroutine.

6.0 Simulation Verification

Before integration of the math model with the host simulation, the dynamic model must be verified. The entire contact dynamics simulation must be validated upon integration before any confidence can be placed in the hardware tests.

The math model was checked out by two means. First, simple forces and moments were applied to the vehicles in each axis with the results verified by hand. The OMV control system was exercised by commanding translational accelerations and angular rates for the different axes and tracking the response of the chaser. More complex runs were then made to verify Hill's equations, orbital effects, and the Newton - Euler equations. These runs were verified through other contact dynamics simulations at Control Dynamics.

The math model has been successfully transferred to the VAX computer and again checked out. The necessary changes have been made to interface it with the host simulation. The integrated simulation can now be validated by using simple springs as the docking mechanism. After the compensation tests are completed, real time results can be generated for various initial condition runs.

An analytic model of the spring system will be used to duplicate the real time results. For the case presented here, a linear translational spring is attached to the chase vehicle shown in Figure 5. The unit vector along the 1 axis of D2, $\underset{\sim}{U1}$, coincides with the spring. $\underset{\sim}{Rrel}$ is the position vector of the D1 frame with respect to D2. Contact between the target vehicle and the spring is checked by comparing the equilibrium spring length to the component of $\underset{\sim}{Rrel}$ along $\underset{\sim}{U1}$. If contact is made, equal and opposite forces proportional to the spring compression are applied to the vehicles.

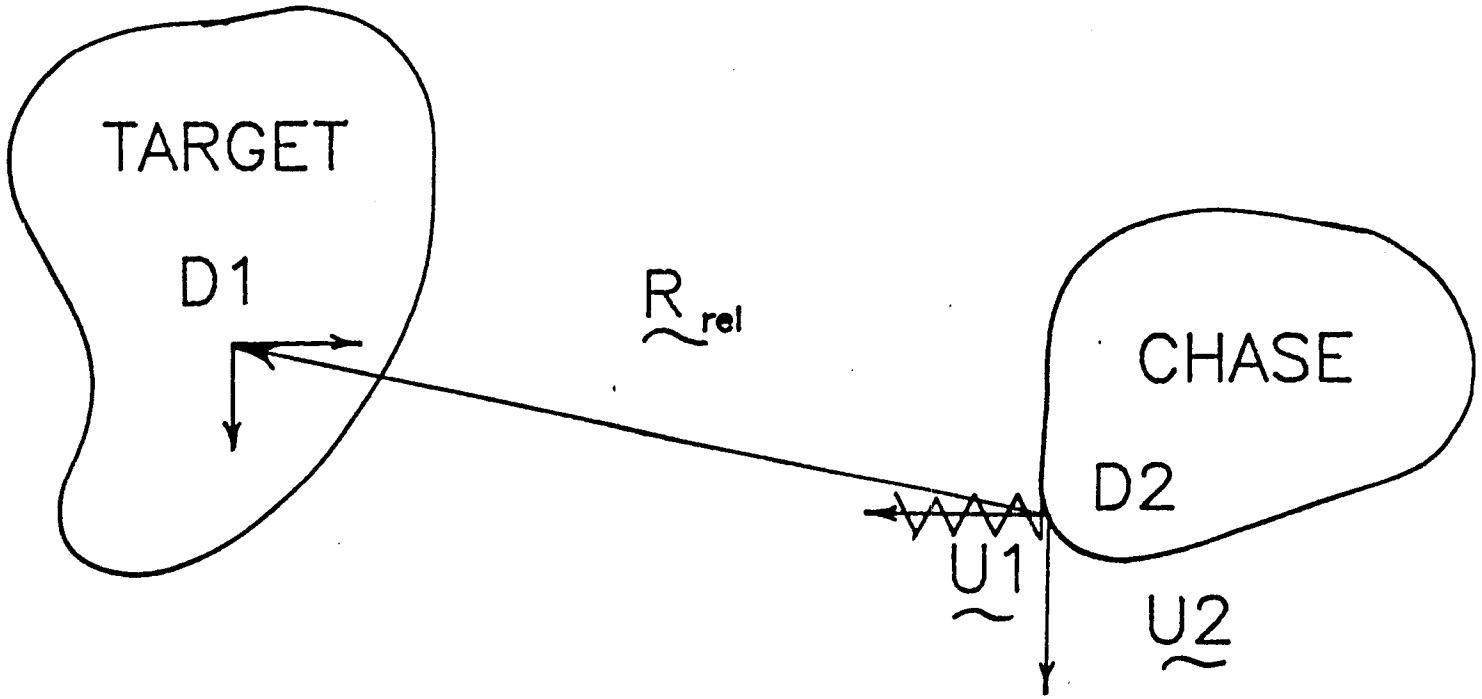

TARGET
D1
$\underset{\sim}{R}_{rel}$
CHASE
D2
$\underset{\sim}{U1}$
$\underset{\sim}{U2}$

Figure 5

The equations governing these contact forces are (33) and (34).

$$(\underset{\sim}{R}_{rel} \cdot \underset{\sim}{U1}) > Leq \;, \quad \underset{\sim}{F}_c = 0 \tag{33}$$

$$(\underset{\sim}{R}_{rel} \cdot \underset{\sim}{U1}) < Leq \;, \underset{\sim}{F}_c = -Ks(Leq - \underset{\sim}{R}_{rel} \cdot \underset{\sim}{U1})\underset{\sim}{U1} \tag{34}$$

Leq = Spring Equilibrium Length

Ks = Spring Constant

$\underset{\sim}{F}_c$ = Contact Force Acting on Chase Vehicle

Figure 6 depicts the run scenario used to generate the results of Figures 7 and 8. The target vehicle is fixed in space with the chase vehicle approaching it at a rate of .5 inches per second. The effects of gravity have been neglected; therefore, the local vertical frame is fixed in space. The chase vehicle has a mass of 326.23 slugs, and the spring constant is 50 pounds per inch. The simulation cycle time or integration step size is 35 milliseconds. Since damping has been neglected in the model, the vehicle exit velocity should be .5 inches per second. Figure 7 shows the vehicle veolcity versus time. When the velocity is zero, the spring compression, and force, is a maximum. This is seen in Figure 8. The velocity is presented in the local vertical frame, while the forces are plotted in the S1 or sensor frame. These results would compare favorably with the real time results for a perfect system with no time lag or numerical problems.

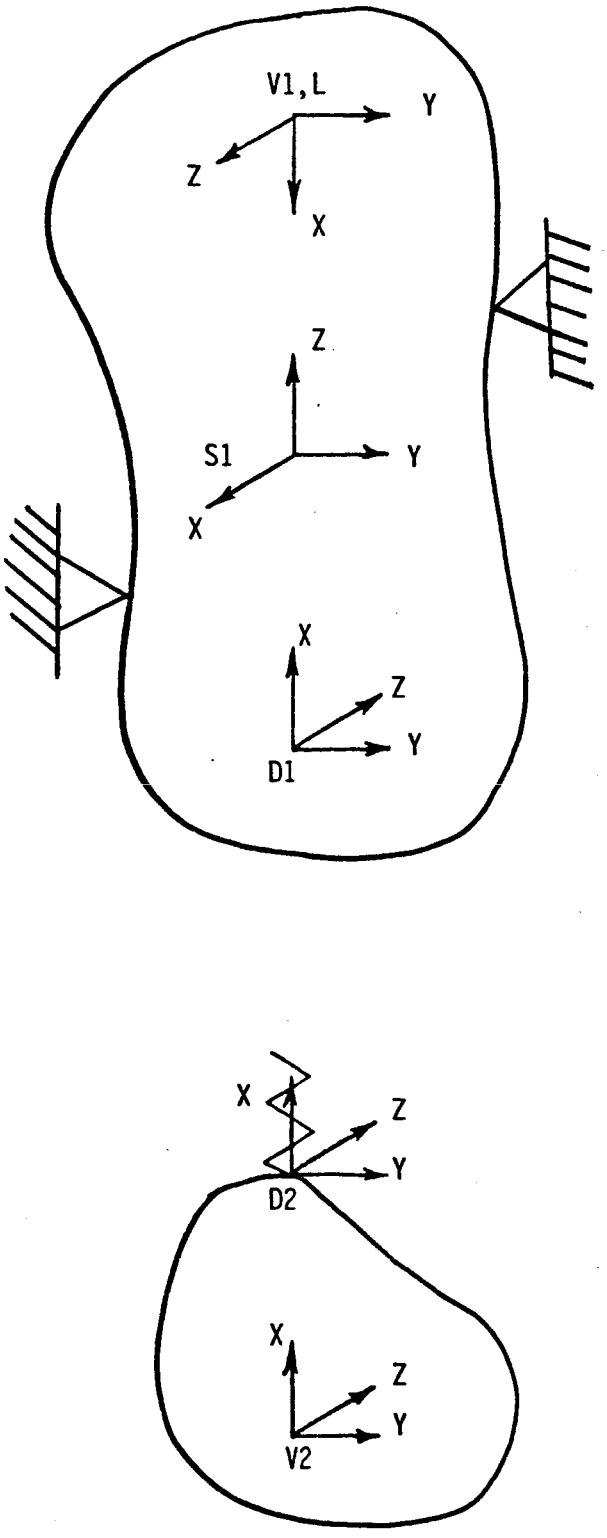


Figure 6

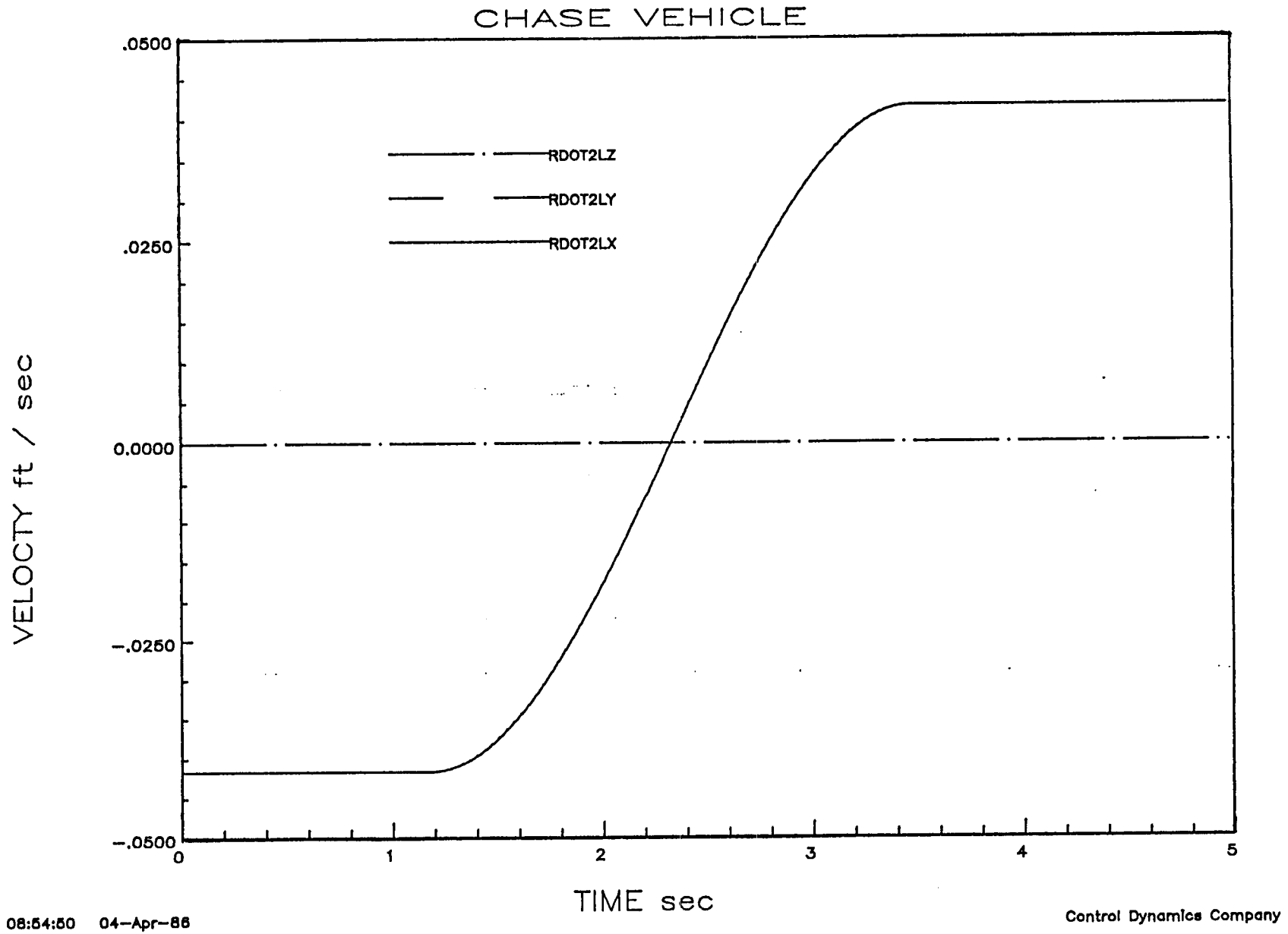

CHASE VEHICLE
.0500
.0250
0.0000
-.0250
-.0500
RDOT2LZ
RDOT2LY
RDOT2LX
VELOCTY ft / sec
0
1
2
3
4
5
TIME sec
08:54:50 04-Apr-86
Control Dynamics Company

Figure 7

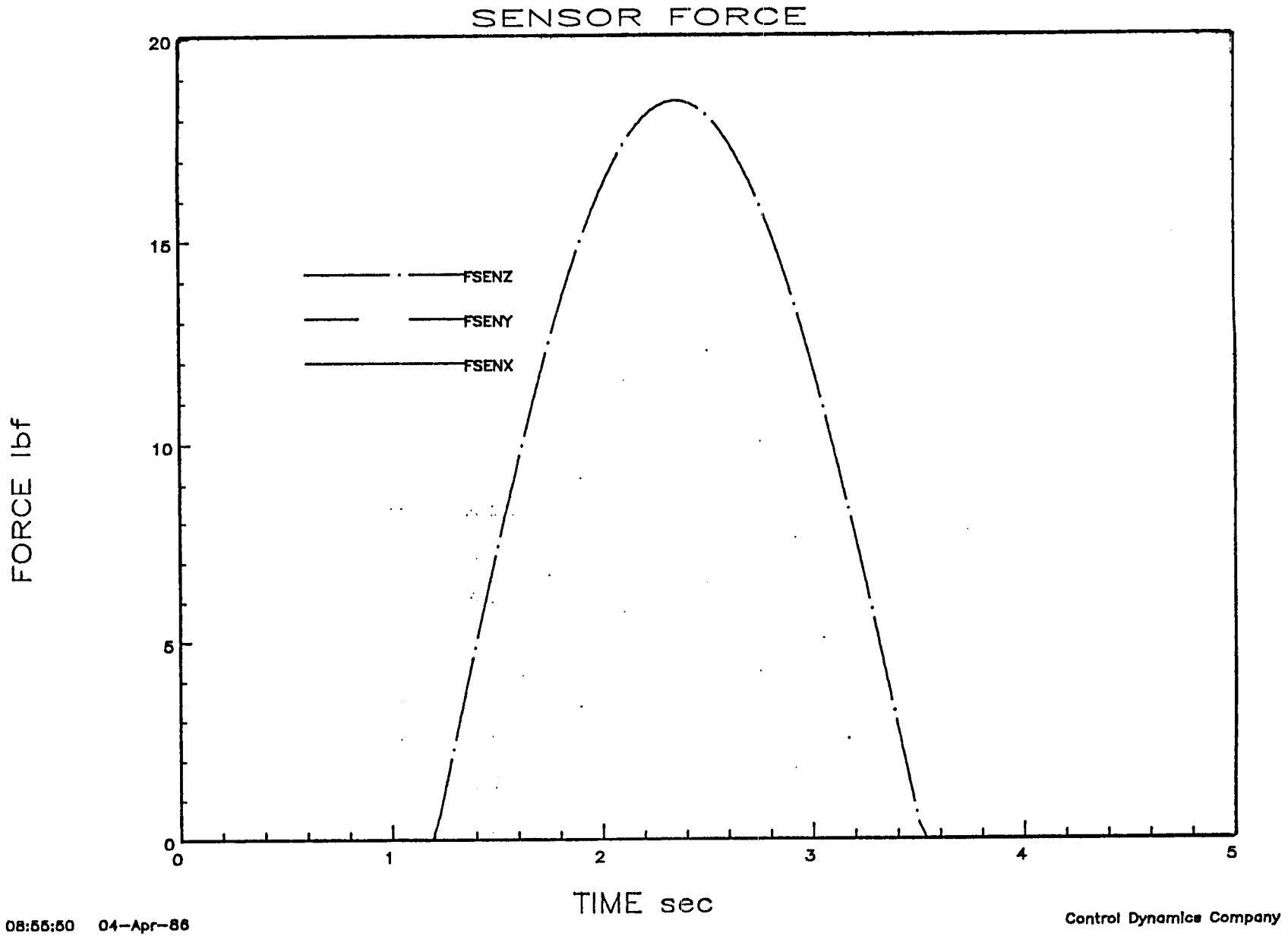

SENSOR FORCE
20
15
10
5
0
FSENZ
FSENY
FSENX
FORCE lbf
0
1
2
3
4
5
TIME sec
08:55:50 04-Apr-86
Control Dynamics Company

Figure 8

7.0 RMS End Effects Math Model

In section 6, an analytical model of the translational spring was used to generate contact forces in place of the actual hardware. These forces were fed into the dynamic math model in place of those measured by the sensor in order to generate the vehicles' responses. This section describes the analytic model of the RMS end effector based on the method of soft constraints. To apply this method, a set of constraint equations describing the end effector are derived. Small violations of these constraints are permitted, but forces proportional to the violation are applied normal to the constraining surfaces.

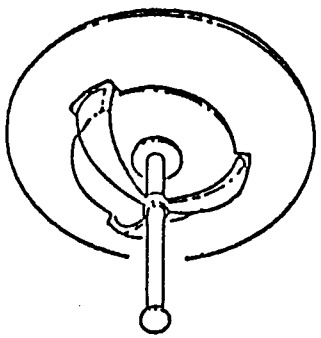
There are three possible points of contact, shown in Figure 9, for which constraint equations are derived. First, there is probe to snare cable contact. Second, the shoulders at the base of the probe can hit the grooves in the collar of the end effector. And third, the shoulders may strike the flat of the collar and miss the grooves. It is assumed that there is no contact between the probe and the inner walls of the end effector. Damping is also neglected.

Figure 10 depicts two orbiting vehicles with the relative position between the two docking ports denoted by $\underset{\sim}{r_r}$. The D1 coordinate frame of the target vehicle is fixed to the tip of the probe or grapple fixture. The D2 frame is at the center of the top of the end effector.

Figure 11 shows the grapple fixture and the snare cables in an arbitrary position. With the grapple fixture inside of the end effector and snared by the cables, then the constraint equations governing the probe position are (35) and (36).

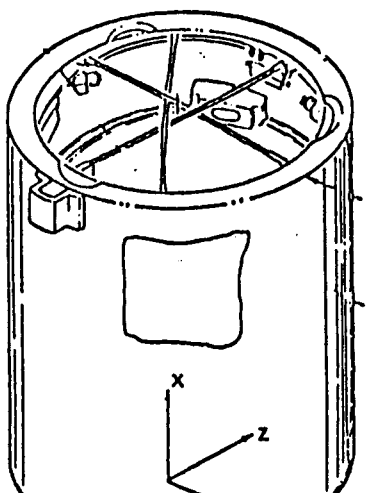
$$\underset{\sim}{d} \geq 0 \qquad (35)$$

POSSIBLE POINTS OF CONTACT



- PROBE TO CABLE

- SHOULDER TO GROOVE



- SHOULDER TO COLLAR

Figure 9

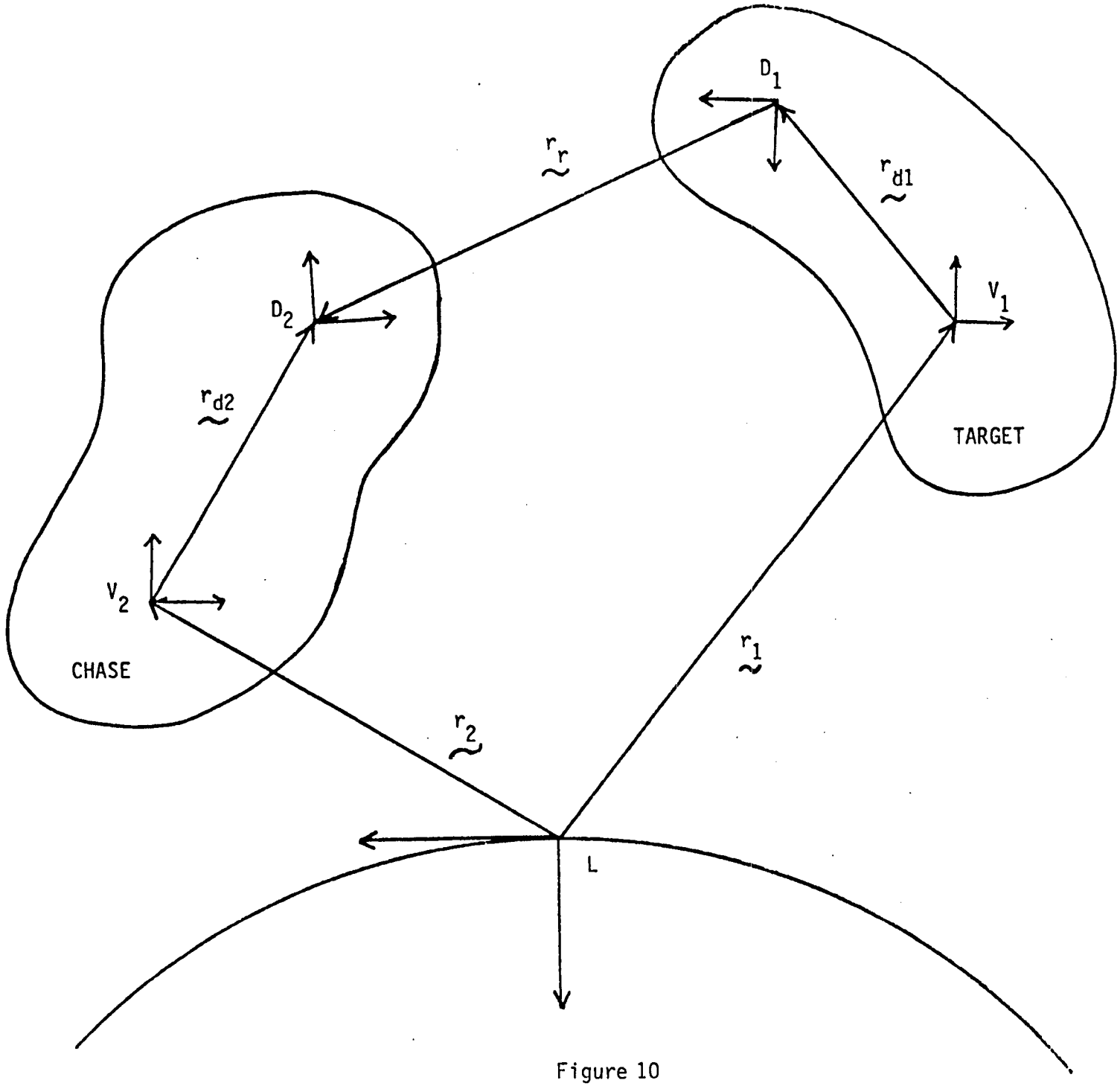


Figure 10

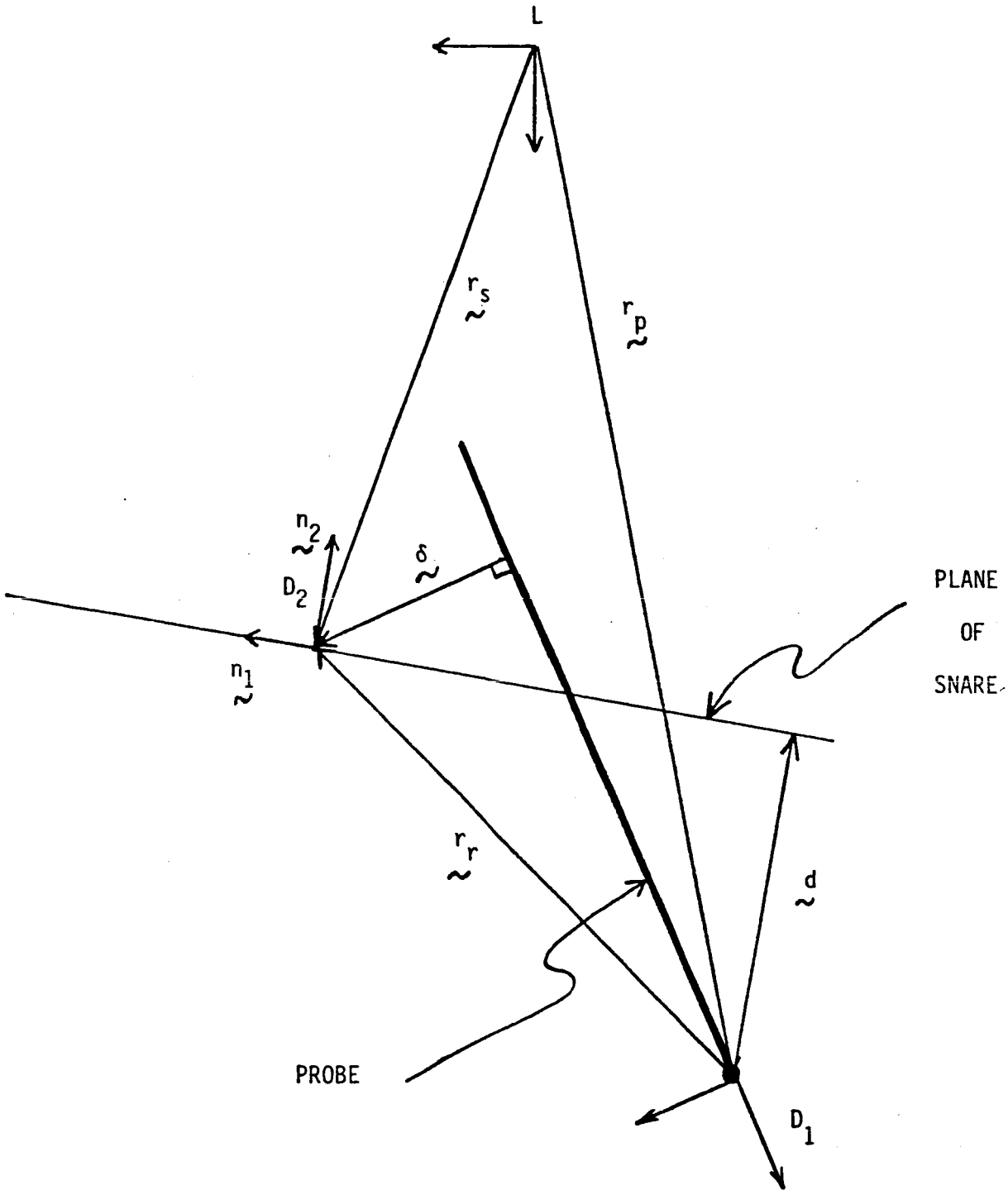


Figure 11

$$\underset{\sim}{\delta} = 0 \tag{36}$$

Equation (35) simply states that once the probe is snared, it may not leave the cylinder of the end effector. Equation (36) implies that the probe should remain in the center of the triangle formed by the three snare cables. If these equations are violated, then the forces of equations (37) and (38) are applied to the probe.

$$\underset{\sim}{F} = K \underset{\sim}{d} \tag{37}$$

$$\underset{\sim}{F} = K \underset{\sim}{\delta} \tag{38}$$

K = Snare Cable Stiffness

Figure 12 depicts the relative positions of the shoulders of the grapple fixture and the collar of the end effector. $\underset{\sim}{S}$ is the position vector from the tip of the probe to the outer edge of a shoulder. $\underset{\sim}{G}$ is the position of the center of a groove with respect to the D2 origin. The width of each groove is W. $\underset{\sim}{\varepsilon}$ is the position vector from a given groove to a given shoulder. $\underset{\sim}{\varepsilon}$ is calculated from equation (39) for each shoulder and groove combination.

$$\underset{\sim}{\varepsilon} = \underset{\sim}{S} - \underset{\sim}{r_r} - \underset{\sim}{G} \tag{39}$$

The constraint equation governing the shoulder position is given by equation (40)

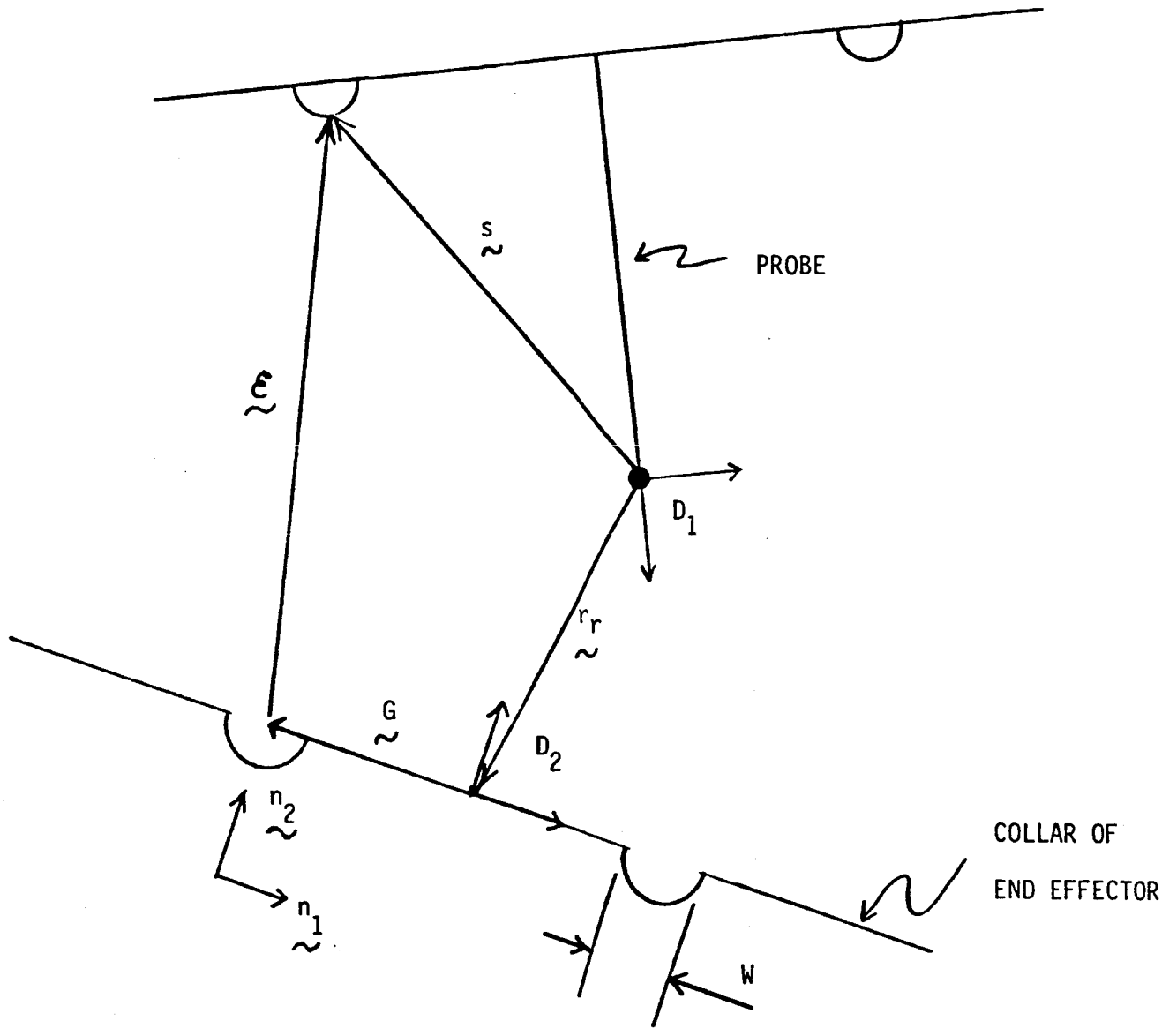


Figure 12

$$\underset{\sim}{\varepsilon} \cdot \underset{\sim}{n_2} \geq 0 \tag{40}$$

This equation simply states that the shoulder may not pass through the collar. However, if equation (40) is violated, then there is either shoulder to groove or shoulder to collar contact. If there is shoulder to groove contact, then the component of $\underset{\sim}{\varepsilon}$ along $\underset{\sim}{n_1}$ will be less than half the groove width. For shoulder to groove contact, the force of equation (41) is applied to the shoulder.

$$\left|\underset{\sim}{\varepsilon} \cdot \underset{\sim}{n_1}\right| \leq w/2 \quad , \quad \underset{\sim}{F_s} = -K\underset{\sim}{\varepsilon} \tag{41}$$

K = Collar Material Stiffness

The collar stiffness is generally of the order of ten thousand pounds per foot. If the component of $\underset{\sim}{\varepsilon}$ along $\underset{\sim}{n_1}$ is larger than the groove half width, then there is shoulder to collar contact. The shoulder force is given by equation (42).

$$\left|\underset{\sim}{\varepsilon} \cdot \underset{\sim}{n_1}\right| > w/2 \quad , \quad \underset{\sim}{F_s} = K\left|\underset{\sim}{\varepsilon} \cdot \underset{\sim}{n_2}\right| \underset{\sim}{n_2} \tag{42}$$

This force is applied to the shoulder in the direction normal to the collar.

The analytic contact force model of the RMS end effector has been implemented into the simulation and exercised. In order to use this model to verify real time results of tests using the end effector, the carriage motor speed and material stiffness should be investigated to learn more accurate values.

8.0 Conclusions and Recommendations

In fulfillment of the contract objective, a generalized math model has been developed to represent the relative motion between two rigid six degree of freedom orbiting vehicles. The model has no restrictions on vehicle size and is modular in form. The NASA OMV control system has been integrated into the model. The dynamics math model is coded in FORTRAN 77 and is currently working on the VAX 11-750 of the docking facility. The interface has been defined to merge the math model with the host simulation and verify with actual real time tests.

In the future, the math model will be examined in an effort to reduce the simulation cycle time. The math model will be modified to streamline the leg length calculations. Control Dynamics will also support the three point contact study through structural analysis and continued improvement of the math model.

APPENDIX 1

VARIABLE DEFINITIONS

AE - Transformation from E Frame to Orbit Plane

AIV1 - Inertia of Target Vehicle about Center of Mass in V1 frame

AIV1I - Inverse of AIV1

AIV2 - Inertia of Chase Vehicle about Center of Mass in V2 Coordinates

AIV2I - Inverse of AIV2

DT - Cycle Time

DT2 - Half Cycle Time

D1D2 - Transformation from D2 to D1 Coordinates

D1L - Transformation from L to D1 Coordinates

D1V1 - Transformation from V1 to D1 Coordinates

D2L - Transformation from L to D2 Coordinates

D2V2 - Transformation from V2 to D2 Coordinates

F1CL - Contact Force Acting on Target Vehicle in L Coordinates

F1CS1 - Contact Force from Sensor in S1 Coordinates

F2CL - Contact Force Acting on Chase Vehicle in L Coordinates

F2CS1 - Contact Force Acting on Chase Vehicle in S1 Coordinates

FJX, FJY, FJZ - OMV Control System Forces in V2 Frame

HTV1 - Angular Momentum of Target Vehicle

HTV2 - Angular Momentum of Chase Vehicle

IFLAG - Flag Used in Sensor Offset Forces

IFREEZ - Freeze Flag Set by KIN

JPASS - Counter of Dynamic Loop Passes

JPOT - Flag Set by KIN

KDAT - Array of Joystick Commands for OMV Control System

KIN - Array of Force and Moment Sensor Values and Flags

MASS1 - Mass of Target Vehicle

MASS2 - Mass of Chase Vehicle

MX, MY, MZ - OMV Control System Torques about Chase Vehicle Center of Mass in V2 Coordinates

NNN - Initialization Variable

NST - Dimension of State Vector

OMO - Angular Velocity of L Frame

OMOSQ - OMO Squared

OMV1 - Angular Velocity of Target Vehicle in V1 Coordinates

OMV2 - Angular Velocity of Chase Vehicle in V2 Coordinates

OMV2V1V2 - Angular Velocity of Chaser with Respect to Target in V2 Coordinates

QV1E - Target Vehicle Quaternions Relating V1 and E Frames

QV2E - Chase Vehicle Quaternions Relating V2 and E Frames

RD1V1V1 - Position Vector of Target Docking Port with Respect to V1 in V1 Coordinates

RD2D1D1 - Position Vector of Chaser Docking Port with Respect to Target Docking Port in D1 coordinates

RD2D1D1D - Time Derivative of RD2D1D1

RD2V2V2 - Position Vector of Chase Docking Port with Respect to V2 in V2 Coordinates

RNU2 - Current Orbit Angle

RNU2O - Initial Orbit Angle

RS1V1V1 - Position Vector of Sensor with Respect to V1 in V1 Coordinates

RS1V1S1 - Position Vector of Sensor with Respect to V1 in S1 Coordinates

RS1V2S1 - Position Vector of Sensor with Respect to V2 in S1 Coordinates

R1L - Position Vector Target Vehicle with Respect to L Frame

R1LD - Velocity Vector of Target Vehicle with Respect to L Frame

R2L	-	Position Vector of Chase Vehicle with Respect to L Frame
R2LD	-	Velocity Vector of Chase Vehicle with Respect to L Frame
S1L	-	Transformation from L to S1 Coordinates
S1V1	-	Transformation from V1 to S1 Coordinates
S1V2	-	Transformation from V2 to S1 Coordinates
T	-	Time
TD	-	OMV Control System Time Delay
T1CGV1	-	Contact Torque Acting at Target Center of Mass in V1 Coordinates
T1CGS1	-	Contact Torque Acting at Target Center of Mass in S1 Coordinates
T1CS1	-	Contact Torque at Sensor in S1 Coordinates
T1G	-	Target Vehicle Gravity Gradient Torque in V1 Coordinates
T2CGS1	-	Contact Torque Acting about Chase Vehicle Center of Mass in S1 Coordinates
T2CGV2	-	Contact Torque Acting about Chase Vehicle Center of Mass in V2 Coordinates
T2G	-	Gravity Gradient Torque Acting on Chase Vehicle in V2 Coordinates
V1L	-	Transformation from L to V1 Coordinates
V1E	-	Transformation from E to V1 Coordinates
V2E	-	Transformation from E to V2 Coordinates
V2L	-	Transformation from L to V2 Coordinates
XLE	-	Transformation from E to L coordinates
Y1	-	Target Vehicle State Vehicle
YD1	-	Target Vehicle State DOT Vector
YDO1	-	Previous Value of YD1
Y2	-	Chase Vehicle State Vector
YD2	-	Chase Vehicle State Dot Vector
YDO2	-	Previous Value of YD2

APPENDIX 2

PROGRAM LISTING

```
C
C       THIS FILE CONTAINS THE NECESSARY COMMON BLOCKS TO INTERFACE
C       THE NASA DOCKING SIMULATION WITH THE CONTROL DYNAMICS DOCKING
C       DYNAMICS SUBROUTINE
C
C
        INTEGER*2 KIN(10),KDAT(7)
        INTEGER IFREEZ,NOUSE,IPOT,IFLAG,JPASS,NNN
C
        REAL T,DT,DT2,TD
        REAL RD2D1D1(3),RD2D1D1D(3),OMV2V1V2(3),D1D2(3,3)
C
C
        COMMON /PASS/ JPASS
        COMMON /INT1/ NNN
        COMMON /CADIN/ KDAT
        COMMON /SENSOR/ KIN,NOUSE,IFREEZ,IFLAG,IPOT
        COMMON /TIME/ T,DT,DT2,TD
        COMMON /RELATIVE/ RD2D1D1,RD2D1D1D,OMV2V1V2,D1D2
C
C
C       T IS TIME
C       DT IS THE CYCLE TIME
C       DT2 IS HALF OF THE CYCLE TIME
C       TD IS TIME DELAY FOR THE OMV CONTROL SYSTEM
C
C       JPASS IS A PROGRAM PASS COUNTER FOR THE OMV CONTROL SYSTEM
C
C       NNN IS INITIALIZATION VARIABLE
C
C       KDAT IS A 7 ELEMENT VECTOR FROM THE JOY STICK OF THE OMV
C       CONTROL SYSTEM.  THE FIRST 6 ELEMENTS ARE X,Y,Z TRANSLATION
C       AND ROLL, PITCH, AND YAW.  ELEMENT 7 IS FOR THE ATTITUDE RATE
C       HOLD SWITCH.
C
C       KIN IS A 10 ELEMENT VECTOR FROM THE FORCE/MOMENT SENSOR.
C       THE FIRST 6 ELEMENTS ARE X,Y,Z, AND ROLL, YAW, PITCH IN
C       SENSOR COORDINATES.  ELEMENT 8 IS USED TO RESET THE OFFSET
C       FORCES.  THE FREEZE FLAG IS SET EQUAL TO ELEMENT 7.
C       IPOT IS SET EQUAL TO ELEMENT 9.
C
C       RD2D1D1 IS THE POSITION VECTOR FROM THE TARGET DOCKING
C       PORT TO THE CHASE DOCKING PORT IN TARGET DOCKING PORT
C       COORDINATES.
C
C       RD2D1D1D IS THE VELOCITY VECTOR OF THE CHASE DOCKINIG PORT
C       WITH RESPECT TO THE TARGET DOCKING PORT IN TARGET DOCKING
C       PORT COORDINATES.
C
C       OMV2V1V2 IS THE ANGULAR VELOCITY VECTOR OF THE CHASE VEHICLE
C       WITH RESPECT TO THE TARGET VEHICLE IN CHASE VEHCILE COORDINATES
C
```

```
C       D1D2 IS THE TRANSFORMATION MATRIX FROM THE CHASE DOCKING PORT
C       TO THE TARGET DOCKING PORT.
C
```

```
C
C       THIS FILE CONTAINS THE COMMON BLOCKS FOR THE ORBITAL PARAMETERS
C       OF THE EQUATIONS OF MOTION AND ALSO THE SIZE OF THE PLANT
C       STATE VECTORS.
C
C
      REAL OMOSQ,OMO,RNU2,RNU20
      REAL XLE(3,3),AE(3,3)
      INTEGER NST
C
C
      COMMON /STAT/ NST
      COMMON /EARTH/ RNU20,RNU2,OMOSQ,OMO
      COMMON /TRANS/ AE,XLE
```

```
C
C       THIS FILE CONTAINS THE COMMON BLOCKS AND DIMENSIONS
C       FOR THE DATA PERTAINING TO BODY 1 OR THE TARGET VEHICLE.
C
        REAL Y1(13),YD1(13),AIV1(3,3),AIV1I(3,3)
        REAL OMV1(3),S1V1(3,3),RS1V1V1(3),R1LD(3),R1L(3)
        REAL HTV1(3),QV1E(4),MASS1,V1L(3,3),YDO1(13)
        REAL S1L(3,3),V1E(3,3),RS1V1S1(3)
        REAL F1CS1(3),T1CGS1(3),T1G(3),T1CS1(3)
        REAL T1CGV1(3),F1CL(3)
        REAL RD1V1V1(3),D1V1(3,3),D1L(3,3)
C
C
C
        EQUIVALENCE (HTV1(1),Y1(1)),
     1              (QV1E(1),Y1(4)),
     2              (R1LD(1),Y1(8)),
     3              (R1L(1),Y1(11))
C
C
C
        COMMON /STATE1/ Y1,YD1,YDO1
        COMMON /MAS1/ MASS1,AIV1,AIV1I,RS1V1V1,RS1V1S1,RD1V1V1
        COMMON /BODY1/ OMV1,T1G
        COMMON /TRANS1/ V1L,S1V1,V1E,S1L,D1V1,D1L
        COMMON /CONTC1/ T1CGV1,F1CL,F1CS1,T1CGS1,T1CS1
```

```
C
C       THIS FILE CONTAINS THE COMMON BLOCKS AND DIMENSIONS
C       FOR THE DATA PERTAINING TO BODY 2 OR THE CHASE VEHICLE.
C
        REAL Y2(13),YD2(13),AIV2(3,3),AIV2I(3,3)
        REAL OMV2(3),D2V2(3,3),RD2V2V2(3),R2LD(3),R2L(3)
        REAL HTV2(3),QV2E(4),MASS2,V2L(3,3),YDO2(13)
        REAL V2E(3,3),RS1V2S1(3)
        REAL F2CS1(3),T2CGS1(3),T2G(3)
        REAL T2CGV2(3),F2CL(3),S1V2(3,3),D2L(3,3)
C
C
C
        EQUIVALENCE (HTV2(1),Y2(1)),
     1              (QV2E(1),Y2(4)),
     2              (R2LD(1),Y2(8)),
     3              (R2L(1),Y2(11))
C
C
C
        COMMON /STATE2/ Y2,YD2,YDO2
        COMMON /MAS2/ MASS2,AIV2,AIV2I,RD2V2V2,RS1V2S1
        COMMON /BODY2/ OMV2,T2G
        COMMON /TRANS2/ V2L,D2V2,V2E,S1V2,D2L
        COMMON /CONTC2/ F2CS1,T2CGS1,T2CGV2,F2CL
```

```
C
C       THIS FILE CONTAINS THE NECESSARY COMMON BLOCKS FOR THE NASA
C       OMV CONTROL SYSTEM.
C
        INTEGER*4 EAX,EAY,EAZ,EMX,EMY,EMZ,NAD,NDCI,NDC,IAUTOP,ISTBSL
        INTEGER*4 MPASS,ISAMP,IJONS(3),IJOFS(3),IS
        REAL SSIN(7),YIN,RTD,DTR,SCX,SCY,SCZ,AFUEL
        REAL P31,P32,P33,P34,PD31,PD32,PD33,PD34,P31M1,P32M1,P33M1
        REAL P34M1,PD31M1,PD32M1,PD33M1,PD34M1,DELAY(80,6)
        REAL WVX2A,WVY2A,WVZ2A,NOISE1,NOISE2,NOISQ
        REAL MX,MY,MZ,FJX,FJY,FJZ
C
C
        COMMON /ADIN/ SSIN,YIN,NDC,NAD,NDCI
        COMMON /DEG/ RTD,DTR
        COMMON /MSB/ ISTBSL
        COMMON /SAMP/ MPASS,ISAMP
        COMMON /LOG/ EAX,EAY,EAZ,EMX,EMY,EMZ,IJONS,IJOFS
        COMMON /MOMV/ AFUEL,WVX2A,WVY2A,WVZ2A
        COMMON /CONFOR/ MX,MY,MZ,FJX,FJY,FJZ
        COMMON /SCALE/ SCX,SCY,SCZ
        COMMON /NOIS/ NOISE1,NOISE2,NOISQ
        COMMON /LAG/ DELAY,IS
        COMMON /ERROR1/ P31,P32,P33,P34,P31M1,P32M1,P33M1,P34M1
        COMMON /ERROR2/ PD31,PD32,PD33,PD34,PD31M1,PD32M1,PD33M1,PD34M1
```

```
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C       PHYSICAL SYSTEM (PLANT) SUBROUTINE FOR CHASE VEHICLE
C
C       THE CHASE VEHICLE IS DENOTED BY THE NUMBER 2.
C       THIS SUBROUTINE CALCULATES THE TIME DERIVATIVE OF THE
C       CHASE VEHICLE STATE VECTOR.  IT USES NEWTON-EULER AND
C       HILL'S EQUATIONS.
C
C       THE INPUT TORQUES ARE FROM THE OMV CONTROL SYSTEM, VEHICLE
C       CONTACT, AND GRAVITY GRADIENT.
C
C       THE INPUT FORCES ARE FROM THE CONTROL SYSTEM AND VEHICLE
C       CONTACT
C
C-----------------------------------------------------------------------
      SUBROUTINE CHASE
C-----------------------------------------------------------------------
        INCLUDE 'cmmn.f'
        INCLUDE 'cmmn2.f'
        INCLUDE 'cmmncon.f'
C-----------------------------------------------------------------------
     .  REAL FXL,FYL,FZL,VTEM(3)
        INTEGER I,J
C-----------------------------------------------------------------------
C       EXTRACT THE PHYSICAL PARAMETERS OUT OF THE STATE VECTOR Y.
C-----------------------------------------------------------------------
C       ASSUME THAT THE STATE VECTOR IS CONSTRUCTED AS FOLLOWS:
C          Y=(HTV2,QV2E,R2LD,R2L)
C
C       HTV2 IS THE ANGULAR MOMENTUM VECTOR OF THE CHASE VEHICLE.
C       QV2E IS THE QUATERNION FOR THE TRANSFORMATION FROM THE
C             INERTIAL, E, FRAME TO THE CHASE VEHICLE FIXED FRAME.
C       R2LD IS THE VELOCITY OF THE CHASE VEHICLE CENTER OF MASS
C             WITH RESPECT TO THE L FRAME AND IN L FRAME COORDINATES.
C       R2L  IS THE POSITON OF THE CHASE VEHICLE CENTER OF MASS
C             WITH RESPECT TO THE L FRAME AND IN L FRAME COORDINATES.
C
C-----------------------------------------------------------------------
      DO 100 I=1,NST
        YDO2(I)=YD2(I)
100   CONTINUE
C-----------------------------------------------------------------------
C       COMPUTE THE TOTAL GRAVITY GRADIENT TORQUES
C-----------------------------------------------------------------------
      DO 1500 I=1,3
      VTEM(I)=0.0
      DO 1500 J=1,3
      VTEM(I)=VTEM(I)+AIV2(I,J)*V2L(J,3)
1500  CONTINUE
      T2G(1)=3.0*OMOSQ*(V2L(2,3)*VTEM(3)-V2L(3,3)*VTEM(2))
      T2G(2)=3.0*OMOSQ*(V2L(3,3)*VTEM(1)-V2L(1,3)*VTEM(3))
```

```
      T2G(3)=3.0*OMOSQ*(V2L(1,3)*VTEM(2)-V2L(2,3)*VTEM(1))
C-----------------------------------------------------------------------
C
C     CALCULATE THE DERIVATIVE OF THE ANGULAR MOMENTUM VECTOR
C     USING NEWTON-EULER EQUATION.
C
C-----------------------------------------------------------------------
      YD2(1)=MX+T2G(1)+T2CGV2(1)-OMV2(2)*HTV2(3)+OMV2(3)*HTV2(2)
      YD2(2)=MY+T2G(2)+T2CGV2(2)-OMV2(3)*HTV2(1)+OMV2(1)*HTV2(3)
      YD2(3)=MZ+T2G(3)+T2CGV2(3)-OMV2(1)*HTV2(2)+OMV2(2)*HTV2(1)
C-----------------------------------------------------------------------
C
C     CALCULATE THE DERIVATIVE OF THE QUATERNION
C
C-----------------------------------------------------------------------
      YD2(4)=.5*(OMV2(3)*QV2E(2)
     $       -OMV2(2)*QV2E(3)+OMV2(1)*QV2E(4))
      YD2(5)=.5*(-OMV2(3)*QV2E(1)
     $       +OMV2(1)*QV2E(3)+OMV2(2)*QV2E(4))
      YD2(6)=.5*(OMV2(2)*QV2E(1)
     $       -OMV2(1)*QV2E(2)+OMV2(3)*QV2E(4))
      YD2(7)=-.5*(OMV2(1)*QV2E(1)
     $       +OMV2(2)*QV2E(2)+OMV2(3)*QV2E(3))
C-----------------------------------------------------------------------
C
C     TRANSFORM FORCES OF CONTROL SYSTEM FROM V2 FRAME TO L FRAME
C
C-----------------------------------------------------------------------
        FXL=FJX*V2L(1,1)+FJY*V2L(2,1)+FJZ*V2L(3,1)
        FYL=FJX*V2L(1,2)+FJY*V2L(2,2)+FJZ*V2L(3,2)
        FZL=FJX*V2L(1,3)+FJY*V2L(2,3)+FJZ*V2L(3,3)
C-----------------------------------------------------------------------
C
C     CALCULATE THE ACCELERATION OF THE VEHICLE CENTER OF MASS
C     WITH RESPECT TO THE L FRAME IN THE L FRAME COORDINATES.
C
C-----------------------------------------------------------------------
      YD2(8)=(FXL+F2CL(1))/MASS2-2.*OMO*R2LD(3)
      YD2(9)=(FYL+F2CL(2))/MASS2-OMOSQ*R2L(2)
      YD2(10)=(FZL+F2CL(3))/MASS2+3.*OMOSQ*R2L(3)+2.*OMO*R2LD(1)
      YD2(11)=Y2(8)
      YD2(12)=Y2(9)
      YD2(13)=Y2(10)
C-----------------------------------------------------------------------
      RETURN
      END
```

```
C
C       SUBROUTINE CNTRLC IS THE NASA RCS THRUSTER CONTROL SYSTEM FOR
C       THE OMV
C
C       THE INPUT TO THIS ROUTINE REQUIRES THE CHASE VEHICLE INERTIAL
C       ANGULAR RATES IN THE BODY FIXED FRAME AND THE ANGULAR VELOCITY
C       OF THE LOCAL VERTICAL FRAME.
C       IT ALSO REQUIRES THE INPUT FROM THE JOY STICK IN THE KDAT
C       VECTOR.
C       THE ROUTINE RETURNS THE CONTROL FORCES AND TORQUES IN THE BODY
C       FIXED FRAME.
C
      SUBROUTINE CNTRLC
C
        INCLUDE 'cmmn.f'
        INCLUDE 'cmmncon.f'
        INCLUDE 'cmmn2.f'
        INCLUDE 'cmmnbb.f'
C
        INTEGER*4 IU,I
        REAL XIN(7),XDAT(7),PHIE2X,PHIE2Y,PHIE2Z,EX,EY,EZ
        REAL PD31M2,PD32M2,PD33M2,PD34M2,DW13,DW23,DW33
        REAL SW13,SW23,SW33,P3NORM,ROLIN,PITIN,YAWIN
        REAL ROLL,PITCH,YAW,RSQ,YSQ,PSQ,RSQ1,PSQ1,YSQ1
        REAL RSQ2,YSQ2,PSQ2,RSQ3,PSQ3,YSQ3,EAXIN,EAYIN,EAZIN
        REAL EAXA,EAYA,EAZA,WVX,WVY,WVZ,WCVX2,WCVY2,WCVZ2
        REAL WVX2,WVY2,WVZ2
        REAL QLIM1
        LOGICAL SW,COMR,COMY,COMP,COM
C
C
C       SET INERTIAL CHASE VEHICLE ANGULAR RATES FOR CONTROL SYSTEM
C
      WVX2A=OMV2(1)
      WVY2A=OMV2(2)
      WVZ2A=OMV2(3)
      WVX2=WVX2A-OMO*V2L(1,2)
      WVY2=WVY2A-OMO*V2L(2,2)
      WVZ2=WVZ2A-OMO*V2L(3,2)
C
C
C     KDAT IS THE ANALOG INPUT FROM THE STICK
C
      DO 1001 I=1,NAD,1
            XDAT(I)=FLOAT(KDAT(I))
            XIN(I)=XDAT(I)*YIN
 1001 CONTINUE
      ROLIN =  XIN(4)*SSIN(4)
      PITIN =  XIN(5)*SSIN(5)
      YAWIN =  XIN(6)*SSIN(6)
```

```
C
C  ***DELAY IS AN ARRAY TO DELAY XIN(I) FROM HAND CONTR.***
C
      IF(IS .GE. NDC+1)IS = 0
      IS = IS + 1

C
      IU = IS - NDC
      IF(IU .LE. 0) IU = IS + 1
C
      DELAY(IS,4) = ROLIN
      DELAY(IS,5) = PITIN
      DELAY(IS,6) = YAWIN
C
      ROLL = DELAY(IU,4)

      PITCH = DELAY(IU,5)
      YAW = DELAY(IU,6)
C
      IF(ROLL .LT. NOISE1 .AND. ROLL .GT. (-NOISE1))ROLL=0.0
      IF(PITCH .LT. NOISE1 .AND. PITCH .GT. (-NOISE1))PITCH=0.0
      IF(YAW .LT. NOISE1 .AND. YAW .GT. (-NOISE1))YAW=0.0
      IF(XIN(7)*SSIN(7) .LT. -.5)GO TO 36
      SW=.FALSE.
      GO TO 37
 36   SW=.TRUE.
 37   IF(ROLL .GE. NOISE1 .OR. ROLL .LE. -NOISE1)GO TO 30
      COMR=.FALSE.
      ROLL=0.0
      GO TO 31
 30   COMR=.TRUE.
 31   IF(PITCH .GE. NOISE1 .OR. PITCH .LE. -NOISE1)GO TO 32
      COMP=.FALSE.
      PITCH=0.0

      GO TO 33
 32   COMP=.TRUE.
 33   IF(YAW .GE. NOISE1 .OR. YAW .LE. -NOISE1)GO TO 34
      COMY=.FALSE.
      YAW=0.0
      GO TO 35
 34   COMY=.TRUE.
 35   COM=COMR .OR. COMP .OR. COMY
      RSQ=ROLL*ABS(ROLL)

      PSQ=PITCH*ABS(PITCH)
      YSQ=YAW*ABS(YAW)
      IF(SW .AND. COM)GO TO 38
      RSQ1=RSQ+RSQ2
      PSQ1=PSQ+PSQ2
      YSQ1=YSQ+YSQ2
      RSQ3=SCX*RSQ1
      PSQ3=SCY*PSQ1
```

```
      YSQ3=SCZ*YSQ1

      GO TO 39
   38 RSQ2=RSQ1
      PSQ2=PSQ1
      YSQ2=YSQ1
   39 WCVX2=RSQ3*DTR
      WCVY2=PSQ3*DTR
      WCVZ2=YSQ3*DTR
C
C       DEFINE THE ERROR ANGLES FROM THE QUATERNION P3
C
      IF(IAUTOP .EQ. 0) GO TO 4
      P31    = 0.0
      P32    = 0.0
      P33    = 0.0
      P34    = 1.0
      PD31M1 = 0.0
      PD32M1 = 0.0
      PD33M1 = 0.0
      PD34M1 = 0.0
      PD31   = 0.0
      PD32   = 0.0
      PD33   = 0.0
      PD34   = 0.0
      GO TO 15
    4 CONTINUE
      PD31M2=PD31M1
      PD32M2=PD32M1
      PD33M2=PD33M1
      PD34M2=PD34M1
      PD31M1=PD31
      PD32M1=PD32
      PD33M1=PD33
      PD34M1=PD34
      DW13=0.5*(WVX2 - WCVX2)
      SW13=0.5*(WVX2 + WCVX2)
      DW23=0.5*(WVY2 - WCVY2)
      SW23=0.5*(WVY2 + WCVY2)
      DW33=0.5*(WVZ2 - WCVZ2)
      SW33=0.5*(WVZ2 + WCVZ2)
      PD31=P34*DW13 - P33*SW23 + P32*SW33
      PD32=P33*SW13 +P34*DW23 - P31*SW33
      PD33= -P32*SW13 + P31*SW23 + P34*DW33
      PD34= -P31*DW13 - P32*DW23 - P33*DW33
      P31M1=P31
      P32M1=P32
      P33M1=P33
      P34M1=P34
      P31=P31M1 + DT2*(3.0*PD31M1 - PD31M2)
      P32=P32M1 +DT2*(3.0*PD32M1 - PD32M2)
      P33=P33M1 + DT2*(3.0*PD33M1 - PD33M2)
      P34=P34M1 + DT2*(3.0*PD34M1 - PD34M2)
```

```
      P3NORM=SQRT(P31**2+P32**2+P33**2+P34**2)
      P31    =P31/P3NORM
      P32    =P32/P3NORM
      P33    =P33/P3NORM
      P34    =P34/P3NORM
   15 CONTINUE
      PHIE2X= -2.0*RTD*P31
      PHIE2Y= -2.0*RTD*P32
      PHIE2Z= -2.0*RTD*P33
      PHIE2X=QLIM1(-5.,PHIE2X,5.)
      PHIE2Y=QLIM1(-5.,PHIE2Y,5.)
      PHIE2Z=QLIM1(-5.,PHIE2Z,5.)
C
      EAXIN  =-XIN( 1)*SSIN( 1)
      EAYIN  =-XIN( 2)*SSIN( 2)
      EAZIN  =+XIN( 3)*SSIN( 3)
C
      DELAY(IS,1) = EAXIN
      DELAY(IS,2) = EAYIN
      DELAY(IS,3) = EAZIN
C
      EAXA = DELAY(IU,1)
      EAYA = DELAY(IU,2)
      EAZA = DELAY(IU,3)
C
      WVX=RTD*WVX2A
      WVY=RTD*WVY2A
      WVZ=RTD*WVZ2A
      EX=67.5*PHIE2X-169.5*(WVX-RSQ3)
      EY=45. *PHIE2Y-110.625*(WVY-PSQ3)
      EZ=45. *PHIE2Z-110.625*(WVZ-YSQ3)
C
      MPASS=MOD(JPASS,ISAMP)
C     ACCEPT HAND CONTROLLER INPUTS ONLY EVERY ISAMP TIMES
      IF(MPASS .NE. 0) GO TO 16
      EAX=0
      EAY=0
      EAZ=0
      EMX=0
      EMY=0
      EMZ=0
      IF(EAXA .GT. NOISE2) EAX=1
      IF(EAXA .LT. -NOISE2) EAX=-1
      IF(EAYA .GT. NOISE2) EAY=1
      IF(EAYA .LT. -NOISE2) EAY=-1
      IF(EAZA .GT. NOISE2) EAZ=1
      IF(EAZA .LT. -NOISE2) EAZ=-1
      IF(IAUTOP .NE. 0) GO TO 160
      IF(EX .GE. 17.) EMX=1
      IF(EX .LE. -17.) EMX=-1
      IF(EY .GE. 22.) EMY=1
      IF(EY .LE. -22.) EMY=-1
      IF(EZ .GE. 22.) EMZ=1
```

```
      IF(EZ .LE. -22.) EMZ=-1
      GO TO 16
C     EXECUTE AT 160 IF IAUTOP .NE. 0
  160 IF(RSQ1 .GT. NOISQ)EMX=1
      IF(RSQ1 .LT. -NOISQ)EMX=-1
      IF(PSQ1 .GT. NOISQ)EMY=1
      IF(PSQ1 .LT. -NOISQ)EMY=-1
      IF(YSQ1 .GT. NOISQ)EMZ=1
      IF(YSQ1 .LT. -NOISQ)EMZ=-1
   16 CONTINUE
      CALL LOGIC
      RETURN
      END
C
      FUNCTION QLIM1(BL,V,TL)
C     THIS FUNCTION LIMITS V TO BL OR TL
C     BL IS BOTTOM LIMIT
C     TL IS TOP LIMIT
      REAL BL,TL,V,QLIM1
      QLIM1=V
      IF(V .LT. BL)QLIM1=BL
      IF(V .GT. TL)QLIM1=TL
      RETURN
      END
C
      SUBROUTINE LOGIC
C
      SAVE
C
      INCLUDE 'cmmncon.f'
      INCLUDE 'cmmnbb.f'
C
      REAL AX1,AY1,AY2,RO,FTR
      REAL AZ1,AZ2,CX,CY,CZ,FM,FTRX,RTIME,STIME,SSUMRT,SUMRT
      REAL SUMF,SUMFX,SUMRTL,WXDB,WYDB,WZDB
      INTEGER ID,IDS,IFLAG1,II,IFX,IFY,IFZ,IROTRY,IROTRZ,ISIDEA
      INTEGER ISDA,ISDB,ISIDEB,JETN,JETC,JET,JTIME,M,NCOUNT
      INTEGER NFT,NFP,I2,IJETON,IJETOF,L,I,J,K
      DIMENSION JETN(24),JETC(24)
      DIMENSION FM(24,6),JET(24),IJETON(3),IJETOF(3)
      DIMENSION ISIDEA(12),ISIDEB(12)
      DIMENSION I2(3)
      DIMENSION JTIME(24),RTIME(24),ID(24),STIME(24)
      DIMENSION SUMRT(24),SUMRTL(24),IDS(24)
      DIMENSION SSUMRT(24)
      DATA
     *IJETON/0,0,0/,
     *IJETOF/0,0,0/
      DATA ISIDEA/1,2,3,7,8,9,13,14,15,19,20,21/
      DATA ISIDEB/4,5,6,10,11,12,16,17,18,22,23,24/
      DATA ISDA,ISDB/0,0/
      DATA WXDB,WYDB,WZDB/.1,.1,.1/
      DATA NFT/0/
```

```
      DATA NFP/0/
C     CONSF = FTR/SPECIFIC IMPULSE(SET AT 220 SEC)=0.031818
      IF(NNN)18,18,30
   18 FTR=10.0
      FTRX=10.0
      AX1=16.523/12.
      AY1=28.529/12.
      AY2=35.096/12.
      AZ1=66.402/12.
      AZ2=72.97/12.
      DO 20 I=1,24
            DO 20 J=1,6
   20 FM(I,J)=0.0
      DO 431 I = 1,24
            JETN(I) = 0
            JETC(I) = 0
            SUMRT(I)=0.0
  431 CONTINUE
      I2(1) = 0
      I2(2) = 0
      I2(3) = 0
      NCOUNT = 0
C     FM(I,J) ARE COEFFICIENTS FOR FORCES AND MOMENTS
C             WHERE I IS THRUSTER NO.
C             THE SIGN IS FROM BOTH FORCE AND ARM.
C      X FORCES
      FM(1,1)=1.0
      FM(2,1)=1.0
      FM(3,1)=1.0
      FM(4,1)=1.0
C     -X FORCES
      FM(5,1)=-1.0
      FM(6,1)=-1.0
      FM(7,1)=-1.0
      FM(8,1)=-1.0
C      Y FORCES
      FM(9,2)=1.0
      FM(10,2)=1.0
      FM(11,2)=1.0
      FM(12,2)=1.0
C     -Y FORCES
      FM(13,2)=-1.0
      FM(14,2)=-1.0
      FM(15,2)=-1.0
      FM(16,2)=-1.0
C     Z FORCES
      FM(17,3)=1.0
      FM(18,3)=1.0
      FM(19,3)=1.0
      FM(20,3)=1.0
C     -Z FORCES
      FM(21,3)=-1.0
      FM(22,3)=-1.0
```

```
      FM(23,3)=-1.0
      FM(24,3)=-1.0
C      X MOMENT(ROLL)
      FM(9,4)=AZ1
      FM(10,4)=AZ1
      FM(15,4)=AZ1
      FM(16,4)=AZ1
      FM(18,4)=AY1
      FM(19,4)=AY1
      FM(22,4)=AY1
      FM(23,4)=AY1
C     -X MOMENT(-ROLL)
      FM(11,4)=-AZ1
      FM(12,4)=-AZ1
      FM(13,4)=-AZ1
      FM(14,4)=-AZ1
      FM(17,4)=-AY1
      FM(20,4)=-AY1
      FM(21,4)=-AY1
      FM(24,4)=-AY1
C      Y MOMENT(PITCH)
      FM(17,5)=AX1
      FM(18,5)=AX1
      FM(23,5)=AX1
      FM(24,5)=AX1
C     -Y MOMENT(-PITCH)
      FM(19,5)=-AX1
      FM(20,5)=-AX1
      FM(21,5)=-AX1
      FM(22,5)=-AX1
C      Z MOMENT(YAW)
      FM(10,6)=AX1
      FM(11,6)=AX1
      FM(14,6)=AX1
      FM(16,6)=AX1
C     -Z MOMENT(-YAW)
      FM(9,6)=-AX1
      FM(12,6)=-AX1
      FM(13,6)=-AX1
      FM(15,6)=-AX1
      RETURN
 30   CONTINUE
      IF(NNN .EQ. 2)GO TO 1050
      FJX = 0.
      FJY=0.0
      FJZ=0.0
      MX=0.0
      MY=0.0
      MZ=0.0
C     FTR   =ACOF*(PLBS-AFUEL)+BCOF
      DO 40 I=1,24
           JET(I)=0
 40   CONTINUE
```

```
      IF(ISTBSL.NE.0) GO TO 5000
      IF(EAX) 210,230,220
210   JET( 5) = 1
      JET( 6) = 1
      JET( 7) = 1
      JET(8) = 1
      GO TO 230
220   JET(1) = 1
      JET(2) = 1
      JET(3) = 1
      JET(4) = 1
230   CONTINUE
      IROTRY = EMX
      IROTRZ = 0
      IFLAG1 = 0
      IF(EMX) 240,300,240
240   IF(EMZ) 250,270,250
250   IF(EMY) 260,290,260
260   IF(EAY) 285,300,285
270   IF(EAY) 280,300,280
280   IF(EMY) 300,290,300
285   IFLAG1 = 1
290   IROTRZ = EMX
      IROTRY = 0
300   CONTINUE
      IF(IROTRY) 320,305,310
305   IF(IROTRZ) 340,350,330
310   JET(9) = 1
      JET(16) = 1
      JET(10) = 1
      JET(15) = 1
      GO TO 350
320   JET( 12) = 1
      JET(14) = 1
      JET(13) = 1
      JET(11) = 1
      GO TO 350
330   JET( 18) = 1
      JET( 19) = 1
      JET(22) = 1
      JET(23) = 1
      GO TO 350
340   JET( 17) = 1
      JET(20) = 1
      JET(21) = 1
      JET(24) = 1
350   CONTINUE
      IF(EMY) 370,380,360
360   JET(17) = 1
      JET(18) = 1
      JET(23) = 1
      JET(24) = 1
      GO TO 380
```

```
370  JET( 19) = 1
     JET(20) = 1
     JET(21) = 1
     JET(22) = 1
380  CONTINUE
     IF(EMZ) 400,410,390
390  JET( 10) = 1
     JET(11) = 1
     JET(14) = 1
     JET(16) = 1
     GO TO 410
400  JET( 9) = 1
     JET( 12) = 1
     JET(13) = 1
     JET(15) = 1
410  CONTINUE
     IF(EAY) 430,440,420
420  JET(9) = 1
     JET(10) = 1
     JET(11) = 1
     JET(12) = 1
     GO TO 440
430  JET( 13) = 1
     JET( 14) = 1
     JET(15) = 1
     JET(16) = 1
440  CONTINUE
     IF(EAZ) 460,470,450
450  JET( 17) = 1
     JET(18) = 1
     JET(19) = 1
     JET(20) = 1
     GO TO 470
460  JET( 21) = 1
     JET( 22) = 1
     JET(23) = 1
     JET(24) = 1
470  IF(JET(14)) 475,475,471
471  IF(JET(9)) 475,475,472
472  JET(14) = 0
     JET(9) = 0
475  IF(JET(10)) 480,480,476
476  IF(JET( 13)) 480,480,477
477  JET(10) = 0
     JET( 13) = 0
480  IF(JET(12)) 485,485,481
481  IF(JET(16)) 485,485,482
482  JET(12) = 0
     JET(16) = 0
485  IF(JET(11)) 490,490,486
486  IF(JET(15)) 490,490,487
487  JET(11) = 0
     JET(15) = 0
```

```
490   IF(JET(17)) 495,495,491
491   IF(JET(22)) 495,495,492
492   JET( 17) = 0
      JET( 22) = 0
495   IF(JET(18)) 500,500,496
496   IF(JET(21)) 500,500,497
497   JET(18) = 0
      JET(21) = 0
500   IF(JET(19)) 505,505,501
501   IF(JET(24)) 505,505,502
502   JET(19) = 0
      JET(24) = 0
505   IF(JET(20)) 510,510,506
506   IF(JET(23)) 510,510,507
507   JET(20) = 0
      JET(23) = 0
510   CONTINUE
      IF(IFLAG1) 560,560,511
511   IF(EAZ) 515,560,515
515   IF(JET(17)) 520,520,516
516   JET(24) = 1
      JET(20) = 1
      JET(18) = 1
      GO TO 560
520   IF(JET(18)) 525,525,521
521   JET(17) = 1
      JET(19) = 1
      JET(23) = 1
      GO TO 560
525   IF(JET(19)) 530,530,526
526   JET(18) = 1
      JET(20) = 1
      JET(22) = 1
      GO TO 560
530   IF(JET(20)) 535,535,531
531   JET(17) = 1
      JET(19) = 1
      JET( 21) = 1
      GO TO 560
535   IF(JET(21)) 540,540,536
536   JET(20) = 1
      JET(22) = 1
      JET(24) = 1
      GO TO 560
540   IF(JET(22))  545,545,541
541   JET(19) = 1
      JET(21) = 1
      JET(23) = 1
      GO TO 560
545   IF(JET(23)) 550,550,546
546   JET(24) = 1
      JET(22) = 1
      JET(18)= 1
```

```
      GO TO 560
  550 IF(JET(24)) 560,560,551
  551 JET(17) = 1
      JET(21) = 1
      JET(23) = 1
      GO TO 560
  560 CONTINUE
      IF(I2(1)) 600,600,565
  565 IF(IROTRY) 590,570,590
  570 IF(IROTRZ) 575,600,575
  575 IF(EAZ) 600,580,600
  580 IF(EMY) 600,581,600
  581 JET(18) = 0
      JET(20) = 0
      JET(21) = 0
      JET(23) = 0
      GO TO 600
  590 IF(EAY) 600,595,600
  595 IF(EMZ) 600,596,600
  596 JET(11) = 0
      JET(10) = 0
      JET(13) = 0
      JET(15) = 0
  600 CONTINUE
      IF(I2(2)) 630,630,605
  605 IF(EMY) 610,630,610
  610 IF(EAZ) 630,615,630
  615 IF(IROTRZ) 630,620,630
  620 JET(17) = 0
      JET(20) = 0
      JET(22) = 0
      JET(23) = 0
  630 CONTINUE
      IF(I2(3)) 700,700,635
  635 IF(EMZ) 640,700,640
  640 IF(EAY) 700,645,700
  645 IF(IROTRY) 700,650,700
  650 JET(11) = 0
      JET(12) = 0
      JET(15) = 0
      JET(16) = 0
  700 CONTINUE
      GO TO 900
C
 5000 CX = WVX2A*RTD
      CY = WVY2A*RTD
      CZ = WVZ2A*RTD
      EMX = 0
      IF(CX.GT.WXDB) EMX = -1
      IF(CX.LT.(-WXDB)) EMX = 1
      EAZ = 0
      IF(CY.GT.WYDB) EAZ = -1
      IF(CY.LT.(-WYDB)) EAZ = 1
```

```
      EAY = 0
      IF(CZ.GT.WZDB) EAY = 1
      IF(CZ.LT.(-WXDB)) EAY = -1
      IFX = 1 + EMX
      IFZ = 1 + EAZ
      IFY = 1 + EAY
      J = 1 + 9*IFX + 3*IFZ + IFY
      GO TO J,(5001,5002,5003,5004,5005,5006,5007,5008,5009,5010,
     *5011,5012,5013,5014,5015,5016,5017,5018,5019,5020,5021,5022,
     *5023,5024,5025,5026,5027)
 5001 CONTINUE
      JET(2) = 1
      JET(6) = 1
      JET(15) = 1
      JET(21) = 1
      JET(17) = 1
      JET(23) = 1
      GO TO 900
 5002 CONTINUE
      JET(6) = 1
      JET(21) = 1
      JET(2) = 1
      JET(8) = 1
      JET(17) = 1
      JET(23) = 1
      GO TO 900
 5003 CONTINUE
      JET(6) = 1
      JET(8) = 1
      JET(15) = 1
      JET(21) = 1
      JET(17) = 1
      JET(23) = 1
      GO TO 900
 5004 CONTINUE
      JET(2) = 1
      JET(23) = 1
      JET(15) = 1
      JET(21) = 1
      JET(6) = 1
      JET(12) = 1
      GO TO 900
 5005 CONTINUE
      JET(15) = 1
      JET(21) = 1
      JET(6) = 1
      JET(12) = 1
      GO TO 900
 5006 CONTINUE
      JET(8) = 1
      JET(17) = 1
      JET(15) = 1
      JET(21) = 1
```

```
      JET(6) = 1
      JET(12) = 1
      GO TO 900
 5007 CONTINUE
      JET(12) = 1
      JET(2) = 1
      JET(15) = 1
      JET(21) = 1
      JET(17) = 1
      JET(23) = 1
      GO TO 900
 5008 CONTINUE
      JET(12) = 1
      JET(15) = 1
      JET(2) = 1
      JET(8) = 1
      JET(17) = 1
      JET(23) = 1
      GO TO 900
 5009 CONTINUE
      JET(12) = 1
      JET(8) = 1
      JET(15) = 1
      JET(21) = 1
      JET(17) = 1
      JET(23) = 1
      GO TO 900
 5010 CONTINUE
      JET(9) = 1
      JET(21) = 1
      JET(2) = 1
      JET(20) = 1
      GO TO 900
 5011 CONTINUE
      JET(9) = 1
      JET(21) = 1
      GO TO 900
 5012 CONTINUE
      JET(9) = 1
      JET(21) = 1
      JET(11) = 1
      JET(17) = 1
      GO TO 900
 5013 CONTINUE
      JET( 2) = 1
      JET(20) = 1
      GO TO 900
 5014 CONTINUE
      GO TO 900
 5015 CONTINUE
      JET(17) = 1
      JET(11) = 1
      GO TO 900
```

```
5016 CONTINUE
     JET(3) = 1
     JET(15) = 1
     JET(2) = 1
     JET(20) = 1
     GO TO 900
5017 CONTINUE
     JET(3) = 1
     JET(15) = 1
     GO TO 900
5018 CONTINUE
     JET(3) = 1
     JET(15) = 1
     JET(11) = 1
     JET(17) = 1
     GO TO 900
5019 CONTINUE
     JET(9) = 1
     JET(5) = 1
     JET(14) = 1
     JET(20) = 1
     JET(18) = 1
     JET(24) = 1
     GO TO 900
5020 CONTINUE
     JET(9) = 1
     JET(18) = 1
     JET(11) = 1
     JET(14) = 1
     JET(5) = 1
     JET(20) = 1
     GO TO 900
5021 CONTINUE
     JET(9) = 1
     JET(11) = 1
     JET(14) = 1
     JET(20) = 1
     JET(18) = 1
     JET(24) = 1
     GO TO 900
5022 CONTINUE
     JET(5) = 1
     JET(20) = 1
     JET(9) = 1
     JET(18) = 1
     JET(3) = 1
     JET(24) = 1
     GO TO 900
5023 CONTINUE
     JET(9) = 1
     JET(18) = 1
     JET(3) = 1
     JET(24) = 1
```

```
      GO TO 900
 5024 CONTINUE
      JET(11) = 1
      JET(14) = 1
      JET(9) = 1
      JET(18) = 1
      JET(3) = 1
      JET(24) = 1
      GO TO 900
 5025 CONTINUE
      JET(3) = 1
      JET(5) = 1
      JET(14) = 1
      JET(20) = 1
      JET(18) = 1
      JET(24) = 1
      GO TO 900
 5026 CONTINUE
      JET(3) = 1
      JET(24) = 1
      JET(11) = 1
      JET(14) = 1
      JET( 5) = 1
      JET(20) = 1
      GO TO 900
 5027 CONTINUE
      JET(3) = 1
      JET(11) = 1
      JET(14) = 1
      JET(20) = 1
      JET(18) = 1
      JET(24) = 1
  900 CONTINUE
      IF(ISDA.EQ.0) GO TO 902
      DO 901 I= 1,12
      II    =ISIDEA(I)
  901 JET(II)=0
      GO TO 912
  902  IF(ISDB.EQ.0) GO TO 904
      DO 903 I = 1,12
      II    =ISIDEB(I)
  903 JET(II)=0
      GO TO 912
  904  CONTINUE
C
C     FAILED JETS AND EFFECTS
C
      DO 910 I=1,3
      IF(IJONS(I) .EQ. 0) GO TO 908
      IF(IJETON(I) .EQ. 0) GO TO 908
      II    =IJETON(I)
      JET(II)=1
  908 IF(IJOFS(I) .EQ. 0) GO TO 910
```

```
      IF(IJETOF(I) .EQ. 0) GO TO 910
      II    =IJETOF(I)
      JET(II)=0
  910 CONTINUE
  912 CONTINUE
C
C     CALCULATE FORCES AND MOMENTS
C
      SUMF  =0.0
      SUMFX=0.0
      NCOUNT = NCOUNT + 1
      DO 920 I=1,24
C     IF JET IS JUST NOW TURNED ON COUNT AS A FIRING STARTED.
      IF(JET(I) .LE. JETN(I)) GO TO 101
      JETC(I)=JETC(I)+1
C     JETC(I)=COUNT OF NO.OF FIRINGS STARTED FOR JET(I) ,
C        FOR THIS RUN.
  101 CONTINUE
C     IF THRUSTER STAYED OFF FROM LAST PASS,  GO TO 920
      IF(JETN(I) .EQ. 1)GO TO 918
      RTIME(I)=0.
      IF(JET(I) .EQ. 0) GO TO 920
      JTIME(I)=0
C     INCREMENT NO. OF FIRINGS THIS DATA PERIOD
      NFP=NFP+1
C     INCREMENT NO. OF FIRINGS THIS RUN.
      NFT=NFT+1
      JETN(I)=JET(I)
      GO TO 919
  918 JTIME(I)=JTIME(I)+1
      IF(JET(I) .EQ. 1) GO TO 919
C     CALCULATE ON TIME AND TOTAL ON TIME
      RTIME(I)=JTIME(I)*DT
      SUMRT(I)=SUMRT(I)+RTIME(I)
      JETN(I) = JET(I)
      GO TO 920
C     IF JET IS ON,ADD ITS THRUST LEVEL(FTR)TO FORCES & MOMENTS.
  919 CONTINUE
      IF(I .LE. 8)GO TO 913
      SUMF  =SUMF+1.0
      GO TO 914
  913 SUMFX=SUMFX+1
  914 FJX   =FJX+FTRX*FM(I,1)
      FJY   =FJY+FTR*FM(I,2)
      FJZ   =FJZ+FTR*FM(I,3)
      MX    =MX+FTR*FM(I,4)
      MY    =MY+FTR*FM(I,5)
      MZ    =MZ+FTR*FM(I,6)
  920 CONTINUE
      AFUEL=AFUEL+(DT/220.)*(FTRX*SUMFX+FTR*SUMF)
C     THE NEXT TWO STATEMENTS ARE USED TO MAKE A FILE OF DATA
C     POINTS TO USE IN THE PLOT PROGRAM
C     WRITE(4,5100)NFT,AFUEL,RO,TIME
```

```
5100  FORMAT(1X,I4,2(1X,F6.2),1X,E12.5)
 2000 CONTINUE
C      IF DATA PERIOD NOT OVER, SKIP CALCULATION OF DURATION TIME.
C       MPASS=MOD(JPASS,ISAMP)
C       JPASS=NO. PROGRAM PASSES
C       ISAMP=SAMPLING RATE ON HAND CONTROL INPUTS
       IF(MPASS .NE. 0) GO TO 927
       J=1
       L=1
       DO 921 I=1,24
C      NO ID SHOULD BE PRINTED IF IT IS EQUAL TO 99.
       ID(I)=99
       IF(RTIME(I) .EQ. 0.)GO TO 922
C      PACK ID AND DURATION TIMES
       ID(J)=I
       STIME(J)=RTIME(I)
       J=J+1
 922   CONTINUE
       IF(SUMRT(I) .EQ. SUMRTL(I))GO TO 921
C      IF TOTAL ON TIME HAS CHANGED REPACK ID AND TOTAL TIME
       IDS(L)=I
       SSUMRT(L)=SUMRT(I)
C      SET TOTAL ON TIME LAST=TOTAL ON TIME
       SUMRTL(I)=SUMRT(I)
       L=L+1
 921   CONTINUE
       IF(J .NE. 1)GO TO 924
       IF(NFP .EQ. 0)GO TO 926
C      WRITE(6,1000)NCOUNT,NFP,NFT,AFUEL,RO
       GO TO 925
 924   CONTINUE
 103   FORMAT(1H ,2X,I5)
       K=J-1
C      WRITE(7,103)K
       IF(K .GT. 10)GO TO 811
       GO TO 925
C      GO TO (801,802,803,804,805,806,807,808,809,810),K
  801  WRITE(6,1001)NCOUNT,NFP,NFT,AFUEL,RO,ID(1),STIME(1)
       GO TO 925
  802  WRITE(6,1002)NCOUNT,NFP,NFT,AFUEL,RO,(ID(I),STIME(I),I=1,2)
       GO TO 925
  803  WRITE(6,1003)NCOUNT,NFP,NFT,AFUEL,RO,(ID(I),STIME(I),I=1,3)
       GO TO 925
  804  WRITE(6,1004)NCOUNT,NFP,NFT,AFUEL,RO,(ID(I),STIME(I),I=1,4)
       GO TO 925
  805  WRITE(6,1005)NCOUNT,NFP,NFT,AFUEL,RO,(ID(I),STIME(I),I=1,5)
       GO TO 925
  806  WRITE(6,1006)NCOUNT,NFP,NFT,AFUEL,RO,(ID(I),STIME(I),I=1,6)
       GO TO 925
  807  WRITE(6,1007)NCOUNT,NFP,NFT,AFUEL,RO,(ID(I),STIME(I),I=1,7)
       GO TO 925
  808  WRITE(6,1008)NCOUNT,NFP,NFT,AFUEL,RO,(ID(I),STIME(I),I=1,8)
       GO TO 925
```

```
  809 WRITE(6,1009)NCOUNT,NFP,NFT,AFUEL,RO,(ID(I),STIME(I),I=1,9)
      GO TO 925
  810 WRITE(6,1010)NCOUNT,NFP,NFT,AFUEL,RO,(ID(I),STIME(I),I=1,10)
      GO TO 925
C     NCOUNT IS THE CURRENT PROGRAM CYCLE IN THIS RUN.
C     NFP IS NO. OF THURSTER FIRINGS PER DATA PERIOD
C     NFT IS TOTAL NO. OF THRUSTER FIRINGS FOR THIS RUN
C     JETS CAN GO ON OR OFF ONLY EVERY ISAMP PROGRAM PASS BECAUSE
C        HAND CONTROLLER INPUTS ARE ACCEPTED THEN ONLY.IN MAIN
C        ISAMP IS THE MAX. NO. OF PROGRAM CYCLES PER DATA PERIOD.
C     JETN(I)= VALUE OF JET(I) AT LAST PROGRAM CYCLE.
  811 PAUSE 'K2LARGE'
  925 CONTINUE
      NFP=0
  926 CONTINUE
      IF(L .EQ. 1)GO TO 927
      M=L-1
C     WRITE(7,103)M
      IF(M .GT. 10)GO TO 928
      GO TO 927
C     GO TO (821,822,823,824,825,826,827,828,829,830),M
  928 PAUSE 'M2LARGE'
      GO TO 927
  821 WRITE(5,1021)NCOUNT,RO,IDS(1),SSUMRT(1)
C     WRITE(4,1040)SSUMRT(1),RO
      GO TO 927
  822 WRITE(5,1022)NCOUNT,RO,(IDS(I),SSUMRT(I),I=1,2)
      GO TO 927
  823 WRITE(5,1023)NCOUNT,RO,(IDS(I),SSUMRT(I),I=1,3)
      GO TO 927
  824 WRITE(5,1024)NCOUNT,RO,(IDS(I),SSUMRT(I),I=1,4)
      GO TO 927
  825 WRITE(5,1025)NCOUNT,RO,(IDS(I),SSUMRT(I),I=1,5)
      GO TO 927
  826 WRITE(5,1026)NCOUNT,RO,(IDS(I),SSUMRT(I),I=1,6)
      GO TO 927
  827 WRITE(5,1027)NCOUNT,RO,(IDS(I),SSUMRT(I),I=1,7)
      GO TO 927
  828 WRITE(5,1028)NCOUNT,RO,(IDS(I),SSUMRT(I),I=1,8)
      GO TO 927
  829 WRITE(5,1029)NCOUNT,RO,(IDS(I),SSUMRT(I),I=1,9)
      GO TO 927
  830 WRITE(5,1030)NCOUNT,RO,(IDS(I),SSUMRT(I),I=1,10)
  927 CONTINUE
      RETURN
 1050 DO 1041 I=1,24
C     WRITE(5,1042)I,SUMRT(I)
 1042 FORMAT(1H ,I3,2X,F6.2)
 1041 CONTINUE
      RETURN
 1000  FORMAT(1H ,I5,1X,I2,1X,I4,2(1X,F6.2))
 1001 FORMAT(1H ,I5,1X,I2,1X,I4,2(1X,F6.2),1(1X,
     * '(',1X,I2,'-',F5.2,')'))
```

```
1002 FORMAT(1H ,I5,1X,I2,1X,I4,2(1X,F6.2),2(1X,
    * '(',1X,I2,'-',F5.2,')'))
1003 FORMAT(1H ,I5,1X,I2,1X,I4,2(1X,F6.2),3(1X,
    * '(',1X,I2,'-',F5.2,')'))
1004 FORMAT(1H ,I5,1X,I2,1X,I4,2(1X,F6.2),4(1X,
    * '(',1X,I2,'-',F5.2,')'))
1005 FORMAT(1H ,I5,1X,I2,1X,I4,2(1X,F6.2),5(1X,
    * '(',1X,I2,'-',F5.2,')'))
1006 FORMAT(1H ,I5,1X,I2,1X,I4,2(1X,F6.2),6(1X,
    * '(',1X,I2,'-',F5.2,')'))
1007 FORMAT(1H ,I5,1X,I2,1X,I4,2(1X,F6.2),7(1X,
    * '(',1X,I2,'-',F5.2,')'))
1008 FORMAT(1H ,I5,1X,I2,1X,I4,2(1X,F6.2),8(1X,
    * '(',1X,I2,'-',F5.2,')'))
1009 FORMAT(1H ,I5,1X,I2,1X,I4,2(1X,F6.2),9(1X,
    * '(',1X,I2,'-',F5.2,')'))
1010 FORMAT(1H ,I5,1X,I2,1X,I4,2(1X,F6.2),10(1X,
    * '(',1X,I2,'-',F5.2,')'))
1021 FORMAT(1H ,I5,F6.2,1(1X,'(',1X,I2,'-',F5.2,')'))
1022 FORMAT(1H ,I5,F6.2,2(1X,'(',1X,I2,'-',F5.2,')'))
1023 FORMAT(1H ,I5,F6.2,3(1X,'(',1X,I2,'-',F5.2,')'))
1024 FORMAT(1H ,I5,F6.2,4(1X,'(',1X,I2,'-',F5.2,')'))
1025 FORMAT(1H ,I5,F6.2,5(1X,'(',1X,I2,'-',F5.2,')'))
1026 FORMAT(1H ,I5,F6.2,6(1X,'(',1X,I2,'-',F5.2,')'))
1027 FORMAT(1H ,I5,F6.2,7(1X,'(',1X,I2,'-',F5.2,')'))
1028 FORMAT(1H ,I5,F6.2,8(1X,'(',1X,I2,'-',F5.2,')'))
1029 FORMAT(1H ,I5,F6.2,9(1X,'(',1X,I2,'-',F5.2,')'))
1030 FORMAT(1H ,I5,F6.2,10(1X,'(',1X,I2,'-',F5.2,')'))
1040 FORMAT(2(2X,F6.2))
     END
```

```
C-----------------------------------------------------------------------
C       TBA DOCKING DYNAMICS SIMULATION
C-----------------------------------------------------------------------
C       AUTHOR:  PATRICK TOBBE, JOHN GLAESE
C-----------------------------------------------------------------------
C       THIS SUBROUTINE IS THE DRIVER FOR THE CONTROL DYNAMICS
C       DOCKING DYNAMICS SUBROUTINE.  THE INPUT CONSISTS OF THE
C       CONTROL SYSTEM JOY STICK DATA AND SENSOR OUTPUT VECTOR
C       KIN AT TIME T. THE SUBROUTINE PRODUCES THE RELATIVE POSITION
C       VECTOR AND TRANSFORMATION MATRIX BETWEEN THE CHASE AND
C       TARGET VEHICLES AT TIME + DT.
C
        SUBROUTINE DYNAMIC
C
        INCLUDE 'cmmn.f'
        INCLUDE 'cmmn1.f'
        INCLUDE 'cmmn2.f'
        INCLUDE 'cmmncon.f'
        INCLUDE 'cmmnbb.f'
C-----------------------------------------------------------------------
        INTEGER I
C-----------------------------------------------------------------------
      IF (NNN) 5,5,10
5     CONTINUE
      CALL STARTT
      CALL INTFAC
      CALL FMTRAN
      CALL CNTRLC
      CALL CHASE
      CALL TARGET
      RETURN
C-----------------------------------------------------------------------
C       TOP OF INTEGRATION LOOP.
C-----------------------------------------------------------------------
10    CONTINUE
      CALL FMTRAN
      CALL CNTRLC
      CALL CHASE
      CALL TARGET
      IF (T.EQ.0.0) THEN
        DO 1500 I=1,NST
                YDO1(I)=YD1(I)
                YDO2(I)=YD2(I)
1500    CONTINUE
      ENDIF
      DO 1000 I=1,NST
      Y1(I)=Y1(I)+DT2*(3.*YD1(I)-YDO1(I))
      Y2(I)=Y2(I)+DT2*(3.*YD2(I)-YDO2(I))
 1000 CONTINUE
C-----------------------------------------------------------------------
C
C     COMPUTE INTERFACE INFORMATION
```

```
C
C-----------------------------------------------------------------------
      CALL INTFAC
C-----------------------------------------------------------------------
C     CALCULATE NEW OMEGA VECTORS
C-----------------------------------------------------------------------
      CALL MVMUL(OMV1,AIV1I,HTV1)
      CALL MVMUL(OMV2,AIV2I,HTV2)
C-----------------------------------------------------------------------
C     NORMALIZE QUATERNIONS
C-----------------------------------------------------------------------
      CALL NORMAL (QV1E)
      CALL NORMAL (QV2E)
C-----------------------------------------------------------------------
      RETURN
      END
```

```
C
C       THIS SUBROUTINE CALCULATES THE CONTACT FORCES AND MOMENTS
C       ABOUT THE CENTERS OF MASS OF THE TARGET AND CHASE VEHICLES.
C
C       THE INPUT IS THE KIN VECTOR FROM THE SENSOR
C
C       THE OUTPUT CONSISTS OF THE VALUES OF IPOT, IFREEZE, IFLAG, AND
C       NOUSE.
C
        SUBROUTINE FMTRAN
C-----------------------------------------------------------------------
        SAVE
C
        INCLUDE 'cmmn1.f'
        INCLUDE 'cmmn2.f'
        INCLUDE 'cmmnbb.f'
C-----------------------------------------------------------------------
        INTEGER*2 THRESH(6),KOFF(8)
        INTEGER I
        REAL RS1V2L(3),FKM(3),MTKM(3),FSC(3),MTSC(3),AIN(6)
        REAL RS1V1L(3),WMAX,WMAXI
C-----------------------------------------------------------------------
        DATA THRESH /100,100,200,100,100,100/
        DATA WMAX /32768./
        DATA FKM /1000.,1000.,1000./
        DATA MTKM /1000.,1000.,1000./
C-----------------------------------------------------------------------
C
C       KOFF IS THE OFFSET FROM FORCES COMING IN
C       THRESH IS THRESHOLD LIMIT ON FORCES
C       ORDER IS X,Y,Z,ROLL,YAW,PITCH
C
C       THE MAXIMUM WORD IS 32768 (16 BITS)
C       FKM AND MTKM ARE MAXIMUM OUTPUTS OF THE FORCE TABLE AS SCALED
C       IN AMPS.  ORDER IS X,Y,Z,ROLL,YAW,PITCH
C
C-----------------------------------------------------------------------
C
C     ALL FORCE AND MOMENT DATA RETURNING FROM THE SENSOR
C     ARE IN S1 OR SENSOR FRAME COORDINATES
C
C-----------------------------------------------------------------------
C
C     T1CS1 IS THE TORQUE ACTING AT THE SENSOR IN SENSOR COORDINATES
C     T1CGS1 IS THE TORQUE ACTING ON THE TARGET VEHICLE CENTER OF MASS
C     IN SENSOR COORDINATES.
C     T2CGS1 IS THE TORQUE ACTING ON THE CHASE VEHICLE CENTER OF MASS
C     IN SENSOR COORDINATES.
C
C-----------------------------------------------------------------------
```

```
C
C     F1CS1 IS THE FORCE AT THE SENSOR IN SENSOR COORDINATES
C
C-----------------------------------------------------------------
C
C     RS1V1S1 IS THE VECTOR FROM THE TARGET VEHICLE CENTER OF MASS
C     TO THE SENSOR IN SENSOR COORDINATES
C
C     RS1V2S1 IS THE VECTOR FROM THE CHASE VEHICLE CENTER OF MASS TO
C     THE SENSOR IN SENSOR COORDINATES
C
C-----------------------------------------------------------------
      IF (NNN) 5,5,15
C            COMPUTE SCALE FACTORS FOR FORCE TABLE INPUTS
  5   CONTINUE
      WMAXI=1./WMAX
C
      DO 3 I=1,3
      FSC(I)=FKM(I)*WMAXI
      MTSC(I)=MTKM(I)*WMAXI
  3   CONTINUE
C
  15  CONTINUE
      IPOT=KIN(9)
C
      IF(KIN(8) .EQ. 0) GO TO 7
C
C     NULL FORCE TABLE OFFSETS TO 0
C     ICFORCE=I2=KIN(8)
C
      DO 9 I=1,6
  9   KOFF(I)=KIN(I)
      IFLAG=1
C
  7   DO 8 I=1,6
      KIN(I)=KIN(I)-KOFF(I)
         IF(KIN(I) .GT. -THRESH(I) .AND. KIN(I) .LT. THRESH(I))KIN(I)=0
      IF(IFLAG .EQ. 0)KIN(I)=0
      AIN(I)=FLOAT(KIN(I))
  8   CONTINUE
C
C     FSC IS SCALE FACTOR ON INPUTS ( X,Y,Z,ROLL,YAW,PITCH)
C
      F1CS1(1)=AIN(1)*FSC(1)
      F1CS1(2)=AIN(2)*FSC(2)
      F1CS1(3)=AIN(3)*FSC(3)
      T1CS1(1)=AIN(4)*MTSC(1)
      T1CS1(2)=AIN(6)*MTSC(3)
      T1CS1(3)=AIN(5)*MTSC(2)
C
      IFREEZ=KIN(7)
      NOUSE=KIN(8)
C-----------------------------------------------------------------
```

```
C
C     COMPUTE CONTACT TORQUES ON TARGET VEHICLE ABOUT ITS CENTER OF MASS
C
      T1CGS1(1)=T1CS1(1)+RS1V1S1(2)*F1CS1(3)-RS1V1S1(3)*F1CS1(2)
      T1CGS1(2)=T1CS1(2)+RS1V1S1(3)*F1CS1(1)-RS1V1S1(1)*F1CS1(3)
      T1CGS1(3)=T1CS1(3)+RS1V1S1(1)*F1CS1(2)-RS1V1S1(2)*F1CS1(1)
C-----------------------------------------------------------------
C
C     TRANSFORM TARGET CONTACT TORQUES TO TARGET VEHICLE FRAME FROM
C     SENSOR FRAME
C     TRANSFORM TARGET CONTACT FORCES TO LOCAL VERTICAL FRAME FROM
C     SENSOR FRAME
C
      CALL VMMUL(T1CGV1,T1CGS1,S1V1)
      CALL VMMUL(F1CL,F1CS1,S1L)
C-----------------------------------------------------------------
C
C     COMPUTE MOMENT ARM FROM CHASE VEHICLE CENTER OF MASS
C     TO THE SENSOR IN LOCAL VERTICAL COORDINATES.
C
      CALL VMMUL (RS1V1L,RS1V1V1,V1L)
      DO 10 I=1,3
        RS1V2L(I)=R1L(I)+RS1V1L(I)-R2L(I)
10    CONTINUE
C
C     TRANSFORM MOMENT ARM TO SENSOR COORDINATES FROM TARGET VEHICLE
C     COORDINATES.
C
      CALL MVMUL(RS1V2S1,S1L,RS1V2L)
C-----------------------------------------------------------------
C
C     ASSUME EQUAL AND OPPOSITE CONTACT FORCES
C
      DO 20 I=1,3
        F2CS1(I)=-F1CS1(I)
20    CONTINUE
C-----------------------------------------------------------------
C
C     COMPUTE CONTACT TORQUES ABOUT CENTER OF MASS OF CHASE VEHICLE
C     IN SENSOR COORDINATES.
C
      T2CGS1(1)=-T1CS1(1)+RS1V2S1(2)*F2CS1(3)-RS1V2S1(3)*F2CS1(2)
      T2CGS1(2)=-T1CS1(2)+RS1V2S1(3)*F2CS1(1)-RS1V2S1(1)*F2CS1(3)
      T2CGS1(3)=-T1CS1(3)+RS1V2S1(1)*F2CS1(2)-RS1V2S1(2)*F2CS1(1)
C-----------------------------------------------------------------
C
C     TRANSFORM CHASE CONTACT TORQUES TO CHASE VEHICLE FRAME FROM
C     SENSOR FRAME
C     TRANSFORM CHASE CONTACT FORCES TO LOCAL VERTICAL FRAME FROM
C     SENSOR FRAME
C
      CALL VMMUL(T2CGV2,T2CGS1,S1V2)
      CALL VMMUL(F2CL,F2CS1,S1L)
```

```
C-----------------------------------------------------------------------
      RETURN
      END
```

```
C
C       THIS SUBROUTINE CALCULATES THE NECESSARY TRANSFORMATION
C       MATRICES USED IN THE DYNAMICS.
C
C       IT OUTPUTS THE POSITION VECTOR FROM THE TARGET DOCKING
C       PORT TO THE CHASE DOCKING PORT IN TARGET DOCKING PORT
C       COORDINATES.
C
C       IT OUTPUTS THE VELOCITY OF THE CHASE DOCKING PORT WITH
C       RESPECT TO THE TARGET DOCKING PORT IN TARGET DOCKING
C       PORT COORDINATES.
C
C       IT OUTPUTS THE ANGULAR VELOCITY OF THE CHASE VEHICLE
C       WITH RESPECT TO THE TARGET VEHICLE IN CHASE VEHICLE
C       COORDINATES.
C
C       IT ALSO OUTPUTS THE TRANSFORMATION MATRIX FROM THE CHASE
C       DOCKING PORT TO THE TARGET DOCKING PORT COORDINATE FRAMES.
C
        SUBROUTINE INTFAC
C-----------------------------------------------------------------------
        INCLUDE 'cmmn.f'
        INCLUDE 'cmmn1.f'
        INCLUDE 'cmmn2.f'
        INCLUDE 'cmmnbb.f'
C-----------------------------------------------------------------------
        REAL RD1V1L(3),RD2V2L(3),RD2D1L(3),RD2D1LD(3),OMV1L(3)
        REAL OMV2L(3),OMV1RL(3),OMV2V1L(3),WRXRD2(3)
        REAL WRLXR2(3),WRLXR1(3)
        INTEGER I,J,K
C-----------------------------------------------------------------------
C       CONSTRUCT COORDINATE TRANSFORMATIONS
C-----------------------------------------------------------------------
C       V1E
C-----------------------------------------------------------------------
      CALL AFQ(V1E,QV1E)
C-----------------------------------------------------------------------
C       V2E
C-----------------------------------------------------------------------
      CALL AFQ(V2E,QV2E)
C-----------------------------------------------------------------------
C
C       XLE IS THE TRANSFORMATION FROM THE INERTIAL E FRAME TO THE
C       LOCAL VERTICAL L FRAME
C
C-----------------------------------------------------------------------
C
C     RNU2 IS THE CURRENT ORBIT ANGLE
C
C-----------------------------------------------------------------------
```

```
      RNU2=RNU20+OMO*T
C-----------------------------------------------------------------------
      CALL EUL2(XLE,RNU2)
      CALL MMUL33(XLE,AE)
C-----------------------------------------------------------------------
C       V1L=V1E*(XLE)T.  V2L=V2E*(XLE)T.
C-----------------------------------------------------------------------
C
C     CALCULATE TRANSFORMATIONS FROM THE LOCAL VERTICAL TO THE CHASE
C     AND TARGET VEHICLE FRAMES.
C
      DO 900 I=1,3
      DO 900 J=1,3
      V1L(I,J)=0.0
      V2L(I,J)=0.0
      DO 900 K=1,3
      V1L(I,J)=V1L(I,J)+V1E(I,K)*XLE(J,K)
      V2L(I,J)=V2L(I,J)+V2E(I,K)*XLE(J,K)
900   CONTINUE
C-----------------------------------------------------------------------
C
C     COMPUTE TRANSFORMATION MATRIX FROM LOCAL VERTICAL FRAME
C     TO THE TARGET DOCKING PORT FRAME
C
      CALL MMUL3(D1L,D1V1,V1L)
C-----------------------------------------------------------------------
C
C     COMPUTE TRANSFORMATION MATRIX FROM LOCAL VERTICAL FRAME
C     TO THE CHASE DOCKING PORT FRAME
C
      CALL MMUL3(D2L,D2V2,V2L)
C-----------------------------------------------------------------------
C
C     COMPUTE TRANSFORMATION MATRIX FROM LOCAL VERTICAL FRAME
C     TO THE SENSOR FRAME
C
      CALL MMUL3(S1L,S1V1,V1L)
C-----------------------------------------------------------------------
C
C     CALCULATE TRANSFORMATION FROM CHASE VEHICLE TO TARGET VEHICLE
C     DOCKING PORT COORDINATES.
C
      DO 1540 I=1,3
      DO 1540 J=1,3
      D1D2(I,J)=0.0
      DO 1540 K=1,3
      D1D2(I,J)=D1D2(I,J)+D1L(I,K)*D2L(J,K)
1540  CONTINUE
C-----------------------------------------------------------------------
C
C     CALCULATE TRANSFORMATION FROM CHASE VEHICLE TO SENSOR COORDINATES
C
      DO 1550 I=1,3
```

```
      DO 1550 J=1,3
      S1V2(I,J)=0.
      DO 1550 K=1,3
      S1V2(I,J)=S1V2(I,J)+S1L(I,K)*V2L(J,K)
1550  CONTINUE
C-----------------------------------------------------------------------
C
C     CALCULATE POSITION AND VELOCITY OF CHASE VEHICLE DOCKING PORT
C     WITH RESPECT TO THE TARGET VEHICLE DOCKING PORT
C
C-----------------------------------------------------------------------
      CALL VMMUL (RD1V1L,RD1V1V1,V1L)
      CALL VMMUL (RD2V2L,RD2V2V2,V2L)
C
C     RD2D1L IS THE POSITION OF THE CHASE DOCKING PORT WITH RESPECT
C     TO THE TARGET DOCKING PORT IN LOCAL VERTICAL COORDINATES
C
      DO 10 I=1,3
        RD2D1L(I)=R2L(I)+RD2V2L(I)-R1L(I)-RD1V1L(I)
10    CONTINUE
C
C     OMV1L AND OMV2L ARE THE ANGULAR VELOCITIES OF THE TARGET AND CHASE
C     VEHICLES, RESPECTIVELY, IN LOCAL VERTICAL COORDINATES
C
      CALL VMMUL (OMV1L,OMV1,V1L)
      CALL VMMUL (OMV2L,OMV2,V2L)
C
C     OMV2V1L IS THE ANGULAR VELOCITY OF THE CHASE VEHICLE WITH
C     RESPECT TO THE TARGET VEHICLE IN LOCAL VERTICAL COORDINATES
C
      DO 20 I=1,3
        OMV1RL(I)=OMV1L(I)
        OMV2V1L(I)=OMV2L(I)-OMV1L(I)
20    CONTINUE
C
C     OMV1RL IS THE ANGULAR VELOCITY OF THE TARGET
C     VEHICLE WITH RESPECT TO THE LOCAL VERTICAL
C     ORBITAL RATE, IN LOCAL VERTICAL COORDINATES.
C
      OMV1RL(2)=OMV1RL(2)-OMO
C
      CALL CROSS (WRXRD2,OMV2V1L,RD2V2L)
      CALL CROSS (WRLXR1,OMV1RL,R1L)
      CALL CROSS (WRLXR2,OMV1RL,R2L)
C
      DO 30 I=1,3
        RD2D1LD(I)=R2LD(I)-WRLXR2(I)+WRXRD2(I)-R1LD(I)+WRLXR1(I)
30    CONTINUE
C-----------------------------------------------------------------------
C
C     TRANSFORM ALL RELATIVE DOCKING PORT DATA FROM THE L FRAME TO
C     THE D1 FRAME.
C
```

```
C-----------------------------------------------------------------------
      CALL MVMUL(RD2D1D1,D1L,RD2D1L)
      CALL MVMUL(RD2D1D1D,D1L,RD2D1LD)
      CALL MVMUL(OMV2V1V2,V2L,OMV2V1L)
C-----------------------------------------------------------------------
      RETURN
      END
```

```
C-----------------------------------------------------------------------
C       XFORMS
C-----------------------------------------------------------------------
C       SUBROUTINE TO GENERATE EULER ROTATION MATRIX ABOUT Y AXIS
C-----------------------------------------------------------------------
      SUBROUTINE EUL2(A,EA)
      REAL A(3,3),EA
      A(2,2)=1.0
      A(2,3)=0.
      A(2,1)=0.
      A(1,2)=0.
      A(3,2)=0.
      A(1,1)=COS(EA)
      A(3,3)=A(1,1)
      A(3,1)=SIN(EA)
      A(1,3)=-A(3,1)
      RETURN
      END
C-----------------------------------------------------------------------
C       SUBROUTINE TO GENERATE EULER ROTATION MATRIX ABOUT Z AXIS
C-----------------------------------------------------------------------
      SUBROUTINE EUL3(A,EA)
      REAL A(3,3),EA
      A(3,3)=1.0
      A(3,1)=0.
      A(3,2)=0.
      A(1,3)=0.
      A(2,3)=0.
      A(1,1)=COS(EA)
      A(2,2)=A(1,1)
      A(1,2)=SIN(EA)
      A(2,1)=-A(1,2)
      RETURN
      END
C-----------------------------------------------------------------------
C       MATRIX PRODUCT OF 3X3 MATRICES WITH RESULT IN LEFT FACTOR
C-----------------------------------------------------------------------
      SUBROUTINE MMUL33(A,B)
      REAL A(3,3),B(3,3),C(3,3)
      INTEGER I,J,K
      DO 10 I=1,3
      DO 10 J=1,3
      C(I,J)=0.
      DO 10 K=1,3
      C(I,J)=C(I,J)+A(I,K)*B(K,J)
10    CONTINUE
      DO 20 I=1,3
      DO 20 J=1,3
      A(I,J)=C(I,J)
20    CONTINUE
      RETURN
      END
```

```
C-----------------------------------------------------------------------
C       MATRIX PRODUCT OF 3X3 MATRICES WITH RESULT IN LEFT FACTOR
C-----------------------------------------------------------------------
      SUBROUTINE MMUL3(C,A,B)
      REAL A(3,3),B(3,3),C(3,3)
      INTEGER I,J,K
      DO 10 I=1,3
      DO 10 J=1,3
      C(I,J)=0.
      DO 10 K=1,3
      C(I,J)=C(I,J)+A(I,K)*B(K,J)
10    CONTINUE
      RETURN
      END
C-----------------------------------------------------------------------
C       MATRIX VECTOR PRODUCT
C-----------------------------------------------------------------------
      SUBROUTINE MVMUL(C,A,B)
      REAL C(3),A(3,3),B(3)
      INTEGER I,J
      DO 10 I=1,3
      C(I)=0.
      DO 10 J=1,3
      C(I)=C(I)+A(I,J)*B(J)
10    CONTINUE
      RETURN
      END
C-----------------------------------------------------------------------
C       VECTOR MATRIX PRODUCT
C-----------------------------------------------------------------------
      SUBROUTINE VMMUL(C,A,B)
      REAL C(3),A(3),B(3,3)
      INTEGER I,J
      DO 10 I=1,3
      C(I)=0.
      DO 10 J=1,3
      C(I)=C(I)+A(J)*B(J,I)
10    CONTINUE
      RETURN
      END
C-----------------------------------------------------------------------
C       SUBROUTINE TO COMPUTE DIRECTION COSINE MATRIX FROM QUATERNION
C-----------------------------------------------------------------------
      SUBROUTINE AFQ(A,Q)
      REAL A(3,3),Q(4),Q1SQ,Q2SQ,Q3SQ,Q4SQ
      REAL Q112,Q113,Q114,Q123,Q124,Q134
      Q1SQ=Q(1)**2
      Q2SQ=Q(2)**2
      Q3SQ=Q(3)**2
      Q4SQ=Q(4)**2
      Q112=2.0*Q(1)*Q(2)
      Q113=2.0*Q(1)*Q(3)
      Q114=2.0*Q(1)*Q(4)
```

```
      Q123=2.0*Q(2)*Q(3)
      Q124=2.0*Q(2)*Q(4)
      Q134=2.0*Q(3)*Q(4)
      A(1,1)=Q1SQ-Q2SQ-Q3SQ+Q4SQ
      A(2,2)=-Q1SQ+Q2SQ-Q3SQ+Q4SQ
      A(3,3)=-Q1SQ-Q2SQ+Q3SQ+Q4SQ
      A(1,2)=Q112+Q134
      A(2,1)=Q112-Q134
      A(3,1)=Q113+Q124
      A(1,3)=Q113-Q124
      A(2,3)=Q123+Q114
      A(3,2)=Q123-Q114
      RETURN
      END
C-----------------------------------------------------------------------
C       SUBROUTINE TO COMPUTE QUATERNIONS FROM DIRECTION COSINES
C-----------------------------------------------------------------------
      SUBROUTINE QFA(Q,A)
      REAL S(4,4),SP(4,4),A(3,3),Q(4),TR,SPMAX,SPJ,SGN
      INTEGER I,J
      DO 10 I=1,3
      DO 10 J=1,3
      S(I,J)=A(I,J)
10    CONTINUE
      S(1,4)=A(2,3)
      S(2,4)=A(3,1)
      S(3,4)=A(1,2)
      TR=A(1,1)+A(2,2)+A(3,3)
      S(4,4)=TR
      S(4,1)=-A(3,2)
      S(4,2)=-A(1,3)
      S(4,3)=-A(2,1)
      DO 20 I=1,4
      DO 20 J=1,4
      SP(I,J)=S(I,J)+S(J,I)
20    CONTINUE
      TR=1.0-TR
      DO 30 I=1,4
      SP(I,I)=SP(I,I)+TR
30    CONTINUE
      I=1
      SPMAX=ABS(SP(1,1))
      DO 40 J=2,4
      SPJ=ABS(SP(J,J))
      IF (SPJ.LT.SPMAX) GOTO 35
      I=J
      SPMAX=ABS(SP(J,J))
35    CONTINUE
40    CONTINUE
      SPMAX=SQRT(SPMAX)
      DO 50 J=1,4
      Q(J)=.5*SP(I,J)/SPMAX
50    CONTINUE
```

```
      SGN=1.0
      IF (Q(4).LT.0.) SGN=-1.0
      DO 60 I=1,4
      Q(I)=Q(I)*SGN
60    CONTINUE
      RETURN
      END
C-----------------------------------------------------------------------
C     SUBROUTINE TO CALCULATE THE INVERSE OF 3X3 MATRIX
C-----------------------------------------------------------------------
      SUBROUTINE AINV (A,B)
      REAL A(3,3),B(3,3),COFA(3,3),DETA
      INTEGER I,J
      COFA(1,1)=A(2,2)*A(3,3)-A(2,3)*A(3,2)
      COFA(1,2)=A(2,3)*A(3,1)-A(2,1)*A(3,3)
      COFA(1,3)=A(2,1)*A(3,2)-A(2,2)*A(3,1)
      COFA(2,1)=A(1,3)*A(3,2)-A(1,2)*A(3,3)
      COFA(2,2)=A(1,1)*A(3,3)-A(1,3)*A(3,1)
      COFA(2,3)=A(1,2)*A(3,1)-A(1,1)*A(3,2)
      COFA(3,1)=A(1,2)*A(2,3)-A(1,3)*A(2,2)
      COFA(3,2)=A(1,3)*A(2,1)-A(1,1)*A(2,3)
      COFA(3,3)=A(1,1)*A(2,2)-A(1,2)*A(2,1)
      DETA=A(1,1)*COFA(1,1)+A(1,2)*COFA(1,2)
     $+A(1,3)*COFA(1,3)
      DO 126 I=1,3
      DO 126 J=1,3
      B(I,J)=COFA(J,I)/DETA
126   CONTINUE
      RETURN
      END
C-----------------------------------------------------------------------
C     SUBROUTINE TO NORMALIZE QUATERNIONS
C-----------------------------------------------------------------------
      SUBROUTINE NORMAL (Q)
      REAL Q(4),QN
      INTEGER I
      QN = 0.0
      DO 10 I=1,4
        QN = QN + Q(I)**2
10    CONTINUE
      QN=1.5-.5*QN
      DO 20 I=1,4
        Q(I)=Q(I)*QN
20    CONTINUE
      RETURN
      END
C-------------------------------------------------------------------------
C       SUBROUTINE TO CALCULATE QUATERNION FROM EULER ANGLE SEQUENCE
C       1,2,3
C-------------------------------------------------------------------------
      SUBROUTINE QINT(AA1,AA2,AA3,Q)
      REAL AA1,AA2,AA3,Q(4),DTR
      REAL AA1R,AA2R,AA3R,CC1,CC2,CC3,SS1,SS2,SS3
```

```
      DTR=.0174532
      AA1R  =AA1*DTR*.5
      AA2R  =AA2*DTR*.5
      AA3R  =AA3*DTR*.5
      CC1   =COS(AA1R)
      SS1   =SIN(AA1R)
      CC2   =COS(AA2R)
      SS2   =SIN(AA2R)
      CC3   =COS(AA3R)
      SS3   =SIN(AA3R)
      Q(1)  =CC2*CC3*SS1+SS2*SS3*CC1
      Q(2)  =SS2*CC3*CC1-CC2*SS3*SS1
      Q(3)  =CC2*SS3*CC1+SS2*CC3*SS1
      Q(4)  =CC2*CC3*CC1-SS2*SS3*SS1
      RETURN
      END
C-----------------------------------------------------------------------
C       SUBROUTINE TO CALCULATE CROSS PRODUCT  C = A X B
C-----------------------------------------------------------------------
        SUBROUTINE CROSS (C,A,B)
        REAL C(3),A(3),B(3)
        C(1)=A(2)*B(3)-A(3)*B(2)
        C(2)=A(3)*B(1)-A(1)*B(3)
        C(3)=A(1)*B(2)-A(2)*B(1)
        RETURN
        END
```

```
C-----------------------------------------------------------------------
C       INITIALIZATION SECTION
C
C       THIS SUBROUTINE DEFINES THE MASS PROPERTIES AND GEOMETRY
C       OF THE CHASE AND TARGET VEHICLES.
C       IT ALSO DEFINES THE INITIAL VEHICLE CONDITIONS, ORBITAL
C       PARAMETERS, AND THE CONTROL SYSTEM VARIABLES.
C
C-----------------------------------------------------------------------
      SUBROUTINE STARTT
C-----------------------------------------------------------------------
        INCLUDE 'cmmn.f'
        INCLUDE 'cmmn1.f'
        INCLUDE 'cmmn2.f'
        INCLUDE 'cmmncon.f'
        INCLUDE 'cmmnbb.f'
C-----------------------------------------------------------------------
        REAL QDUM(4),RORBK,GMEK,ALTK,REAK,RLA2K,RLA3K
        REAL PI,TEM2(3,3),RFD,XKFM
        INTEGER I,J
C-----------------------------------------------------------------------
      PI=3.1415926
      RFD=PI/180.
C-----------------------------------------------------------------------
C     DIMENSION OF THE  STATE VECTOR
      NST=13
C-----------------------------------------------------------------------
C
C     THE ORIGIN OF THE VEHICLE FIXED FRAME, V, IS AT THE CENTER OF MASS
C
C-----------------------------------------------------------------------
C     TARGET VEHICLE MASS
      MASS1=118.11
C-----------------------------------------------------------------------
C
C     THE VEHICLE MOMENT OF INERTIA MATRIX IS IN THE VEHICLE FIXED FRAME
C     AND IS ABOUT THE VEHICLE CENTER OF MASS.
C
C     AIV IS THE INERTIA MATRIX
C     AIVI IS INVERSE THE OF INERTIA MATRIX
C
C-----------------------------------------------------------------------
C     TARGET VEHICLE MOMENT OF INERTIA MATRIX BY ROWS
      DO 10 I=1,3
        DO 10 J=1,3
                AIV1(I,J)=0.0
10    CONTINUE
      AIV1(1,1)=723.41
      AIV1(2,2)=2290.8
      AIV1(3,3)=2290.8
      CALL AINV (AIV1,AIV1I)
C-----------------------------------------------------------------------
```

```
C
C     THE INITIAL VEHICLE POSITION AND VELOCITY VECTORS ARE WITH RESPECT
C     TO THE LOCAL VERTICAL, L, FRAME AND EXPRESSED IN THE LOCAL VERTICAL
C     FRAME COORDINATES.
C
C-----------------------------------------------------------------------
C     INITIAL TARGET VEHICLE LINEAR POSITION AND RATE
      DO 20 I=1,3
        R1L(I)=0.0
        R1LD(I)=0.0
20    CONTINUE
C-----------------------------------------------------------------------
C
C     THE VEHICLE ANGULAR RATES ARE WITH RESPECT TO THE INERTIAL, E,
C     FRAME AND EXPRESSED IN THE VEHICLE FRAMES.
C
C-----------------------------------------------------------------------
C     INITIAL TARGET VEHICLE INERTIAL ANGULAR RATES
C     IN DEGREES / SECOND
      OMV1(1)=0.0
      OMV1(2)=0.0
      OMV1(3)=0.0
      DO 5 I=1,3
        OMV1(I)=OMV1(I)*RFD
5     CONTINUE
C-----------------------------------------------------------------------
C     EULER ANGLES, IN DEGREES, FOR TYPE 1,2,3 SYSTEM TO DEFINE
C     TRANSFORMATION FROM LOCAL VERTICAL TO TARGET FRAME
C     DEFINES TRANSFORMATION V1L
C-----------------------------------------------------------------------
      CALL QINT (0.,0.,0.,QDUM)
      CALL AFO (V1L,QDUM)
C
C     CHASE VEHICLE MASS
      MASS2=326.23
C-----------------------------------------------------------------------
C     CHASE VEHICLE MOMENT OF INERTIA MATRIX BY ROWS
      DO 25 I=1,3
        DO 25 J=1,3
              AIV2(I,J)=0.0
25    CONTINUE
      AIV2(1,1)=6221.
      AIV2(2,2)=3216.
      AIV2(3,3)=3394.
      CALL AINV (AIV2,AIV2I)
C-----------------------------------------------------------------------
C     INITIAL CHASE VEHICLE LINEAR POSITION AND RATE
C     WITH RESPECT TO LOCAL VERTICAL IN LOCAL VERTICAL COORD
      DO 30 I=1,3
        R2L(I)=0.0
        R2LD(I)=0.0
30    CONTINUE
C-----------------------------------------------------------------------
```

```
C     INITIAL CHASE VEHICLE INERTIAL ANGULAR RATES
C     IN DEGREES / SECOND
      OMV2(1)=0.0
      OMV2(2)=0.0
      OMV2(3)=0.0
      DO 15 I=1,3
        OMV2(I)=OMV2(I)*RFD
15    CONTINUE
C-----------------------------------------------------------------------
C     EULER ANGLES, IN DEGREES, FOR TYPE 1,2,3 SYSTEM TO DEFINE
C     TRANSFORMATION FROM LOCAL VERTICAL TO CHASE FRAME
C     DEFINES TRANSFORMATION V2L
C-----------------------------------------------------------------------
      CALL QINT (180.,0.,180.,QDUM)
      CALL AFQ (V2L,QDUM)
C-----------------------------------------------------------------------
C      INPUT DOCKING/BERTHING PORT DEFINITION DATA
C-----------------------------------------------------------------------
C
C     THE DOCKING PORT LOCATION VECTOR IS FROM THE VEHICLE CENTER OF MASS
C     TO THE ORIGIN OF THE DOCKING PORT FRAME.  IT IS EXPRESSED IN
C     THE COORDINATES OF THE VEHICLE FIXED FRAME.
C
C     S1 IS THE TARGET VEHICLE SENSOR COORDINATE FRAME
C
C     D1 IS THE DOCKING PORT COORDINATE FRAME ON THE TARGET VEHICLE
C
C     D2 IS THE CHASE VEHICLE DOCKING PORT COORDINATE FRAME
C
C     S1V1 IS THE TRANSFORMATION MATRIX FROM TARGET VEHICLE COORDINATES
C     TO SENSOR COORDINATES.
C
C     D1V1 IS THE TRANSFORMATION MATRIX FROM TARGET VEHICLE COORDINATES
C     TO DOCKING PORT COORDINATES
C
C     D2V2 IS THE TRANSFORMATION MATRIX FROM CHASE VEHICLE COORDINATES TO
C     DOCKING PORT COORDINATES OF THE CHASE VEHICLE
C
C     RS1V1V1 IS THE POSITION VECTOR OF THE SENSOR IN TARGET VEHICLE COORDINATES
C     WITH RESPECT TO THE TARGET CENTER OF MASS.
C
C     RD2V2V2 IS THE POSITION VECTOR OF THE CHASE DOCKING PORT IN CHASE
C     VEHICLE COORDINATES WITH RESPECT TO THE CHASE CENTER OF MASS.
C
C     RD1V1V1 IS THE POSITION VECTOR OF THE DOCKING PORT IN TARGET
C     VEHICLE COORDINATES WITH RESPECT TO THE TARGET CENTER OF MASS.
C
C-----------------------------------------------------------------------
C     S1V1 TRANSFORMATION BY ROWS
      DO 45 I=1,3
        DO 45 J=1,3
              S1V1(I,J)=0.0
45    CONTINUE
```

```
      S1V1(1,3)=1.
      S1V1(2,2)=1.
      S1V1(3,1)=-1.
C
C     D2V2 TRANSFORMATION BY ROWS
      DO 40 I=1,3
        DO 40 J=1,3
              D2V2(I,J)=0.0
40    CONTINUE
      D2V2(1,1)=1.
      D2V2(2,2)=1.
      D2V2(3,3)=1.
C
C     D1V1 TRANSFORMATION BY ROWS
      DO 50 I=1,3
        DO 50 J=1,3
              D1V1(I,J)=0.0
50    CONTINUE
      D1V1(1,1)=-1.
      D1V1(2,2)=1.
      D1V1(3,3)=-1.
C
C     INPUT SENSOR LOCATION VECTOR
C     ON TARGET VEHICLE
      RS1V1V1(1)=1.91667
      RS1V1V1(2)=0.0
      RS1V1V1(3)=0.0
C
C     INPUT DOCKING PORT LOCATION VECTOR
C     ON CHASE VEHICLE
      RD2V2V2(1)=3.78125
      RD2V2V2(2)=0.0
      RD2V2V2(3)=0.0
C
C     INPUT DOCKING PORT LOCATION VECTOR ON TARGET VEHICLE
      RD1V1V1(1)=4.0
      RD1V1V1(2)=0.0
      RD1V1V1(3)=0.0
C-----------------------------------------------------------------------
C
C     TRANSFORM SENSOR POSITION VECTOR FROM TARGET VEHICLE COORDINATES
C     TO SENSOR COORDINATES.
C
      CALL MVMUL(RS1V1S1,S1V1,RS1V1V1)
C-----------------------------------------------------------------------
C     INITIAL ORBIT ANGLE NU20 IN DEGREES
      RNU20=0.
      RNU20=RNU20*RFD
C-----------------------------------------------------------------------
C     NOMINAL ORBIT ALTITUDE IN NAUTICAL MILES
      ALTK=260.
C-----------------------------------------------------------------------
C     ASCENDING NODE ANGLE IN DEGREES
```

```
      RLA2K=0.
      RLA2K = RLA2K * RFD
C-----------------------------------------------------------------------
C     INCLINATION ANGLE IN DEGREES
      RLA3K=0.
      RLA3K = RLA3K * RFD
C-----------------------------------------------------------------------
      XKFM=12.0*6076.0/39.37E3
C-----------------------------------------------------------------------
      ALTK=ALTK*XKFM                          !ALTITUDE IN KM
C-----------------------------------------------------------------------
      GMEK=3.9886E5                    !KM**3-S**-2
        gmek=0.0
C-----------------------------------------------------------------------
C
C     CALCULATE THE RADIUS AND FREQUENCY OF THE NOMINAL CIRCULAR ORBIT
C     REAK IS THE RADIUS OF THE EARTH IN KILOMETERS.
C
C-----------------------------------------------------------------------
      REAK=6371.2
      RORBK=REAK+ALTK
      OMOSQ=GMEK/RORBK**3
C-----------------------------------------------------------------------
C
C     OMO IS THE FREQUENCY OF THE NOMINAL CIRCULAR ORBIT
C
      OMO=SQRT(OMOSQ)
C-----------------------------------------------------------------------
C
C     CALCULATE THE INITIAL ANGULAR MOMENTUM VECTOR FOR EACH VEHCILE.
C     THESE ARE EXPRESSED IN THE VEHICLE FIXED FRAMES.
C
C-----------------------------------------------------------------------
      CALL MVMUL(HTV1,AIV1,OMV1)
      CALL MVMUL(HTV2,AIV2,OMV2)
C-----------------------------------------------------------------------
      DO 140 I=1,NST
        YD1(I)=0.0
        YD2(I)=0.0
140   CONTINUE
C-----------------------------------------------------------------------
C
C     CALCULATE THE INITIAL TRANSFORMATIONS FROM THE INERTIAL
C     FRAME TO THE VEHICLE FIXED FRAMES.
C
C-----------------------------------------------------------------------
      CALL EUL2(TEM2,RLA2K)
      CALL EUL3(AE,RLA3K)
      CALL MMUL33(AE,TEM2)
      CALL EUL2(XLE,RNU20)
      CALL MMUL33(XLE,AE)
      CALL MMUL3(V1E,V1L,XLE)
      CALL MMUL3(V2E,V2L,XLE)
```

```
      CALL QFA(QV1E,V1E)
      CALL QFA(QV2E,V2E)
C-----------------------------------------------------------------------
C
C     INPUT INITIAL DATA FOR OMV CONTROL SYSTEM
C
C-----------------------------------------------------------------------
        RTD=57.29578
        DTR=.0174532
        IS=0
C
        NDC=TD/DT
        NDCI=NDC+1
        DO 180 I=1,NDCI
        DO 180 J=1,6
               DELAY(I,J)=0.0
180     CONTINUE
C
C       CURRENT NUMBER OF A-D SIGNALS IN THE CONTROL VECTOR KDAT IS 7
C
        NAD=7
        AFUEL=0.0
        DO 190 I=1,3
               IJONS(I)=0.0
               IJOFS(I)=0.0
190     CONTINUE
        IAUTOP=0
        MPASS=0
        ISTBSL=0
        NOISE1=.05
        NOISE2=.05
        NOISQ=NOISE1**2
        ISAMP=1
C
C       SET SCALE FACTORS FOR A-D
C
        SCX=2.
        SCY=2.
        SCZ=2.
        DO 220 I=1,7
               SSIN(I)=1.
220     CONTINUE
        SSIN(4)=10./6.7
        SSIN(5)=10./3.8
        SSIN(6)=10./7.9
        YIN=1./2047.
C
C       SET INITIAL ERROR ANGLES TO ZERO
C
        CALL QINT(0.,0.,0.,QDUM)
        P31=QDUM(1)
        P32=QDUM(2)
        P33=QDUM(3)
```

```
      P34=QDUM(4)
      P31M1=P31
      P32M1=P32
      P33M1=P33
      P34M1=P34
      PD31=0.0
      PD32=0.0
      PD33=0.0
      PD34=0.0
      PD31M1=PD31
      PD32M1=PD32
      PD33M1=PD33
      PD34M1=PD34
C
      RETURN
      END
```

```
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C       PHYSICAL SYSTEM (PLANT) SUBROUTINE FOR TARGET VEHICLE
C
C       THE TARGET VEHICLE IS DENOTED BY THE NUMBER 1.
C       THIS SUBROUTINE CALCULATES THE TIME DERIVATIVE OF THE
C       TARGET VEHICLE STATE VECTOR.  IT USES NEWTON-EULER AND
C       HILL'S EQUATIONS.
C
C       THE INPUT TORQUES ARE FROM THE VEHICLE CONTACT AND GRAVITY
C       GRADIENT.
C
C       THE INPUT FORCES ARE FROM THE VEHICLE CONTACT.
C
C-----------------------------------------------------------------------
      SUBROUTINE TARGET
C-----------------------------------------------------------------------
        INCLUDE 'cmmn.f'
        INCLUDE 'cmmn1.f'
C-----------------------------------------------------------------------
        REAL VTEM(3)
        INTEGER I,J
C-----------------------------------------------------------------------
C       EXTRACT THE PHYSICAL PARAMETERS OUT OF THE STATE VECTOR Y.
C-----------------------------------------------------------------------
C       ASSUME THAT THE STATE VECTOR IS CONSTRUCTED AS FOLLOWS:
C          Y=(HTV1,QV1E,R1LD,R1L)
C
C       HTV1 IS THE ANGULAR MOMENTUM VECTOR OF THE TARGET VEHICLE.
C       QV1E IS THE QUATERNION FOR THE TRANSFORMATION FROM THE
C             INERTIAL, E, FRAME TO THE TARGET VEHICLE FIXED FRAME.
C       R1LD IS THE VELOCITY OF THE TARGET VEHICLE CENTER OF MASS
C             WITH RESPECT TO THE L FRAME AND IN L FRAME COORDINATES.
C       R1L IS THE POSITON OF THE TARGET VEHICLE CENTER OF MASS
C             WITH RESPECT TO THE L FRAME AND IN L FRAME COORDINATES.
C
C-----------------------------------------------------------------------
      DO 100 I=1,NST
        YDO1(I)=YD1(I)
100   CONTINUE
C-----------------------------------------------------------------------
C       COMPUTE THE TOTAL GRAVITY GRADIENT TORQUES
C-----------------------------------------------------------------------
      DO 1500 I=1,3
      VTEM(I)=0.0
      DO 1500 J=1,3
      VTEM(I)=VTEM(I)+AIV1(I,J)*V1L(J,3)
1500  CONTINUE
      T1G(1)=3.0*OMOSQ*(V1L(2,3)*VTEM(3)-V1L(3,3)*VTEM(2))
      T1G(2)=3.0*OMOSO*(V1L(3,3)*VTEM(1)-V1L(1,3)*VTEM(3))
```

```
      T1G(3)=3.0*OMOSQ*(V1L(1,3)*VTEM(2)-V1L(2,3)*VTEM(1))
C-----------------------------------------------------------------------
C
C     CALCULATE THE DERIVATIVE OF THE ANGULAR MOMENTUM VECTOR
C     USING NEWTON-EULER EQUATION.
C
C-----------------------------------------------------------------------
      YD1(1)=T1G(1)+T1CGV1(1)-OMV1(2)*HTV1(3)+OMV1(3)*HTV1(2)
      YD1(2)=T1G(2)+T1CGV1(2)-OMV1(3)*HTV1(1)+OMV1(1)*HTV1(3)
      YD1(3)=T1G(3)+T1CGV1(3)-OMV1(1)*HTV1(2)+OMV1(2)*HTV1(1)
C-----------------------------------------------------------------------
C
C     CALCULATE THE DERIVATIVE OF THE QUATERNION
C
C-----------------------------------------------------------------------
      YD1(4)=.5*(OMV1(3)*QV1E(2)
     $       -OMV1(2)*QV1E(3)+OMV1(1)*QV1E(4))
      YD1(5)=.5*(-OMV1(3)*QV1E(1)
     $       +OMV1(1)*QV1E(3)+OMV1(2)*QV1E(4))
      YD1(6)=.5*(OMV1(2)*QV1E(1)
     $       -OMV1(1)*QV1E(2)+OMV1(3)*QV1E(4))
      YD1(7)=-.5*(OMV1(1)*QV1E(1)
     $       +OMV1(2)*QV1E(2)+OMV1(3)*QV1E(3))
C-----------------------------------------------------------------------
C
C     CALCULATE THE ACCELERATION OF THE VEHICLE CENTER OF MASS
C     WITH RESPECT TO THE L FRAME IN THE L FRAME COORDINATES.
C
C-----------------------------------------------------------------------
      YD1(8)=F1CL(1)/MASS1-2.*OMO*R1LD(3)
      YD1(9)=F1CL(2)/MASS1-OMOSQ*R1L(2)
      YD1(10)=F1CL(3)/MASS1+3.*OMOSQ*R1L(3)+2.*OMO*R1LD(1)
      YD1(11)=Y1(8)
      YD1(12)=Y1(9)
      YD1(13)=Y1(10)
C-----------------------------------------------------------------------
      RETURN
      END
```