DEVELOPMENT, IMPLEMENTATION, AND VALIDATION OF CONTROL ALGORITHMS FOR THE DUAL-ARM ROBOTIC MANIPULATION OF A SINGLE OBJECT

S. J. Tricamo
F. L. Swern
Peritus, Inc.
767 Broadway
Norwood, NJ 07648

N. P. Coleman
Project Engineer
ARDEC

May 1990

U.S. ARMY ARMAMENT RESEARCH, DEVELOPMENT AND ENGINEERING CENTER

Fire Support Armaments Center
Picatinny Arsenal, New Jersey

Approved for public release; distribution unlimited.
The views, opinions, and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.

The citation in this report of the names of commercial firms or commercially available products or services does not constitute official endorsement by or approval of the U.S. Government.

Destroy this report when no longer needed by any method that will prevent disclosure of contents or reconstruction of the document. Do not return to the originator.
**Title:** Development, Implementation, and Validation of Control Algorithms for the Dual-Arm Robotic Manipulation of a Single Object

**Personal Author(s):** S. J. Tricamo and F. L. Swern, Peritus, Inc., N. P. Coleman, Project Engineer, ARDEC

**Abstract:**

A control algorithm to achieve the dual-arm robotic manipulation of a single object was devised, implemented, and tested. The algorithm uses input from wrist mounted force/torque transducers to sense and correct for any trajectory inconsistencies that may develop during the cooperative manipulation of an object by two PUMA 560 robots. Test results validated the performance of the control system.

**Field of Gradation Codes:**

<table>
<thead>
<tr>
<th>Field</th>
<th>Group</th>
<th>Sub-Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>COSATI</td>
<td>CODES</td>
<td></td>
</tr>
</tbody>
</table>

**Subject Terms (Continue on Reverse if Necessary and Identify by Block Number):**

- Robotics
- Automation
- Dual-arm manipulation

**Source of Funding Numbers:**

Program Element No. | Project No. | Task No. | Work Unit Accession No.

**Title of Report:** Final

**Time Covered:**

- FROM Feb 89 TO Dec 89

**Date of Report:**

- May 1990

**Page Count:**

- 73

**DISTRIBUTION/AVAILABILITY OF ABSTRACT:**

- UNCLASSIFIED/UNLIMITED

**ABSTRACT SECURITY CLASSIFICATION:**

- UNCLASSIFIED

**NAME OF RESPONSIBLE INDIVIDUAL:**

- HAZNEDARI

**TELEPHONE (INCLUDE AREA CODE):**

- 880-3316

**OFFICE SYMBOL:**

- SM5CAR-IMI-I
7a. (cont)

AFDEC, F-SAG
Fire Control Division (SMGAR-FSF-RC)
Pica tinny Arsenal, NJ 07806-5000
CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>Background</td>
<td>2</td>
</tr>
<tr>
<td>Technical Discussion</td>
<td>3</td>
</tr>
<tr>
<td>Robot Configuration</td>
<td>3</td>
</tr>
<tr>
<td>Control Approach</td>
<td>13</td>
</tr>
<tr>
<td>Interface to Vision System</td>
<td>17</td>
</tr>
<tr>
<td>Summary of Results</td>
<td>18</td>
</tr>
<tr>
<td>Conclusions and Recommendations for Future Work</td>
<td>19</td>
</tr>
<tr>
<td>References</td>
<td>21</td>
</tr>
<tr>
<td>Appendix - Copy of Implementation Software</td>
<td>23</td>
</tr>
<tr>
<td>Distribution List</td>
<td>67</td>
</tr>
</tbody>
</table>
INTRODUCTION

The objective of this task was to develop and validate control algorithms for the dual-arm robotic manipulation of a single object. This objective has been achieved.

The specific tasks performed to attain the objective were:

1) Prepare control boards suitable for the processing and data handling necessary for the dual-arm robotic manipulation of a single object using wrist mounted force/torque sensors as feedback devices.

2) Devise and implement a means for estimating the weight of the gripped object and eliminating that reaction from the individual force/torque transducer readings.

3) Implement an interface with an external vision system that is to be used to locate the initial position and orientation of the object to be manipulated and, using that input, individually guide each arm to an initial grip point.

4) Implement a technique for removing any initial misalignment existing between either gripper and the gripped object.

5) Implement a control technique to allow the trajectory of one of the cooperating manipulators to adapt to any misalignment error occurring during the dual-arm manipulation of the object.

6) Validate the control algorithm and develop a working demonstration of the control approach, and

7) Document all control software and test results.
II. BACKGROUND

In some military applications using automation it may be necessary to manipulate a single object using multiple robot arms. This will occur when the weight of the object exceeds the payload of a single arm or when a relatively large component of a weapons system or structure must be positioned with great accuracy.

Stern and Tricamo have published a number of works dealing with two-arm robotic manipulation and control [1]-[9]. The work detailed in this report represents an extension and development of many of the concepts embodied in these publications.

The approach offers a number of advantages, including the following:

- the capability of operating each robot autonomously in the event that loss of grip or slippage occurs at the individual robot grip points on the object,

- the capability of sensing contact between the object and the environment,

- the development of an interface with an external vision system capable of providing the location of the object to be manipulated, and

- the capability being modified to allow for the teach-mode operation of the dual-arm manipulators.

Other authors have also dealt with various aspects of the control of cooperating robots [10]-[14] and similar closed-chain configurations.
The control approach required for multi-arm robotic manipulation differs significantly from that used in autonomous operation. When two arms grasp a single object, the resulting structure forms a closed kinematic chain. The number of actuators in this new structure is greater than the minimum needed to position the object. This redundancy can result in the creation of constraint forces within the object and manipulators. Such a condition is brought about when position or orientation errors caused by manipulator miscalibration, grip point slippage, or similar effects cause interaction between the various actuators in each arm. These forces may be of sufficient magnitude to make accurate positioning of the object difficult to achieve.

The approach implemented in the present work to achieve multi-arm cooperative robotic control is based upon the use of force/torque transducers to measure constraint forces in the object and manipulator system brought about by trajectory inconsistencies between individual robots. These constraint reaction measurements are used to determine the magnitude of position and orientation errors existing between the grippers of the cooperating arms. Evaluation of these errors provides the basis for a compensation scheme to control the coordination of the two arms.

A detailed description of the development, implementation, and testing of the control algorithm used to perform the tasks outline in section I of this report is given below.

A. Robot Configuration

1. Kinematic Model

A schematic of the two-arm configuration is shown in Figure III-1. The world coordinate reference frames of each
FIGURE III-1: SCHEMATIC OF TWO-ARM CONFIGURATION
robot are aligned with one another and their origins are 36.0 inches apart. The origin of the world reference frame for the multi-robot configuration is taken as coincident with that of robot 2. The coordinate axes of the wrist mounted force/torque transducers each has a different degree of misalignment with respect to their corresponding end effector frames. This misalignment is compensated for by appropriate transformations, as shown below.

As shown in the figure, the following transformations are defined for each of robots 1 and 2 as denoted by the appropriate subscripts:

\[ [T_1], [T_2] = \text{transformation from base frame to end effector frame} \]

\[ [E_1], [E_2] = \text{transformation from end effector frame to force/torque transducer frame} \]

\[ [G_1], [G_2] = \text{transformation from force/torque transducer frame to gripper frame} \]

\[ [H_{12}] = \text{transformation between frames of grippers 1 and 2, respectively} \]

\[ [B_{21}] = \text{transformation between individual world frames for robots 2 and 1, respectively} \]

Assuming perfect grip, the transformations are related as follows:

\[ [B_{21}][T_1][E_1][G_1][H_{12}] = [T_2][E_2][G_2] \quad (1) \]

where transforms \([B_{21}], [E_1], [G_1], [E_2],\) and \([G_2]\) are known and constant. Transforms \([T_1]\) and \([T_2]\) vary as each arm moves. The elements of the latter transforms are available
as output from the VAL II software in each robot controller. Determination of the remaining transform, \( H_{12} \) is the subject of the next section.

2. Force/Torque Model

Transform \( H_{12} \) could be found using equation (1) if there were no kinematic errors in either robot and if the object was perfectly located in each gripper. However, neither of these conditions is present and the determination of the relative positioning of each gripper must be made by other means.

This was accomplished using the force/torque sensor readings. Due to the fact that grip in the vertical direction (world z-axis) was loose, leading to inaccurate z-force and x-moment readings, it was decided to manipulate the object in a plane parallel to the X-Y world plane. This simplification in no way limits the applicability of the devised control technique.

A schematic of the object in grip is shown in Figure III-2. Assuming, for simplification, that the object in grip and the manipulator linkages are rigid, the compliance between the grip surfaces and the object may be modeled as parallel springs, as shown in Figure III-3. The components drawn in solid lines represent the actual positions of the gripper and object while the dashed reference line represents a datum for measuring the actual spring displacements. Only the gripper and object of robot 1 are shown; that of robot 2 would be similar in appearance.

In Figure III-3, the following nomenclature is used:

\[ x_{gi}, x_{pi} = \text{position displacement at spring end of gripper and part (object),} \]

\( (i=1,2) \) respectively, as measured from the undeflected position of the spring.
FIGURE III-2: SCHEMATIC OF OBJECT IN GRIP
FIGURE III-3: MODELING OF GRIP REACTIONS
Assuming the same spring constant, $k$, for each contacting surface, the forces developed at the attachment points for each spring are given by:

\[ f_{g1} = k(x_{g1} - x_{p1}) \tag{2} \]
\[ f_{g2} = k(x_{g2} - x_{p2}) \tag{3} \]
\[ f_{p1} = k(x_{p1} - x_{g1}) \tag{4} \]
\[ f_{p2} = k(x_{p2} - x_{g2}) \tag{5} \]

The force and torque developed on the gripper are:

\[ F_{g1} = f_{g1} + f_{g2} \tag{6} \]
\[ T_{g1} = s(f_{g2} - f_{g1}) \tag{7} \]

The position and orientation errors for the gripper corresponding to these forces and torques are given by:

\[ d_{g1} = (x_{g1} + x_{g2})/2 \tag{8} \]
\[ \theta_{g1} = (x_{g2} - x_{g1})/2s \tag{9} \]

Similar expressions for the force, torque, position error, and orientation error in the object are:

\[ F_{p1} = -F_{g1} \tag{10} \]
\[ T_{p1} = -T_{g1} \quad (11) \]

\[ d_{p1} = \frac{x_{p1} + x_{p2}}{2} \]
\[ = d_{g1} - Fr_{g1}/2k \quad (12) \]

\[ \theta_{p1} = \frac{x_{p2} - x_{p1}}{2s} \]
\[ = \theta_{g1} - T_{g1}/2ks^2 \quad (13) \]

where equations (12) and (13) were obtained with the help of equations (2),(3),(6),(7),(10, and (11).

Equations (10)-(13) may be expressed in the following matrix form:

\[
\begin{bmatrix}
F_{p1} \\
T_{p1} \\
d_{p1} \\
\theta_{p1}
\end{bmatrix}
= 
\begin{bmatrix}
-1 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 \\
-1/2k & 0 & 0 & 0 \\
0 & -1/2ks^2 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
F_{g1} \\
T_{g1} \\
d_{g1} \\
\theta_{g1}
\end{bmatrix}
\quad (14)
\]

In a similar manner, the following matrix expression may be obtained for the gripper of robot 2:

\[
\begin{bmatrix}
F_{g2} & -1 & 0 & 0 & 0 \\
T_{g2} & 0 & -1 & 0 & 0
\end{bmatrix}
= 
\begin{bmatrix}
F_{p2} \\
T_{p2}
\end{bmatrix}
\quad (15)
\]
For the rigid bar, as shown in Figure III-4, the following matrix expression relating quantities at robot 1 and 2 grip points may be determined:

\[
\begin{bmatrix}
F_{p2} \\
T_{p2} \\
d_{p2} \\
\theta_{p2}
\end{bmatrix} = \begin{bmatrix}
-1 & 0 & 0 & 0 \\
-L & -1 & 0 & 0 \\
0 & 0 & 1 & -L \\
0 & 0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
F_{p1} \\
T_{p1} \\
d_{p1} \\
\theta_{p1}
\end{bmatrix}
\]

(16)

In the above, it should be noted that if a flexible object were gripped equation (16) would have to be modified to include the elastic properties of the object. The overall approach, however, would remain the same.

Combining expressions (14), (15), and (16) results in the following relationship between reactions and displacements for the two grippers:

\[
\begin{bmatrix}
F_{g2} \\
T_{g2} \\
d_{g2} \\
\theta_{g2}
\end{bmatrix} = \begin{bmatrix}
-1 & 0 & 0 & 0 \\
-L & -1 & 0 & 0 \\
-1/k & -L/2k^2 & 1 & -L \\
-L/2k^2 & -1/k^2 & 0 & 1
\end{bmatrix} \begin{bmatrix}
F_{g1} \\
T_{g1} \\
d_{g1} \\
\theta_{g1}
\end{bmatrix}
\]

(17)
FIGURE III-4: REACTIONS ON RIGID OBJECT
To determine the relative linear and angular displacements of gripper 2 (slave) with respect to gripper 1 (master), let \( d_{g1} = \theta_{g1} = 0 \) in expression (17). Using the first two lines of this same expression, the following is obtained:

\[
\begin{bmatrix}
  d_{g2} \\
  \theta_{g2}
\end{bmatrix} = \frac{1}{2} k_s L^2 \begin{bmatrix}
  -(L^2 - 2s^2) & L \\
  -L & 2
\end{bmatrix} \begin{bmatrix}
  F_{g2} \\
  T_{g2}
\end{bmatrix} \tag{18}
\]

Expression (18) then gives a relationship between the sensed force and torque reactions at the slave gripper due to the misalignment between the grippers and the corresponding linear and angular displacements required to eliminate that misalignment. As such, it represents the basis for the control approach. By repeated sampling of the force/torque reactions, corrections as defined by expression (18) are made to the location and orientation of the slave gripper. Previous work has shown (see reference [7]) that even in the presence of noise and other errors in the sensed reactions or physical parameters the control approach is stable and will result in eliminating the misalignment over relatively few iterations.

B. Control Approach

1. Hardware Configuration

The force/torque readings were obtained using two LORD Model 15/50 transducers. This device has a parallel port available that transmits three force and three torque readings as a single group at rates up to 100 groups per second. The transducer also has a language that can be used to perform simple thresholding operations on its readings, and to logically combine these discrete outputs.
Referring to Figure III-5, the central component of the control system is an INTEL 8614 single board processor containing an 8086 processor. This processor is used as a discrete controller with an iteration time of 20 msec to match that of the PUMA 560. The controlling INTEL 8614 processor is connected to a second 8614 board, serving as an I/O controller, via a Multi-Bus I. The I/O controller contains sufficient parallel and serial ports to interface both force/torque sensors as well as the VAL II ALTER interface.

In the development of the control strategy, it was noted that the iteration time of the controller is fixed by the availability of force/torque data from the transducers and the ability of the VAL processor to accept and execute movement commands. Figure III-6 shows the iteration time of the various loops involved, including the loops internal to the robot controllers.

In testing the system, it was further noted that performance was limited by the processing capability of the 8086 processor.

2. Gravity Compensation

An advantage gained in the use of two force/torque transducers was the ability to easily measure external reactions acting on the manipulated object by simply summing the corresponding readings of each platform. In the absence of external reactions, the combined readings on all axes would be zero, indicating the presence of only constraint reactions. Finite sums indicate the presence of an external force or torque.

The only external reaction dealt with in the present application was that due to gravity. The algorithm calculates the weight of the object as indicated above. Care must be taken in evaluating this quantity to be certain that
FIGURE III-5: HARDWARE INTERCONNECTION DIAGRAM
FIGURE III-6: HARDWARE TIMING DIAGRAM

16
all readings are transformed to a common reference frame before summation is undertaken. After the weight measurement is taken, it is assumed that the weight is equally distributed between the two grippers and an appropriate correction is made to individual sensor readings to bias off this gravity effect, thereby leaving only constraint reactions readings for use in correcting misalignments.

3. Rate Filter

To avoid sudden movements due to erroneous force/torque readings due to noise or other effects, a rate filter was employed to condition the input data. This was accomplished by averaging the sensor readings on each channel over 8 samples as the iteration scheme progressed. This technique worked well, but it did result in some degradation of performance due to the slower response inherent with this approach.

C. Interface to Vision System

The control algorithm provides a means of using an external vision system to locate and transmit the initial grip point locations and orientations for the two manipulators.

If both robots have not reached their grip point, then the controller sends back position corrections only based on vision system input. After both robots reach their grip point, both navigational and constraint based feedback may occur. Coordinate changes due to vision system input must be removed for the robot to reach the target location. After navigation is complete, force feedback is implemented.
IV. SUMMARY OF RESULTS

The control algorithms were validated using the following sequence of operations for the dual-arm configuration:

1. Each robot arm moves to a preset location under autonomous operation

2. A target grip point is computed for each manipulator using input data from the vision system

3. Each arm is commanded to move to the target grip point

4. Each gripper closes to grasp the object

5. A path is computed back to the positions taught in step 1.

6. Both arms move to target position as force/torque algorithm removes any misalignment in grippers

7. With both grippers now aligned, coordinated movement to new location is accomplished under force/torque control

The test sequence was successfully executed many times using objects of differing weight and stiffness. The control algorithms functioned as intended in each case.
V. CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE WORK

A control algorithm and corresponding hardware to accomplish the dual-arm robotic manipulation of a single object was devised, implemented, and successfully tested. The algorithm uses input from wrist mounted force/torque transducers to sense and correct for any misalignment or trajectory inconsistencies that may develop during the cooperative manipulation of the test object. The control algorithm is capable of taking in data from an external vision system to guide each robot arm to its initial grip position.

The existing system represents one of the few dual-arm robotic test facilities in the nation. To further improve and extend its performance capabilities, the following suggestions are made for future work:

- upgrade the processor in the control system to a 386
- interface the control system to Matrix AR
- develop a dual-arm teach mode capability
- evaluate the possibility of attaining significant payload increases using the inherent strength of the dual-arm closed-chain configuration
- incorporate a dual-arm control technique capable of using external force signals to accomplish a given objective (assembling parts, for example)
- evaluate the advantage and feasibility of multi-arm manipulation using more than two manipulators.
REFERENCES


11. Alfred and Salvesen; "Coordinated control of two robot arms"; Proc. IEEE Int'l Conf. on Robotics; 1984; pp. 468-473


13. Ishida; "Force Control in Coordination of Two Arms"; Proc. of 5th Int'l Joint Conf. on Artificial Intelligence, Aug. 1977; pp. 717-722

APPENDIX

COPY OF IMPLEMENTATION SOFTWARE
1  BLOCK DATA
  
  C NAME:  Block Data
  C PURPOSE:  Initialize Common Blocks
  C DESCRIPTION:  This module contains values of constants used in other
  C FORTRAN modules in the program:
  C
  C Main Common Variables are set to zero to insure that the controller
  C starts from some known state.
  C
  $INCLUDE (FTORQUE.INC)
  1  $NOLIST
  2=1 INTEGER FTMFLAG, FTD0IT, FTDONE, FSTART, FTMFLAG2, ROBDIF1, ROBDIF2
  3=1 INTEGER FTGRI5, R2MSG, GRIPPD, STARTD, R1MSG, FCALC, AUTO, FTAUTO
  4=1 INTEGER MVSKED, MVBAK, MVRAK
  5=1 CHARACTER*40 PROMPT
  6=1 CHARACTER*3 DISPT
  7=1 LOGICAL*1 RESET, FTPLOT
  8=1 C PARAMETER (R2MSG=2, R1MSG=1)
  9=1 PARAMETER (AUTO=34, GRIPPD=20, STARTD=65)
  10=1 PARAMETER (FSTART=1, FTAUTO=2, FTGRI5=4, FTDONE=8)
  11=1 PARAMETER (FTDOIT=16, NAVSW=32, MOVEDONE=64, FCALC=128)
  12=1 PARAMETER (MVSKED=1, MVBAK=2, MVRAK=4)
  13=1 COMMON/FTCOM/ FORCE(14), FTBIAS(14), FTDIF(14), FSCALE, TSCA5E, TDZ
  14=1 COMMON/FTSUM/F12SUM(6), F22SUM(6), F2SUM(6), FSUM(6)
  15=1 COMMON/FTMAT/XGAIN, YGAIN, ZGAIN, ALENTH, FTMAT2(6,6),
  =1 1 FTMAT(6,6), RSCALE(6)
  16=1 COMMON/XFORM/CTTOW1(3,3), CTTOW2(3,3), GT00L2(6,6),
  =1 4 FTOOL1(3), FTOOL2(3), MTOOL1(3), MTOOL2(3), GTOOL1(3,3)
  17=1 COMMON/VECT/DX, DY, DZ, DX2, DY2, PHI22, DPHIM
  18=1 COMMON/MSCOM/PROMPT, RESET, FTPLOT(28)
  19=1 COMMON/MCNTL/IOUTER, IFLAG, DISPT, FTMFLAG, FTMFLAG2
  20=1 COMMON/NAVECT/BX, BY, BZ, RACKPT(4,2), ROBPT1(6), ROBPT2(6)
  21=1 COMMON/NAVEPT/ROBDIF1(6), ROBDIF2(6), ROBDIF2(6), ROBDIF2(6),
  =1 1 RDMAX, RMAX, NCALC
  22=1 DATA FORC3E, FTBIAS, FTDIF/42*0.0/
  23=1 DATA RSCALE/32.0, 32.0, 32.0, 182.0444, 182.0444, 182.0444/
  24=1 DATA FTMFLAG, FTMFLAG2/2*0/
  25=1 DATA FTPLOT/36*0.0/
  26=1 DATA PROMPT/ 'ENTER COMMAND' /
  27=1 END
SUBROUTINE CINIT(I,IN)

NAME: Constant Initialization

PURPOSE: Initialize Constants in the FORTRAN programs

DESCRIPTION: This program allows certain constants to be set in mem
and then moved over to be used by the FORTRAN subroutines. It is
easier to change these constants, using the debug monitor, on the
fly rather than recompile the FORTRAN program.

INTEGER*2 IN,I
$INCLUDE (FTORQUE.INC)
$Nolist

1 INTEGER FTMFLAG, FTDOIT, FTDONE, FSTART, FTMFLAG2, ROBDIF1, ROBDIF2
2 INTEGER R2MSG, GRIPPD, STARTD, R1MSG, FCALC, AUTO, FTAUTO
3 INTEGER MVSKED, MVBK, MVRK
4 CHARACTER*40 PROMPT
5 CHARACTER*3 DISPT
6 LOGICAL*1 RESET, FTPLOT
7 C
8 PARAMETER (R2MSG=2, R1MSG=1)
9 PARAMETER (AUTO=34, GRIPPD=20, STARTD=65)
10 PARAMETER (FSTART=1, PTAUTO=2, FTMGRIP=4, FTDONE=8)
11 PARAMETER (FTDOIT=16, NAVSW=32, MOVEDONE=64, FCALC=128)
12 PARAMETER (MVSKED=1, MVBK=2, MVRK=4)
13 COMMON/FTCOM1/FORCE(14), FTBIAS(14), FTDF(14), FSCL, TSCALE, TDZ
14 COMMON/FTSUM/F1ZSUM(6), F2ZSUM(6), F2SUM(6)
15 COMMON/FTMAT/XGAIN, YGAIN, ZGAIN, 2GAIN, ALNTH, FTMAT2(6, 6),
16 COMMON/FTMATX/XGAIN, YGAIN, ZGAIN, 2GAIN, ALNTH, FTMAT2(6, 6),
17 COMMON/XFORM/CTTOW(3, 3), CTTOW2(3, 3), GTOOL2(6, 6),
18 COMMON/FOC/DX, DY, DZ, DX2, DY2, PHIZ2, DPHIM
19 COMMON/MSG/COM/PROMPT, RESET, FTPLOT(28)
20 COMMON/MCTRL/IOUTER, IFLAG, DISPT, FTMFLAG, FTDFLAG2
21 COMMON/NAVEXT/BD, BY, BZ, RACKPT(4, 2), ROBPT1(6), ROBPT2(6)
22 COMMON/NAVEXT/ROBDIF1(6), ROBDIF2(6), ROBDIF3(6), ROBDIF2(6),
23 IOUTER=0

This call sets the words "REINIT" on the monitor screen to verify tha
the RS232 interface to that screen is operational

CALL DJSCH('REINIT')

To set the Debug print flag

IF(I.EQ.3) FTMFLAG=IN

Scaling constants for the force/torque algorithms

FSCL=1.0
TSCL=1.0

Set force/torque matrix gains
FORTRAN-86 COMPILER
CONTRL.FOR

28 XGAIN=.0075
29 YGAIN=.0004
30 ZGAIN=.0030
31 ALENTH=11.0
32 CALL GAINM

C Rate Constants used for the algorithms
C
33 DPHIM = .02
34 RDMAX = 5.0
35 RTMAX = 2.0
C Set the initialization flag and other flags
C
36 FTMFLAG=0
37 FTMFLAG2=0
38 NCALC=0
C Clear arrays and data areas to prevent startup transients
C
39 DO 10 J=1,6
40 ROBPT1(J)=0.0
41 ROBPT2(J)=0.0
42 ROBWDIF1(J)=0.0
43 ROBWDIF2(J)=0.0
44 ROBDIF1(J)=0
45 10 ROBDIF2(J)=0
C Reset grip angle to zero
C
46 PHIZ2=0.0
47 CALL TOOLT2(0.0,0.0,0.0,-1)
C Set base offset of two robots
C
48 BX=-36.0
49 BY=0.0
50 BZ=0.0
51 RETURN
52 END
SUBROUTINE CKFTF (I, IFTIN)
C
NAME: Check Force/Torque Data
C
PURPOSE: Preprocess data from each axis of Force/Torque Sensor
C
DESCRIPTION: The raw Force/Torque data is corrected for sensed
gravity.
C
INPUTS:  I  - Force/Torque sensor axis identification number
C  IFTIN  - Force/Torque sensor reading
C
INTEGER*2 I, IFTIN, FTID(14)
REAL VFIL, DOBIAS
DATA FTID/1, 2, 3, 4, 0, 1, 5, 6, 3, 4, 0/

$INCLUDE (FTORQUE.INC)
$NOLIST

INTEGER FTMFLAG, FTDIOT, FTDONE, FSTART, FTMFLAG2, ROBDIF1, ROBDIF2
INTEGER FTGRIPE, R2MSG, GRIPPD, STARTD, RIMG, FSCALE, AUTO, FTAUTO
INTEGER MVSKED, MVBAK, MVRAK
CHARACTER*40 PROMPT
CHARACTER*3 DISPT
LOGICAL*1 RESET, FTPLOT

PARAMETER (R2MSG=2, RIMG=1)
PARAMETER (AUTO=34, GRIPPD=20, STARTD=65)
PARAMETER (FSTART=1, FTAUTO=2, FTGRIPE=4, FTDONE=8)
PARAMETER (FTDIOT=16, NVSW=32, MOVEDONE=64, FSCALE=128)
PARAMETER (MVSKEQ=1, MVBAK=2, MVRAK=4)

COMMON/FTCOM/FORCE(14), FTBIAS(14), FTDIFF(14), FSCALE, TSACLE, TDZ
COMMON/FTSUM/FZSUM(6), FSUM(6), FSUM(6)
COMMON/XGAIN, YGAIN, ZGAIN, ZGAIN, ALENTH, FTMAT2(6, 6),
   FTMAT(6, 6), MSCALE(6)
COMMON/XFORM/CTTOW1(3, 3), CTTOW2(3, 3), GTOOL2(6, 6),
   GTOOL1(3, 3), GTOOL1(3, 3)
COMMON/VECT/DX, DY, DZ, DX2, DX2, PHI2, DPHI
COMMON/MSCCOM/PROMPT, RESET, FTPLOT(28)
COMMON/MCTFL/IOUTER, IFLAG, DISPT, FTNAME, FTDIFF2
COMMON/NAVPT/ROBDIP1(6), ROBDIP2(6), ROBPT1(6), ROBPT2(6),
   RDMAX, RMAX, NCALC

JFTIN=IFTIN
FRIN=JFTIN
GOTO (1, 2, 3, 4, 6, 7), FTID(I)

FORCE(I)=FRIN
GOTO 5

FORCE(I)=FRIN+TDZ*FORCE(I+4)
GOTO 5

FORCE(I)=FRIN-TDZ*FORCE(I+2)
GOTO 5
FORTRAN-86 COMPILER
CCNTRL.FOR

C
34 4
FORCE(I) = FRIN
35
GOTO 5
C
36 6
FORCE(I) = FRIN - TDZ * FORCE(I+4)
37
GOTO 5
C
38 7
FORCE(I) = FRIN + TDZ * FORCE(I+2)
39
GOTO 5
C
Calculate the Gravity compensated force (FTDIF)
C
40 5
FTDIF(I) = FORCE(I) - FTBIAS(I)
C
41
RETURN
42
END
SUBROUTINE CONTL(DATAFLG,SYSFLAG,ROB1DAT,ROB1CMD,ROB2DAT,ROB2CMD,
   FTSAMP1,FTSAMFP2)

NAME: Control

PURPOSE: Performs high rate (inner loop) force/torque and positioning
         calculations to interface the VAL alter command

DESCRIPTION: A new set of force/torque readings is processed. If
             the robots have started, but are not at their grip points,
             guidance is supplied to the new grip point. After the grip
             point is reached, the force/torque matrix multiplies each force
             reading to obtain a new set of displacements.

INPUTS: DATAFLG = Flags showing which data is valid on input, out
        SYSFLAG = Flags showing the state of the system
        ROB1DAT = Input data from robot 1 including positional tr
        ROB2DAT = Input data from robot 2 including positional tr
        FTSAMP1 = Two force/torque samples from the wrist of robo
        FTSAMP2 = Two force/torque samples from the wrist of robo

OUTPUTS: ROB1CMD = New positional delta for robot 1
         ROB2CMD = New positional delta for robot 2

INTEGER*2 ROB1DAT(15),ROB2DAT(15),ROB1CMD(7),ROB2CMD(7),
   FTSAMP1(36),FTSAMFP2(36),JDFLAG
3 INTEGER*1 DATAFLG,SYSFLAG,IDFLAG
4 EQUIVALENCE (IDFLAG,JDFLAG)

#include (Ftorque,inc)

5=1 INTEGER FTMFLAG,FTDOIT,FTDONE,FSTART,FTMFLAG2,ROBDIF1,ROBDIF2
6=1 INTEGER FTRGRIP,R2MSG,GRIFFD,STARTD,R1MSG,FCALC,AUTO,FTAUTO
7=1 INTEGER MVSKEW,MVBAK,MVRAK
8=1 CHARACTER*40 PROMPT
9=1 CHARACTER*3 DISPT
10=1 LOGICAL*1 RESET,FPTPLOT
=1 C
11=1 PARAMETER (R2MSG=2,R1MSG=1)
12=1 PARAMETER (AUTO=34,GRIFFD=20,STARTD=65)
13=1 PARAMETER (FSTART=1,FTAUTO=2,FTGRIFF=4,FTDONE=8)
14=1 PARAMETER (FTDOIT=16,MVSEW=32,MVSEDONE=64,FCALC=128)
15=1 PARAMETER (MVSEK=1,MVBAK=2,MVRAK=4)
=1 C
16=1 COMMON/FTCOM/FORCE(14),FTBIAS(14),FTDIF(14),FSCALE,TScale,TDZ
17=1 COMMON/FTSUM/F2SUM(6),F2SUM(6),F3SUM(6),F3SUM(6)
18=1 COMMON/FTMAT/AGAIN,YGAIN,2GAIN,2GAIN,2GAIN,2GAIN,2GAIN,2GAIN,2GAIN,2GAIN,
19=1 COMMON/XFORM/CTTOW1(3,3),CTTOW2(3,3),CTTOW3(3,3)
20=1 COMMON/FTOOL(3),FTOOL2(3),FTOOL3(3),FTOOL4(3)
21=1 COMMON/VECT/DX,DY,DZ,DX2,DX2,PHIZ2,DPHTM
22=1 COMMON/MSGCOM/PROMPT,RESET,FPTPLOT
23=1 COMMON/HUNET/IUSER IFLAG,DISPT,FTMFLAG,FTMFLAG
24=1 COMMON/NAVEXT/BX,BY,RAKPT(4,2),ROBT1(6),ROBT2(6)
25=1 COMMON/RMVEPT/ROB1(6),ROB2(6),ROBDIF1(6),ROBDIF2(6),ROBDIF1(6),ROBDIF2(6)
26=1 COMMON/RMVE1/RMAX,RTMAX,RCALC
FORTRAN-86 COMPILER
CCNTRL.FOR

C Translate force/torque variables to floating point
C
DO 10 I=6,1,-1
CALL CKFTF(I,FTSAMPl(I+1))
CALL CKFTF(I+7,FTSAMP2(I+1))
CONTINUE

IF(IAND(IFLAG,STARTD).EQ.STARTD) THEN
IDFLAG=DATAFLG

Translation algorithms to new rack location
IF (IAND(FTMFLAG,FTGRIP).EQ.0) THEN
If both robots have not reached the grip point, then
the controller sends back position corrections only
derived from camera input
IF(IAND(FTMFLAG,FCALC).NE.0) THEN
NCALC=NCALC+1

Robot Position Commands
ROBlCMD(1)=256*63
ROB2CMD(1)=256*63
DO 20 I=1,6
ROBlCMD(I+1)=ROBDIF1(I)
ROB2CMD(I+1)=ROBDIF2(I)
20 CONTINUE
JDFLAG=IOR(JDFLAG,R2MSG)
JDFLAG=IOR(JDFLAG,R1MSG)
ENDIF
ELSE
Force/torque Control Algorithms
After both robots reach grip point, both navigational and force
feedback may occur. Coordinate changes due to camera input must
be removed for robot to reach target location. After navigation
is complete, force feedback is implemented.
ROB2CMD(1)=256*63
ROBlCMD(1)=256*63
IF(IAND(FTMFLAG,FCALC).NE.0) THEN
Do navigational movement
NCALC=NCALC+1
DO 30 I=1,6
ROBlCMD(I+1)=ROBDIF1(I)
ROB2CMD(I+1)=ROBDIF2(I)
30 CONTINUE
ELSE
Do force/torque feedback
DO 32 I=1,6
31
FORTRAN-86 COMPILER
CONTRL.FOR

53 XSUM=0.0
54 IF (IAND(FTMFLAG,FTDOIT).NE.0) THEN
55 DO 35 J=1,6
56 35 XSUM=XSUM+FTDIF(J+7)*FTMATX(I,J)
57 ENDIF
58 ROB2CMD(I+1)=XSUM*RSCALE(I)
59 ROB1CMD(I+1)=0
60 CONTINUE
61ENDIF

C Set robot messages are available
C
62 JDFLAG=IOR(JDFLAG,RI MSG)
63 JDFLAG=IOR(JDFLAG,R2MSG)
64 ENDIF
65 DATAFLAG=IDFLAG

C
66 ENDIF

C
67 RETURN
68 END
SUBROUTINE OUTER(FORTFLAG, SYSFLAG, ROB1DAT, ROB1CMD, ROB2DAT, ROB2CMD, 
                   FTSAMP1, FTSAMP2)
C
NAME: Outer Control loops
C
PURPOSE: Perform low rate control calculations including gravity 
         compensation and navigation
C
DESCRIPTION: This routine updates system flags in common and clears 
               all flags, etc. when the task is complete. It also calls 
               gravity compensation subroutine and navigation subroutine to 
               update (slow moving) quantities for the next group of iteration 
C
INPUTS: FORTFLAG  - Flags summarizing the system state 
           SYSFLAG  - Flags showing the state of each robot in the sy 
           ROB1DAT  - Input data from robot 1 including positional tr 
           ROB2DAT  - Input data from robot 2 including positional tr 
           FTSAMP1  - Two Force/Torque samples from the wrist of robo 
           FTSAMP2  - Two Force/Torque samples from the wrist of robo 
C
OUTPUTS: ROB1CMD  - Robot 1 commands zeroed when robot 1 is done 
           ROB2CMD  - Robot 2 commands zeroed when robot 2 is done 

INTEGER*2 ROB1DAT(15), ROB2DAT(15), ROB1CMD(7), ROB2CMD(7), 
1 FTSAMP1(36), FTSAMP2(36), JDFLAG, JFFLAG 
1 INTEGER SEG, MOTION, T11, T21, T31, T12, T22, T32, T13, T23, T33, T14, 
1 T24, T34 
1 PARAMETER(SEG=2, MOTION=3, T11=4, T21=5, T31=6, T12=7, T22=8, 
1 T32=9, T13=10, T23=11, T33=12, T14=13, T24=14, T34=15) 
1 INTEGER*1 DATAFLG, SYSFLAG, IDFLAG, IFFLAG, FORTFLAG(2) 
1 EQUIVALENCE (IDFLAG, JDFLAG), (IFFLAG, JFFLAG)
C
$INCLUDE (FTORQUE.INC)
=1 $NOLIST 
7=1 INTEGER FTMFLAG, FTDIOT, FTDONE, FSTART, FTMFLAG2, ROBDIF1, ROBDIF2 
8=1 INTEGER FTGRIPl, R2MSG, GRIPPD, STARTD, R1MSG, FCALC, AUTO, FTAUTO 
9=1 INTEGER MVSKEPl, MVBAK, MVRAK 
10=1 CHARACTER*40 PROMPT 
11=1 CHARACTER*3 DISPF 
12=1 LOGICAL*1 RESET, FTPLOT 
=1 C 
13=1 PARAMETER (R2MSG=2, R1MSG=1) 
14=1 PARAMETER (AUTO=34, GRIPPD=20, STARTD=65) 
15=1 PARAMETER (FSTART=1, FTAUTO=2, FTGRIPl=4, FTDONE=8) 
16=1 PARAMETER (FTDIOPl=16, NAVSW=32, MOVEDONE=64, FCALC=128) 
17=1 PARAMETER (MVSKEPl=1, MVBAK=2, MVRAK=4) 
=1 C 
18=1 COMMON/FTCOM/FORCE(14), FTBIAS(14), FTDIF(14), FSCALE, TSCHRE, TDZ 
19=1 COMMON/FTSUM/F1SUMP, FZSUMP(6), F2SUMP(6), FSUMP(6) 
20=1 COMMON/FTMAT/XGAIN, YGAIN, ZGAIN, ALENTH, FTMAT2(6, 6), 
1 FTMATX(6, 6), FSCALE(6) 
21=1 COMMON/XFORM/CTTW1(3, 3), CTTW2(3, 3), GTOOLL(3, 3), GTOOL2(6, 6), 
=1 4 FTOOLL(3), FTOOL2(3), MTOOLL(3), MTOOL2(3), GTOOLL(3, 3) 
22=1 COMMON/VECT/DX, DY, DZ, DX2, DY2, PHII2, DPHIM
FORTRAN-86 COMPILER
CCNTRL.FOR

COMMON/MSGCOM/PROMPT, RESET, FTPL(28)
COMMON/NCNTL/IOUTER, IFLAG, DISPT, FTMFLAG, FTMFLAG2
COMMON/NAVEC/BX, BY, BZ, RACKP(4,2), ROBPT1(6), ROBPT2(6)
COMMON/NAVEP/ROBDIF1(6), ROBDIF2(6), ROBDIF1(6), ROBDIF2(6),
=1 1 RMAX, RTMAX, NCALC

IOUTER=IOUTER+1

Housekeeping for flags and variables

JDFLAG=0
JFFLAG=0
IDFLAG=SYSFLAG
IFFLAG=FORTFLAG(1)
IFLAG=JDFLAG
FTMFLAG=IAN(FTMFLAG,240)+IAN(JFFLAG,15)

Reset everything when robot has dropped START

IF (IAN(FTMFLAG,FSTART).NE.FSTART) THEN
ROB1CMD(1)=0
ROB2CMD(1)=0
DO 10 I=1,6
ROB1CMD(I+1)=0
ROB2CMD(I+1)=0
10 CONTINUE
NCALC=0
FTMFLAG=0
FTMFLAG2=0
ELSE
Call Gravity Compensation and Make Navigational Calculations

CALL GRAV(ROB1DAT,ROB2DAT)

Rack coordinates supplied by the camera are represented as increments
rack coordinates the robot is taught. To properly compensate,
navigation is done by the following steps:
1) Move to grip point robot was taught
2) Compute true grip point from camera increments
3) Move to true grip point
4) Grasp object
5) Compute path to taught grip point
6) Move back to taught grip point
7) Continue force/torque algorithms

IF (IAN(IFLAG,GRIPPD).EQ.GRIPPD) THEN
IF (IAN(FTMFLAG,FTGRIP).EQ.0) THEN
At grip point, but not gripped
IF (IAN(FTMFLAG2,MVSKED).NE.0) THEN
Then compute true grip point, set flags to do only once
CALL RAXMOV(1)
FTMFLAG2=IAN(FTMFLAG2,255-MVSKED)
FTMFLAG2=IOR(FTMFLAG2,MVBK)
FTMFLAG=IOR(FTMFLAG,NAVSW)
ENDIF
ENDIF
ENDIF

34
ELSE
   C Object gripped
   IF (IAND(FTMFLAG2,MVBAK).NE.0) THEN
   C Compute path back to taught grip point, reset flags
      CALL RAKMOV(2)
      FTMFLAG2=IAND(FTMFLAG2,255-MVBAK)
      FTMFLAG=IOR(FTMFLAG,NAVSW)
   ENDIF
ENDIF
C
C Compute movement increments for current path
C
CALL NAV(ROB1DAT,ROB2DAT)
ENDIF
C
C Return synchronization flag to main process
C
JFFLAG=IAND(FTMFLAG,240)
FORTFLAG(2)=JFFLAG
RETURN
END
SUBROUTINE GRAV(ROB1DAT, ROB2DAT)

C NAME: Gravity Compensation
C PURPOSE: Compute external forces on the object and set FTBIAS to reflect these external forces
C DESCRIPTION: The transformation matrix supplied by VAL is translated into real numbers. A vector between the grippers is also calculated. A dimensional vector representing the direction from robot 2 to robot 1 in robot 2 tool coordinates is calculated. Forces are then translated to the world coordinate system, and summed. It is assumed that any forces after summation is due to external forces, and these forces are shared between the two robots by setting FTBIAS.

C INPUTS: ROB1DAT - Robot 1 transformation data supplied by VAL
C ROB2DAT - Robot 2 transformation data supplied by VAL

INTEGER*2 ROB1DAT(15), ROB2DAT(15)
INTEGER SEG, MOTION, T11, T21, T31, T12, T22, T32, T13, T23, T33, T14, T24, T34
PARAMETER(SEG=2, MOTION=3, T11=4, T21=5, T31=6, T12=7, T22=8, T13=9, T14=10, T23=11, T33=12, T14=13, T24=14, T34=15)

$INCLUDE (FTORQUE.INC)
$NOLIST

INTEGER FTFLAG, FTDOIT, FTDONE, FSTART, FTMPFLAG1, FTMPFLAG2
INTEGER ROBDIF1, ROBDIF2
INTEGER MVSKED, MVRAK
INTEGER MVRAK(40), PROMPT
CHARACTER*3 DISPT
LOGICAL*1 RESET, FTPLOT
PARAMETER (R2MSG=2, R1MSG=1)
PARAMETER (AUTO=34, GRIPPP=20, STARTD=65)
PARAMETER (FSTART=1, FTAUTO=2, FTGRIP=4, FTDONE=8)
PARAMETER (FTDOIT=16, NAVSW=32, MOVEDONE=64, FCALC=128)
PARAMETER (NVSKED=1, MVBAK=2, MVRAK=4)

COMMON/FTCOM/FORCE(14), FTBIAS(14), FTDIFF(14), FSCALE, TSCALE, TDZ
COMMON/FTSUM/FSUM16(6), FSUM26(6), FSUM36(6), FSUM46(6)
COMMON/FTMAT/XGAIN, YGAIN, ZGAIN, ALENT1, FTHAT2
COMMON/XFORM/CTT0O1(3,3), CTTOW2(3,3), CTOOL1(3), CTOOL2(6,6)
COMMON/XFORM/CTT0O1(3,3), CTT002(3,3), CTOOL1(3), CTOOL2(3), CTOOL1(3,3)
COMMON/VECT/DX, DY, DZ, DX2, DY2, PHIZ2, DHIM
COMMON/MSGCOM/PROMPT, RESET, FTPLOT(28)
COMMON/NAVEC/BX, BY, BZ
COMMON/NAVEPT/ROBDIF1 (6), ROBDIF2 (6), ROBDIF3 (6), ROBDIF4 (6), ROBDIF5 (6)
COMMON/NAVEPT/ROBDIF1 (6), ROBDIF2 (6), ROBDIF3 (6), ROBDIF4 (6), ROBDIF5 (6)
COMMON/NAVEPT/ROBDIF1 (6), ROBDIF2 (6), ROBDIF3 (6), ROBDIF4 (6), ROBDIF5 (6)
RMAX, RTMAX, NCALC

C Calculate transformations and position vectors from VAL inputs
FORTRAN-86 COMPLTIER
CONTRL. FOR

C Compute Robot positions
RX1=REAL(ROB1DAT(T14))/32.0
RY1=REAL(ROB1DAT(T24))/32.0
RZ1=REAL(ROB1DAT(T34))/32.0
RX2=REAL(ROB2DAT(T14))/32.0
RY2=REAL(ROB2DAT(T24))/32.0
RZ2=REAL(ROB2DAT(T34))/32.0

C Compute distances between grippers
DX=BX+(RX1-RX2)/25.4
DY=BY+(RY1-RY2)/25.4
DZ=BZ+(RZ1-RZ2)/25.4
TDZ=6.0

C Directional cosines
CTTOW1(1,1)=REAL(ROB1DAT(4))/16384.0
CTTOW1(2,1)=REAL(ROB1DAT(5))/16384.0
CTTOW1(3,1)=REAL(ROB1DAT(6))/16384.0
CTTOW1(1,2)=REAL(ROB1DAT(7))/16384.0
CTTOW1(2,2)=REAL(ROB1DAT(8))/16384.0
CTTOW1(3,2)=REAL(ROB1DAT(9))/16384.0
CTTOW1(1,3)=REAL(ROB1DAT(10))/16384.0
CTTOW1(2,3)=REAL(ROB1DAT(11))/16384.0
CTTOW1(3,3)=REAL(ROB1DAT(12))/16384.0

CTTOW2(1,1)=REAL(ROB2DAT(4))/16384.0
CTTOW2(2,1)=REAL(ROB2DAT(5))/16384.0
CTTOW2(3,1)=REAL(ROB2DAT(6))/16384.0
CTTOW2(1,2)=REAL(ROB2DAT(7))/16384.0
CTTOW2(2,2)=REAL(ROB2DAT(8))/16384.0
CTTOW2(3,2)=REAL(ROB2DAT(9))/16384.0
CTTOW2(1,3)=REAL(ROB2DAT(10))/16384.0
CTTOW2(2,3)=REAL(ROB2DAT(11))/16384.0
CTTOW2(3,3)=REAL(ROB2DAT(12))/16384.0

C Direction to Robot 1 in Robot 2 Coordinates
DX2=CTTOW2(1,1)*DX+CTTOW2(2,1)*DY+CTTOW2(3,1)*DZ
DY2=CTTOW2(1,2)*DX+CTTOW2(2,2)*DY+CTTOW2(3,2)*DZ

C Realign the control Matrix
CALL TOOLT2(DX2,DY2,0)

C Force/Torque summation calculations

C Translate Forces and Torques to world coordinates
DO 5 I=1,3
FTOOL1(I)=0.0
MTOOL1(I)=0.0
FTOOL2(I)=0.0
MTOOL2(I)=0.0
DO 5 J=1,3
FTOOL1(I)=FTOOL1(I)+CTTOW1(I,J)*FORCE(7-J)
MTOOL1(I)=MTOOL1(I)+CTTOW1(I,J)*FORCE(4-J)
FTOOL2(I)=FTOOL2(I)+CTTOW2(I,J)*FORCE(14-J)
MTOOL2(I)=MTOOL2(I)+CTTOW2(I,J)*FORCE(11-J)
5 CONTINUE

Sum Forces and Torques over both platforms

DO 15 I=1,3
F1ZSUM(I)=MTOOL1(4-I)
F1ZSUM(I+3)=FTOOL1(4-I)
F2ZSUM(I)=MTOOL2(4-I)
F2ZSUM(I+3)=FTOOL2(4-I)
15 FZSUM(I)=F1ZSUM(I)-F2ZSUM(I)
FZSUM(I+3)=F1ZSUM(I+3)+F2ZSUM(I+3)

Moment compensation for forces at the other platform
(Weight component is ignored in this compensation)

Mz = Mz + Fx*Dy + Fy*Dx
My = My + Fy*Dz + Fz*Dx
Mx = Mx + Fx*Dy + Fz*Dy

Transform the Force sum to Tool Coordinates of Robot 2

DO 25 I=1,3
FSUM(I)=0.0
FSUM(I+3)=0.0
25 CONTINUE

Calculate load balancing biases

FTBIAS(I)=-FSUM(I)/2.0
FTBIAS(I+7)=FSUM(I)/2.0
FTBIAS(I+7)=-.95*FTBIAS(I+7)+.05*FSUM(I)/2.0

Did we hit something?
( CHECK WHETHER FZSUM IS POINTING DOWN, AND HAS CHANGED SIGNIFICAN

RETURN
END
SUBROUTINE NAV(ROB1DAT,ROB2DAT)

NAME: Navigation

PURPOSE: Compute navigational path for robot to reach rack point

DESCRIPTION: If the robot is moving toward a set rackpoint, this routine calculates the incremental rate of movement. Movement is open loop. This subroutine assumes that the robot has actually moved that number of increments at its current incremental rate. First, the routine updates the movement goal by subtracting the distance already moved. Then, it calculates the movement differential by dividing the move goal by 25 (approx no. of control subr executions/Nav execution). This value is limited to prevent severe robot motions. Then, these moves are transformed to tool coordinates.

INPUTS:
ROB1DAT - Robot 1 transformation data supplied by VAL
ROB2DAT - Robot 2 transformation data supplied by VAL

INTEGER*2 ROB1DAT(15),ROB2DAT(15)
INTEGER SEG,MOTION,T11,T21,T31,T12,T22,T32,T13,T23,T33,T14,
        T24,T34
PARAMETER(SEG=2,MOTION=3,T11=4,T21=5,T31=6,T12=7,T22=8,
        T32=9,T13=10,T23=11,T33=12,T14=13,T24=14,T34=15)

$INCLUDE (FTORQUE.INC)

INTEGER FTMFLAG,FTD0IT,FTD0NE,FSTART,FTMFLAG2,ROBDIFI,ROBDIF2
INTEGER MVSKED,MVBAK,MVRAX
CHARACTER*40 PROMPT
CHARACTER*3 DISPT
LOGICAL*1 RESET,FTPLOT
PARAMETER (R2MSG=2,R1MSG=1)
PARAMETER (AUTO=34,GRIFFP=20,STARTD=65)
PARAMETER (FSTART=1,FTAUTO=2,FTGRIP=4,FTDONE=8)
PARAMETER (FTDOIT=16,NAVSW=32,MOVEDONE=64,FCALC=128)
PARAMETER (FTDOIT=16,NAVSW=32,MOVEDONE=64,FCALC=128)
PARAMETER (MVSKED=1,MVBAK=2,MVRAX=4)

COMMON/FTCOM/FORCE(14),FTBIAS(14),FTDIF(14),FSCALE,TSCALE,TDZ
COMMON/FTSUM/F1SUM(6),F2SUM(6),F2SUM(6),FSUM(6)
COMMON/VECT/RACKPT (4,2),ROBPT1 (6),ROBPT2 (6)
COMMON/NAVECT/BX,BY,BZ,ROBDIF1(6),ROBDIF2(6),ROBDIF1(6),ROBDIF2(6),
            RDAX,RTMAX,NCALC
IF (IAND(FTMFLAG,NAVSW).NE.0) THEN
FTMFLAG=IAND(FTMFLAG,255=FCALC)
ROBMOV=0.0

Update for Robot 1 position for increments traveled so far

(ONLY WHEN POINTS ARE INCREMENTAL)
ROBPT1(1)=ROBPT1(1)-NCALC*ROBDIF1(1)
CALL SUB2(ROBPT1(1),NCALC*ROBDIF1(1),ROBPT1(1))
ROBPT1(2)=ROBPT1(2)-NCALC*ROBDIF1(2)
CALL SUB2(ROBPT1(2),NCALC*ROBDIF1(2),ROBPT1(2))
ROBPT1(6)=ROBPT1(6)-NCALC*ROBDIF1(6)
CALL SUB2(ROBPT1(6),NCALC*ROBDIF1(6),ROBPT1(6))

Compute Robot One Desired Differential Motion in world coordinates

ROBDIF1(1)=ROBPT1(1)/25.0
For Absolute points, use
ROBDIF1(1)=(ROBPT1(1)-RX1)/25.0
CALL LIMZ(ROBDIF1(1),RMAX)
ROBDIF1(2)=ROBPT1(2)/25.0
For Absolute points, use
ROBDIF1(2)=(ROBPT1(2)-RY1)/25.0
CALL LIMZ(ROBDIF1(2),RMAX)
ROBDIF1(3)=0.0
ROBDIF1(4)=0.0
ROBDIF1(5)=0.0
ROBDIF1(6)=ROBPT1(6)/25.0
CALL LIMZ(ROBDIF1(6),RMAX)

Translate to tool coordinates for Robot 1

DO 5 I=1,3
ROBDIFX=0.0
DO 6 J=1,3
ROBDIFX=ROBDIFX+CTTOWL(I,J)*ROBDIF1(J)
CONTINUE
ROBDIF1(I)=ROBDIFX*RSSCALE(I)
ROBMOV=ROBMOV+ROBDIF1(I)
CONTINUE

Need angular translation here for gripping angle

DO 7 I=1,3
ROBDIF1(I+3)=ROBDIF1(I+3)*RSSCALE(I+3)
ROBMOV=ROBMOV+ROBDIF1(I+3)
CONTINUE

Update for Robot 2 position for increments traveled so far

(ONLY WHEN POINTS ARE INCREMENTAL)
ROBPT2(1)=ROBPT2(1)-NCALC*ROBDIF2(1)
CALL SUB2(ROBPT2(1),NCALC*ROBDIF2(1),ROBPT2(1))
ROBPT2(2)=ROBPT2(2)-NCALC*ROBDIF2(2)
CALL SUB2(ROBPT2(2),NCALC*ROBDIF2(2),ROBPT2(2))
ROBPT2(6)=ROBPT2(6)-NCALC*ROBDIF2(6)
FORTRAN-86 COMPILER

CALL SUBZ(ROBPT2(6),NCALC*ROBWDIF2(6),ROBPT2(6))
C Compute Robot Two Desired Differential Motion in world coordinates
C
ROBWDIF2(1)=ROBPT2(1)/25.0
C For Absolute points, use
C
ROBWDIF2(1)=(ROBPT2(1)-RX2)/25.0
CALL LIMZ(ROBWDIF2(1),RDMA)
ROBWDIF2(2)=ROBPT2(2)/25.0
C For Absolute points, use
C
ROBWDIF2(2)=(ROBPT2(2)-RY2)/25.0
CALL LIMZ(ROBWDIF2(2),RDYMAX)
ROBWDIF2(3)=0.0
ROBWDIF2(4)=0.0
ROBWDIF2(5)=0.0
ROBWDIF2(6)=ROBPT2(6)/25.0
CALL LIMZ(ROBWDIF2(6),RMAX)
C
C Translate to tool coordinates for Robot 2
C
DO 9 I=1,3
ROBDIFX=0.0
DO 10 J=1,3
ROBDIFX=ROBDIFX+CTTOW2(I,J)*ROBWDIF2(J)
10 CONTINUE
ROBDIF2(I)=ROBDIFX*RSCALE(I)
ROBMOV=ROBMOV+ROBDIF2(I)
9 CONTINUE
C
C Need angular translation here for gripping angle
C
DO 11 I=1,3
ROBDIF2(I+3)=ROBWDIF2(I+3)*RSCALE(I+3)
ROBMOV=ROBMOV+ROBDIF2(I+3)
11 CONTINUE
C
C Update count variable and flag
C
NCALC=0
FTMFLAG=IOR(FTMFLAG,FCALC)
IF (ROBMOV.EQ.0.0) THEN
FTMFLAG=IAND(FTMFLAG,255-NAVSW)
FTMFLAG=IOR(FTMFLAG,Movedone)
ENDIF
ELSE
C No navigation
FTMFLAG=IAND(FTMFLAG,255-FCALC)
ENDIF
C
RETURN
END
SUBROUTINE RACKMOV(N)

NAME: Rack Move

PURPOSE: Compute required movement of rack for coordinates received

DESCRIPTION: the offset of the rack from the current coordinates is computed, including the required rotational angle at each gripper.

INPUTS: N - flag
1 - Compute move to rack from current position
2 - Compute move back to position from rack

$INCLUDE (FTORQUE.INC)

PARAMETER (R2MSG=2,R1MSG=1)
PARAMETER (AUTO=34,GRIFF=26,STARTD=65)
PARAMETER (FSTART=1,FTAUTO=2,FTGRIFF=4,FTDONE=6)
PARAMETER (FTDONE=16,HAVSN=32,MOVEDONE=64,FCALC=128)
PARAMETER (MVSked=1,MVBAK=2,NVRAK=4)

COMMON/FTCON/FORCE(14),FTBIAS(14),FTDIF(14),FSCAL,TSCEAL,TDZ
COMMON/FTSUM/FT2SUM(6),F2ZSUM(6),F2SUM(6)
COMMON/FTMAT/XGAIN,YGAIN,ZGAIN,ALENTH,FTMAT2(6,6),

FTMAT(6,6),RSCALE(6)
FOOL(3),FOOL2(3),MTOOL1(3),MTOOL2(3)

COMMON/VECT/DX,DY,DZ,DX2,DY2,FHZ2,PHIM
COMMON/MSGCOM/PROMPT,RESET,FTPLOT(28)
COMMON/MCNTL/IOUTER,IFLAG,DISPT,FTMPILAG,FTMPILAG2
COMMON/NAVECT/EX,BY,BZ,RACKPT(4,2),ROBPT1(6),ROBPT2(6)
COMMON/NAVPT/ROBDIF1(6),ROBDIF2(6),ROBWIDIF1(6),ROBWIDIF2(6),

RDAMX,RDONX,NCALC

C Compute angle to move robots

IF (N.EQ.1) THEN
    ANGLE=57.3*ATAN2(RACKPT(1,1)+25.4*DX-RACKPT(2,1),
    RACKPT(1,2)+25.4*DY-RACKPT(2,2))-ATANG(DX,DY))
ENDIF

C Robot One

ROBPT1(1)=RACKPT(1,1)
ROBPT1(2)=RACKPT(1,2)
ROBPT1(3)=0.0
ROBPT1(4)=0.0
ROBPT1(5)=0.0
ROBPT1(6)=ANGLE
```
C Robot Two

31 ROBPT2(1)=RACKPT(4,1)
32 ROBPT2(2)=RACKPT(4,2)
33 ROBPT2(3)=0.0
34 ROBPT2(4)=0.0
35 ROBPT2(5)=0.0
36 ROBPT2(6)=ANGLE

C To move back, negate all values

37 IF(N.EQ.2) THEN
38     DO 10 I=1,6
39     ROBPT1(I)=-ROBPT1(I)
40 10     ROBPT2(I)=-ROBPT2(I)
41     ENDIF

C

42 RETURN
43 END
```
SUBROUTINE PANEL(SHFLAG, DATAFLG, FTSAMPL1, FTSAMPL2, SYSFLAG, TIME)

C NAME: Panel

C PURPOSE: Print data on a display screen for monitoring

C DESCRIPTION: A new screen of data is formatted approximately every seven
seconds. The top of the screen is status information on the system
while the bottom of the screen can be changed by command from the
console. The available screens are:

- NAV - Display navigation data, positions, rates
- SUN - Display force summation data, base, gripper offsets
- XFR - Display transformation matrices supplied by VAL
- MAT - Display current force/feedback matrix
- FOR - Display raw force/torque readings

C

C INPUTS: SHFLAG - Shared Common Area Flags
DATAFLG - Flags showing which data is valid on input, output
SYSFLAG - Flags showing the state of the system
ROBDAT1 - Input data from robot 1 including positional trace
ROBDAT2 - Input data from robot 2 including positional trace
FTSAMP1 - Two Force/Torque samples from the wrist of robot 1
FTSAMP2 - Two Force/Torque samples from the wrist of robot 2

2 INTEGER*1 SHFLAG(10), DATAFLG, K, SYSFLAG
3 INTEGER*2 FTSAMP1(36), FTSAMP2(36), J, TIME(2), IPC
4 CHARACTER*1 HEXDIG(16)
5 DATA HEXDIG/10',11,'12','13','14','15','16','17','18','19','A'/
6 EQUIVALENCE (J, K)

$INCLUDE (FTORQUE.INC)

$NOLIST

1 INTEGER FTFLAG, FTDIOT, FTDONE, FSTART, FTFLAG2, ROBDIP1, ROBDIP2
2 INTEGER FTGRIP, R2MSG, GRIPPD, STARTD, R1MSG, FCALC, AUTO, FTAUTO
3 INTEGER MVSKEP, MVBAK, MVRAK
4 CHARACTER*40 PROMPT
5 CHARACTER*3 DISPT
6 INTEGER*1 RESET, FTFLAG
7 INTEGER*1
8 INTEGER (AUTO=34, GRIPPD=29, STARTD=65)
9 INTEGER (FSTART=1, FTAUTO=2, FTGRIP=4, FTDONE=8)
10 INTEGER (FTDOIT=16, NAVSW=32, MOVEDONE=64, FCALC=128)
11 PARAMETER (R2MSG=2, R1MSG=1)
12 PARAMETER (AUTO=34, GRIPPD=29, STARTD=65)
13 PARAMETER (FSTART=1, FTAUTO=2, FTGRIP=4, FTDONE=8)
14 PARAMETER (FTDOIT=16, NAVSW=32, MOVEDONE=64, FCALC=128)
15 PARAMETER (MVSKED=1, MVBAK=2, MVRAK=4)
16 PARAMETER (MVSKED=1, MVBAK=2, MVRAK=4)
17 PARAMETER (MVSKED=1, MVBAK=2, MVRAK=4)
18 COMMON/FTCOM/ FORCE(14), FTBIAS(14), FTDF(14), FSCL, TSCL, TDZ
19 COMMON/FTCOM/FZSUM(6), FZSUM(6), FSUM(6), FSUM(6)
20 COMMON/FTMAT/XGAIN, YGAIN, ZGAIN, ALENTH, FTMAT2(6, 6)
21 COMMON/XFORM/ CTTOOL1(3, 3), CTTOOL2(3, 3), GTOOL2(6, 6)
22 COMMON/VECT/DX, DY, DZ, DX2, DY2, PHIZ2, DPHIM
23 COMMON/HSCOM/PROMPT, RESET, FTFLAG2(28)
24 COMMON/HSCOM/IFLAG, DISPT, FTFLAG, FTFLAG2
FORTRAN-86 COMPILER

COMMON/NAVECT/BX, BY, BZ, RACKPT(4, 2), ROBPT1(6), ROBPT2(6),
COMMON/NAVEPT/ROBDIF1(6), ROBDIF2(6), ROBWDIF1(6), ROBWDIF2(6),
=1 1 RDMAX, RTMAX, NCALC

Erase the screen, new page
CALL DISPCH(CHAR(27))
CALL DISPCH(CHAR(42))

Buffer and Flag status
J=0
CALL DISPCH('SHARE BUFFER FLAG - ')
K=SHFLAG(1)
I=J/16
CALL DISPCH(HEXDIG(I+1))
I=J-16*I
CALL DISPCH(HEXDIG(I+1))
CALL DISPCH(' FTMFLAG - ')
K=FTMFLAG
I=J/16
CALL DISPCH(HEXDIG(I+1))
I=J-16*I
CALL DISPCH(HEXDIG(I+1))
CALL DISPCH(' FTMFLAG2 - ')
K=FTMFLAG2
I=J/16
CALL DISPCH(HEXDIG(I+1))
I=J-16*I
CALL DISPCH(HEXDIG(I+1))
CALL DISPCH(CHAR(13))
CALL DISPCH(CHAR(10))

Device status
CALL DISPCH('DEVICE FLAGS - ')
I=SHFLAG(2)
CALL DISPCH('ROB 1: ')
IF (IAND(SHFLAG(2), 1).NE.0) THEN
   CALL DISPCH('ON ')
ELSE
   CALL DISPCH('OFF ')
ENDIF
CALL DISPCH('ROB 2: ')
IF (IAND(SHFLAG(2), 2).NE.0) THEN
   CALL DISPCH('ON ')
ELSE
   CALL DISPCH('OFF ')
ENDIF
CALL DISPCH('F/T 1: ')
IF (IAND(SHFLAG(2), 4).NE.0) THEN
   CALL DISPCH('ON ')
ELSE
   CALL DISPCH('OFF ')
ENDIF
CALL DISPCH('F/T 2: ')

45
FORTRAN-86 COMPILER
CONTRL.FOR

72 IF (IAND(SHFLAG(2),8).NE.0) THEN
73 CALL DISPCH('ON ')
74 ELSE
75 CALL DISPCH(' ')
76 ENDIF
77 CALL DISPCH('BRD 1: ')
78 IF (IAND(SHFLAG(2),16).NE.0) THEN
79 CALL DISPCH('ON ')
80 ELSE
81 CALL DISPCH(' ')
82 ENDIF
83 CALL DISPCH('BRD 2: ')
84 IF (IAND(SHFLAG(2),32).NE.0) THEN
85 CALL DISPCH('ON ')
86 ELSE
87 CALL DISPCH(' ')
88 ENDIF
89 CALL DISPCH(CHAR(13))
90 CALL DISPCH(CHAR(10))
91 IF (IAND(SHFLAG(2),64).NE.0) THEN
92 CALL DISPCH(' ')  
93 CALL DISPCH('ALGORITHM ACTIVE')
94 CALL DISPCH(CHAR(13))
95 ENDIF
96 CALL DISPCH(CHAR(10))
97 CALL DISPCH(CHAR(10))

C Error Flags

98 CALL DISPCH('ERROR FLAG ROBOT 1 - ')
99 K=SHFLAG(3)
100 I=J/16
101 CALL DISPCH(HEXDIG(I+1))
102 I=J-16*I
103 CALL DISPCH(HEXDIG(I+1))
104 CALL DISPCH(' ')
105 K=SHFLAG(5)
106 I=J/16
107 CALL DISPCH(HEXDIG(I+1))
108 I=J-16*I
109 CALL DISPCH(HEXDIG(I+1))
110 CALL DISPCH(' NAV LOOPS = ')
111 CALL DISPN0(NCALC,0)
112 CALL DISPCH(CHAR(13))
113 CALL DISPCH(CHAR(10))
114 CALL DISPCH(CHAR(10))

C

115 CALL DISPCH('ERROR FLAG ROBOT 2 - ')
116 K=SHFLAG(4)
117 I=J/16
118 CALL DISPCH(HEXDIG(I+1))
119 I=J-16*I
120 CALL DISPCH(HEXDIG(I+1))
121 CALL DISPCH(' ')
122 K=SHFLAG(6)
123 I=J/16
124 CALL DISPCH(HEXDIG(I+1))
FORTRAN-86 COMPILER

CONTRL.FOR

125 I=J-16*I
126 CALL DISPCH(HEXDIG(I+1))
127 CALL DISPCH(' OUTER LOOPS = ')
128 CALL DISPNO(IOUTER,0)
129 IOUTER=0
130 CALL DISPCH(CHAR(13))
131 CALL DISPCH(CHAR(10))
132 CALL DISPCH(CHAR(10))

C
133 CALL DISPCH('OVERFLOW FLAG BOARD 1 - ')
134 K=SHFLAG(7)
135 I=J/16
136 CALL DISPCH(HEXDIG(I+1))
137 I=J-16*I
138 CALL DISPCH(HEXDIG(I+1))
139 CALL DISPCH(' ')
140 K=SHFLAG(9)
141 I=J/16
142 CALL DISPCH(HEXDIG(I+1))
143 I=J-16*I
144 CALL DISPCH(HEXDIG(I+1))
145 CALL DISPCH(CHAR(13))
146 CALL DISPCH(CHAR(10))
147 CALL DISPCH(CHAR(10))

C
148 CALL DISPCH('OVERFLOW FLAG BOARD 2 - ')
149 K=SHFLAG(8)
150 I=J/16
151 CALL DISPCH(HEXDIG(I+1))
152 I=J-16*I
153 CALL DISPCH(HEXDIG(I+1))
154 CALL DISPCH(' ')
155 K=SHFLAG(10)
156 I=J/16
157 CALL DISPCH(HEXDIG(I+1))
158 I=J-16*I
159 CALL DISPCH(HEXDIG(I+1))
160 CALL DISPCH(CHAR(13))
161 CALL DISPCH(CHAR(10))
162 CALL DISPCH(CHAR(10))

C Branch here to individual display screens

C
163 IF(DISPT.EQ.'NAV') GOTO 3010
164 IF(DISPT.EQ.'SUM') GOTO 3030
165 IF(DLSPT.EQ.'XFR') GOTO 3050
166 IF(DISPT.EQ.'MAT') GOTO 3060

C FOR - Display raw force/torque readings

C
167 CALL DISPCH(' FORCE/TORQUE SENSOR 1 READINGS - ')
168 CALL DISPNO(FTSAMPL1(1),0)
169 CALL DISPCH(CHAR(13))
170 CALL DISPCH(CHAR(10))
171 CALL DISPCH(CHAR(10))
172 DO 10 I=1,7
173 CALL DISPNO(FTSAMPL1(I+1),0)
CALL DISPCH(' 
')
10 CONTINUE
CALL DISPCH(CHAR(13))
CALL DISPCH(CHAR(10))
CALL DISPCH(CHAR(10))

CALL DISPCH(' FORCE/TORQUE SENSOR 2 READINGS - 
')
CALL DISPNO(FTSAMPL2(1),0)
CALL DISPCH(CHAR(13))
CALL DISPCH(CHAR(10))
CALL DISPCH(CHAR(10))
DO 20 I=1,7
CALL DISPNO(FTSAMPL2(I+1),0)
CALL DISPCH(' ') CONTINUE
CALL DISPCH(CHAR(13))
CALL DISPCH(CHAR(10))
CALL DISPCH(CHAR(10))
GO TO 5000

C NAV - Display Navigational data
C
3010 CALL DISPCH(' ROBOT 1 ALTER - RACK X1= 
')
I=10*RACKPT(1,1)
CALL DISPNO(I,2)
CALL DISPCH(' Y1= 
')
I=10*RACKPT(1,2)
CALL DISPNO(I,2)
CALL DISPCH(' X2= 
')
I=10*RACKPT(2,1)
CALL DISPNO(I,2)
CALL DISPCH(' Y2= 
')
I=10*RACKPT(2,2)
CALL DISPNO(I,2)
CALL DISPCH(CHAR(13))
CALL DISPCH(CHAR(10))
DO 11 I=1,6
J=10*ROBPT1(I)
CALL DISPNO(J,2)
CALL DISPCH(' 
') CONTINUE
CALL DISPCH(CHAR(13))
CALL DISPCH(CHAR(10))
DO 12 I=1,6
J=10*ROBWDIR1(I)
CALL DISPNO(J,2)
CALL DISPCH(' 
') CONTINUE
CALL DISPCH(CHAR(13))
CALL DISPCH(CHAR(10))
DO 13 I=1,6
J=10*(ROBWDIR1(I)/RScale(I))
CALL DISPNO(J,2)
CALL DISPCH(' 
') CONTINUE
CALL DISPCH(CHAR(13))
CALL DISPCH(CHAR(10))
FORTRAN-86 COMPILER
CONTRL.FOR

C
227 CALL DISPCH(' ROBOT 2 ALTER - RACK X1=')
228 I=10*RACKPT(3,1)
229 CALL DISPNO(I,2)
230 CALL DISPCH(' Y1=')
231 I=10*RACKPT(3,2)
232 CALL DISPNO(I,2)
233 CALL DISPCH(' X2=')
234 I=10*RACKPT(4,1)
235 CALL DISPNO(I,2)
236 CALL DISPCH(' Y2=')
237 I=10*RACKPT(4,2)
238 CALL DISPNO(I,2)
239 CALL DISPCH(CHAR(13))
240 CALL DISPCH(CHAR(10))
241 DO 14 I=1,6
242 J=10*ROBPT2(I)
243 CALL DISPNO(J,2)
244 CALL DISPCH(')
245 14 CONTINUE
246 CALL DISPCH(CHAR(13))
247 CALL DISPCH(CHAR(10))
248 DO 15 I=1,6
249 J=10*ROBWDIF2(I)
250 CALL DISPNO(J,2)
251 CALL DISPCH(')
252 15 CONTINUE
253 CALL DISPCH(CHAR(13))
254 CALL DISPCH(CHAR(10))
255 DO 16 I=1,6
256 J=10*(ROBDIF2(I)/RSCALE(I))
257 CALL DISPNO(J,2)
258 CALL DISPCH(')
259 16 CONTINUE
260 CALL DISPCH(CHAR(13))
261 CALL DISPCH(CHAR(10))

C
262 GO TO 5000
C
C SUM - Display force/torque summation data
C
263 3030 CALL DISPCH(' FORCE Z 1 ')
264 DO 21 I=1,6
265 J=10*F1ZSUM(I)
266 CALL DISPNO(J,2)
267 CALL DISPCH(')
268 21 CONTINUE
269 CALL DISPCH(CHAR(13))
270 CALL DISPCH(CHAR(10))
271 CALL DISPCH(' FORCE Z 2 ')
272 DO 22 I=1,6
273 J=10*F2ZSUM(I)
274 CALL DISPNO(J,2)
275 CALL DISPCH(')
276 22 CONTINUE
277 CALL DISPCH(CHAR(13))
278 CALL DISPCH(CHAR(10))
FORTRAN-86 COMPILER

CALL DISPCH(' DOWN FORCE ')
DO 23 I=1,6
J=10*FSUM(I)
CALL DISPNO(J,2)
CALL DISPCH(' ')
23 CONTINUE
CALL DISPCH(CHAR(13))
CALL DISPCH(CHAR(10))
CALL DISPCH(' FORCE DIFF ')
DO 24 I=1,6
J=10*FTDIFF(I+7)
CALL DISPNO(J,2)
CALL DISPCH(' ')
24 CONTINUE
CALL DISPCH(CHAR(13))
CALL DISPCH(CHAR(10))
CALL DISPCH(' BAR SIZE ')
J=10*DX
CALL DISPNO(J,2)
CALL DISPCH(' ')
J=10*DY
CALL DISPNO(J,2)
CALL DISPCH(' ')
J=10*DZ
CALL DISPNO(J,2)
CALL DISPCH(' ')
CALL DISPCH(CHAR(13))
CALL DISPCH(CHAR(10))
CALL DISPCH(' BASE DIFF ')
J=10*BX
CALL DISPNO(J,2)
CALL DISPCH(' ')
J=10*BY
CALL DISPNO(J,2)
CALL DISPCH(' ')
J=10*BZ
CALL DISPNO(J,2)
CALL DISPCH(' ')
CALL DISPCH(CHAR(13))
CALL DISPCH(CHAR(10))
CALL DISPCH(' ROBOT 1 TRANSFORM ')
CALL DISPCH(' ROBOT 2 TRANSFORM ')
CALL DISPCH(CHAR(13))
CALL DISPCH(CHAR(10))
GOTO 5000

C XFR - Display current robot transformation

CALL DISPCH(' ROBOT 1 TRANSFORM ')
CALL DISPCH(' ROBOT 2 TRANSFORM ')
CALL DISPCH(CHAR(13))
CALL DISPCH(CHAR(10))
CALL DISPCH(CHAR(10))
DO 34 L=1,3
DO 32 I=1,3
J=1000*CTTowl(L,I)
CALL DISPNO(J,4)
CALL DISPCH(' ')
34 CONTINUE
FORTRAN-86 COMPILER

CONTRL.FOR

333 CALL DISPCH(' 
334 DO 33 I=1,3
335 J=1000*CTTW2(L,I)
336 CALL DISPNO(J,4)
337 CALL DISPCH(' 
338 33 CONTINUE
339 CALL DISPCH(CHAR(13))
340 CALL DISPCH(CHAR(10))
341 CALL DISPCH(CHAR(10))
342 34 CONTINUE
343 GOTO 5000

C MAT - Display force feedback matrix
C
344 3060 CALL DISPCH(' PHIZ -')
345 J = 100*PHIZ2
346 CALL DISPNO(J,3)
347 CALL DISPCH(' DX2 -')
348 J = 100*DX2
349 CALL DISPNO(J,3)
350 CALL DISPCH(' DY2 -')
351 CALL DISPCH(' ')
352 J = 100*DY2
353 CALL DISPNO(J,3)
354 CALL DISPCH(CHAR(13))
355 CALL DISPCH(CHAR(10))
356 DO 38 L=1,6
357 DO 36 I=1,6
358 J=100000*FTMATX(L,I)
359 CALL DISPNO(J,6)
360 CALL DISPCH(' ')
361 36 CONTINUE
362 CALL DISPCH(CHAR(13))
363 CALL DISPCH(CHAR(10))
364 38 CONTINUE

C Complete display, print prompt
C
365 5000 J=0
366 CALL DISPCH('SYSTEM FLAG -')
367 K=SYSFLAG
368 I=J/16
369 CALL DISPCH(HEXDIG(I+1))
370 I=J-16*I
371 CALL DISPCH(HEXDIG(I+1))
372 CALL DISPCH(' ')
373 CALL DISPCH('IDLE TIME -')
374 FPC=100.0*TIME(1)/TIME(2)
375 IPC=FPC
376 CALL DISPNO(IPC,0)
377 CALL DISPCH(' PERCENT ')

C
378 CALL DISPCH(CHAR(13))
379 CALL DISPCH(CHAR(10))
380 CALL DISPCH(CHAR(10))

C
381 CALL DISPCH(PROMPT)
C If reset flag was set, reset cumulative error flags
C
382 IF (RESET) THEN
383 SHFLAG(5)=0
384 SHFLAG(6)=0
385 SHFLAG(9)=0
386 SHFLAG(10)=0
387 ENDIF
388 RESET=.FALSE.
389 RETURN
390 END
SUBROUTINE DACDO(FTSAMP1, FTSAMP2)

C NAME: DAC Routine
C PURPOSE: Output data to DAC for plotting
C DESCRIPTION: Calls assembler DAC output routine seven times
C INPUTS: FTSAMP1 - Seven force/torque readings from robot 1
C          FTSAMP2 - Seven force/torque readings from robot 2
C
INTEGER*2 FTSAMP1(7), FTSAMP2(7), I

DO 10 I = 1, 7
CALL DAC(FTSAMP2(I), I)
10 CONTINUE

RETURN
END
SUBROUTINE COMMAND(LINE,NSW)

CHARACTER*(*) LINE

NAME: Command

PURPOSE: Process input commands and take proper action

DESCRIPTION:

INPUTS: LINE - Command Line
NSW - Switch for communicating with assembler driver

CHARACTER*1 CMDS(10),NOS(6),CHARA,DIRS(5)
DIMENSION DEL(6)

$INCLUDE (FORQUINC.INC)
=1 $NOLIST

INTEGER FTMFLAG,FTDOIT,FTDONE,FSSTART,FTMFLAG2,ROBDIF1,ROBDIF2
INTEGER FTGRI P,R2MSG,GRIPPED,STARTD,R1MSG,FCALC,AUTO,FTAUTO
INTEGER MVSKEO,MVBAK,MTGRAK
CHARACTER*40 PROMPT
CHARACTER*3 DlSP
LOGICAL*1 RESET,FTPLOT

PARAMETER (R2MSG=2,R1MSG=1)
PARAMETER (AUTO=34,GRI PPD=20,STARTD=65)
PARAMETER (FSSTART=1,FAULTO=2,FTGRIP=4,FTDONE=8)
PARAMETER (FTDOIT=16,NAVSW=32,MVEDONE=64,FCALC=128)
PARAMETER (MVSKEO=1,MVBAK=2,MTGRAK=4)

COMMON/FTCOM/FORC(14),FTBIAS(14),FTDIFF(14),FSSCALE,TSCALE,TDZ
COMMON/FTSUM/FZSUM(6),FZSUM(6),SFUM(6)
COMMON/FTMAT/XGAIN,YGAIN,ZGAIN,ALETH,FTMAT2(6,6),
FTMNX(6,6),RSacle(6)
COMMON/XF0RNX,CTTOW1(3,3),CTTOW2(3,3),GTOOL2(6,6),
FTOOL1(3),FTOOL2(3),MTOOL1(3),MTOOL2(3),GTOOL1(3,3)
COMMON/VECT/BX,BY,BZ,DX2,DY2,PHI22,DPH1M
COMMON/MSGCOM/PROMPT,RESET,FTPLOT(28)
COMMON/MCXTL,IOUTER,IFLAG,DISP,FTMFLAG,FTMFLAG2
COMMON/NAVEP/BX,BY,BZ,RAKPT(4,2),ROBPT1(6),ROBPT2(6)
COMMON/NAVEP/ROBDIF1(6),ROBDIF2(6),ROBDIF1(6),ROBDIF2(6),
RDMAx,RTRMAx,N_CALC

DATA CMDS/"R","M","P","S","G","K","D","C","Q","F"/
DATA NOS/1,"2","3","4","5","6"/
DATA DEL/10.0,10.0,10.0,10.0,10.0,10.0/
DATA DIRS/"X","Y","Z","A","P"/

IF (NSW.EQ.2) GOTO 100

DO 1 I=1,10
IF (LINE(111).EQ.CMDS(1)) GOTO 5
CONTINUE
1 PROMPT='COMMAND NOT KNOWN'
RETURN
FORTRAN-86 COMPILER

CONTRL.FOR

C
35  PROMPT='ENTER COMMAND'
36  GOTO (10,20,30,40,50,60,70,80,90,99),I
C
C  R - Reset command
C
37  10  RESET=.TRUE.
38  NSW=1
39  FTMFLAG=IAND(FTMFLAG,255-NAVSW-FTDOIT-MOVEDONE-FCALC)
40  DO 15  I=1,6
41  ROBDIF1(I)=0
42  ROBDIF2(I)=0
43  ROBSDIF1(I)=0.0
44  ROBSDIF2(I)=0.0
45  ROBP1(I)=0.0
46  15  ROBP2(I)=0.0
47  RETURN
C
C  M - Modify command
C
48  20  CHARA=LINE(3:3)
49  DO 21  I=1,5
50  IF(CHARA.EQ.DIRS(I)) GOTO 24
51  21  CONTINUE
52  PROMPT='BAD MODIFY DIRECTION'
53  RETURN
54  24  I=5
55  CALL GETNO(LINE,I,FNO,CHARA)
56  GO TO (22,23,25,27,28),I
57  RETURN
58  22  XGAIN = FNO
59  GOTO 26
60  23  YGAIN = FNO
61  GOTO 26
62  25  ZGAIN = FNO
63  GOTO 26
64  27  ALENTH=FNO
65  CALL GAINM
66  CALL TOOLT2(0.0,0.0,1)
67  RETURN
68  28  DPHIM=FNO
69  RETURN
C
C  P - Print command - no longer implemented
C
70  RETURN
C
C  S - Enter New Co-ordinates for robotic movement
C
71  40  I=3
72  DO 42  I=1,4
73  CALL GETNO(LINE,I,RACKPT(I1,1),CHARA)
74  IF((CHARA.NE.'').OR.(I.LT.0)) GOTO 43
75  CALL GETNO(LINE,I,RACKPT(I1,2),CHARA)
76  IF((I1.EQ.4).AND.(I.LT.0)) GOTO 45
77  IF((CHARA.NE.'').OR.(I.LT.0)) GOTO 43
78  42  CONTINUE

55
FORTRAN-86 COMPILER

CONTRL.FOR

79 43 PROMPT='RACKPOINT ERROR'
80 45 FTMFLAG2=IOR(FTMFLAG2,MVSKED)
81 RETURN

C

G - Go Command

82 50 NSW=4
83 IF(IAND(FTMFLAG2,MVSKED).EQ.0) FTMFLAG=IOR(FTMFLAG,MOVEDONE)
84 RETURN

C

K - Keyboard Guidance Command

85 60 NSW=2
86 PROMPT = 'ENTER DIRECTION '
87 RETURN

C

D - Display Command

88 70 DISPT = LINE(3:5)
89 RETURN

C

C - Calibrate command

90 80 IF(LINE(3:3).EQ.'5') GOTO 86
91 I=3
92 CALL GETNO(LINE,I,X1,CHARA)
93 IF((CHARA.NE.' ').OR.(I.LT.0)) GOTO 83
94 CALL GETNO(LINE,I,Y1,CHARA)
95 IF((CHARA.NE.' ').OR.(I.LT.0)) GOTO 83
96 CALL GETNO(LINE,I,Z1,CHARA)
97 IF(I.GE.0) GOTO 83
98 BX = X1
99 BY = Y1
100 BZ = Z1
101 RETURN

102 83 PROMPT = 'CALIBRATE ERROR'
103 RETURN

104 86 BX = -DX
105 BY = -DY
106 BZ = -DZ
107 RETURN

C

Q - Quit command

108 90 CALL DISPCH(CHAR(27))
109 CALL DISPCH(CHAR(42))
110 DO 92 I=1,10
111 CALL DISPCH(CHAR(10))
112 92 CONTINUE
113 CALL DISPCH(CHAR(13))
114 CALL DISPCH('PROCESSOR TWO IN TERMINATION LOOP')
115 CALL DISPCH(CHAR(10))
116 CALL DISPCH(CHAR(10))
117 CALL DISPCH(CHAR(10))
118 CALL DISPCH('

119 NSW = 3
120 RETURN

C
C F - Force/Torque feedback control

121 99 IF(LINE(3:3).EQ.'O') THEN
122      FTMFLAG=IOR(FTMFLAG,FTDOIT)
123      ELSE
124      FTMFLAG=IAND(FTMFLAG,255-FTDOIT)
125     ENDIF
126     RETURN

C Directional Control

127 100 CALL GETDIR(LINE,NDIR,IEND)
128 IF(IEND.NE.0) THEN
129      NSW=1
130      PROMPT = 'ENTER COMMAND'
131     RETURN
132    ENDIF
133 IF (NDIR) 130,120,110
134 120 RETURN
135 130 ROBPT1(-NDIR)=ROBPT1(-NDIR)+DEL(-NDIR)
136 ROBPT2(-NDIR)=ROBPT2(-NDIR)+DEL(-NDIR)
137 GOTO 140
138 110 ROBPT1(NDIR)=ROBPT1(NDIR)-DEL(NDIR)
139 ROBPT2(NDIR)=ROBPT2(NDIR)-DEL(NDIR)
140 140 FTMFLAG=IOR(FTMFLAG,NAVSW)
141 RETURN

C END
C SUBROUTINE GETDIR(LINE,NDIR,LEAVE)
C Gets the motion direction from keyboard input line
C
C CHARACTER*1 LINE
 CHARACTER*1 DIRKEY(19), CHAR
 DATA DIRKEY/'1','2','3','4','5','6','Q','W','E','R',
  /T','Y','G','B','N','D','F','P','A'/
C $INCLUDE (FTORQUE.INC)
C $NOLIST

INTEGER FTMFLAG, FTDOIT, FTDONE, FSTART, FTMFLAG2, ROBDIF1, ROBDIF2
INTEGER MVSKED, MVBAK, MVRK
INTEGER NWGRP,MVPPD, STAT4Mq
INTEGER VKDMB(4), PRMPT4RA
INTEGER*1 DISPT
LOGICAL existing, FTPLOT

PARAMETER (R2MSG=2, R1MSG=1)
PARAMETER (AUTO=34, GRIPP=20, STARTD=65)
PARAMETER (FSTART=1, FTAUTO=2, FTGRIP=4, FTDONE=8)
PARAMETER (FTDOIT=16, NAVSW=32, MOVEDONE=64, FCALC=128)
PARAMETER (MVSKED=1, MVBAK=2, MVRK=4)

COMMON/FCOM/FRCE(A.), FTBIA(14), FTDIF(14), FSCALE, TSCALE, TDZ
COMMON/FTSUM/FZSUM(6), F2ZSUM(6), FSUM(6)
COMMON/FTMAT/XGAIN, YGAIN, ZGAIN, ALPHNT, FTMAT2(6,6),
   FMAT(6,6), RSSCALE(6)
COMMON/XFORM/CTTOW1(3,3), CTTOW2(3,3), GTOOL2(6,6),
   GTOOL1(3,3)
COMMON/VECT/DX, DY, DZ, DX2, DX2, PHIZ2, PHIN
COMMON/MSGCM/PROMPT, RESET, FTPLOT(28)
COMMON/HCTRL/OUTER, IFLAG, DISPT, FTMFLAG, FTMFLAG2
COMMON/NAVSW/RBPT1(6), RBPT2(6)
COMMON/NAVORG/ROBDIF1(6), ROBDIF2(6), ROBDIF3(6), ROBDIF4(6),
   MVRK(6,6), MVBAK(6,6)
COMMON/XFORM/CTTOW1(3,3), CTTOW2(3,3), GTOOL2(6,6),
   GTOOL1(3,3)

CHAR=LINE(1:1)
LEAVE=0
IF((.NOT.CHAR.EQ.'?') .OR. CHAR.EQ.'?') THEN
  LEAVE=1
  RETURN
ENDIF
DO 10 I=1,18
10 CONTINUE
IF (CHAR.EQ.DIRKEY(I)) GOTO 20
CONTINUE
PROMPT = 'BAD DIRECTION KEY'
NDIR=0
RETURN

20 IF(I.GT.12) I=I-6
IF(I.GT.6) THEN
  NDIR=I-6
ELSE
  NDIR=-I
ENDIF
43       PROMPT = 'ENTER DIRECTION'
44       RETURN
45       END
SUBROUTINE GETNO(LINE,N,A,ECHAR)
C
C Translates Number from input line
C
C Inputs
C
LINE (Character) Input line to parse
N (Integer) Starting position in input line
C
C Outputs
C
A (Real) Number returned
ECHAR (Character) Delimiting character at end of line
C
CHARACTER(*) LINE
CHARACTER*35 TEMP
CHARACTER*1 NOS(13),ECHAR
DATA NOS/'1','2','3','4','5','6','7','8','9','0','

$INCLUDE (FORTRAN.INC)

INTEGER FTMFLAG,FTDOIT,FTDONE,FSTART,FTFLAG2,ROBDIF1,ROBDIF2
INTEGER FTHINGRIP,R2MSG,GRIPPD,STARTD,R1MSG,FCALC,AUTO,FTAUTO
INTEGER MVSKED,MVBAK,MVRAK
CHARACTER*40 PROMPT
CHARACTER*3 DISPT
LOGICAL*1 RESET,FTPLOT
PARAMETER (R2MSG=2,R1MSG=1)
PARAMETER (AUTO=34,GRIPPD=20,STARTD=65)
PARAMETER (FSTART=1,FTAUTO=2,FTHINGRIP=4,FTDONE=8)
PARAMETER (FTDOIT=16,NAVSW=32,MOVEDONE=64,FCALC=128)
PARAMETER (MVSKED=1,MVBAK=2,MVRAK=4)
COMMON/FTCOM/FORCE(14),FTBIAS(14),FTDIFF(14),FSCALE,TSSCALE,TDZ
COMMON/FTSUM/F1ZSUM(6),F2ZSUM(6),FZSUM4(6),FSUM(6)

COMMON/XFORM/CTT01(3,3),CTT02(3,3),GTOOL2(6,6),
   FT01(3),FT02(3),MTOOL1(3),MTOOL2(3),GTOOL1(3,3)
COMMON/VECT/DX,DY,DZ,DX2,DY2,PHIZ2,DPHIM
COMMON/MSGCOM/PROMPT,RESET,FTPLOT(28)
COMMON/MCINT/IOUTER,IFLAG,DISPT,FTMFLAG,FTMFLAG2
COMMON/NAVECT/BX,BY,BZ,RACKPT(4,2),ROBDPT1(6),ROBDPT2(6)
COMMON/NAVEPT/ROBDIF1(6),ROBDIF2(6),ROBDIF2(6),ROBDIF2(6)

A=0.0
IF(N.LT.0) RETURN
AN0=0.0
APLUS=1.0
ADOT=0.0
DO 10 I=N,50
ECHAR=LINE(I:I)
FORTRAN-86 COMPILER
CONTRL.FOR

33 IF(ICHAR(ECHAR).EQ.13) GOTO 110
34 IF((ECHAR.EQ.,').OR.(ECHAR.EQ.,')) GOTO 100
35 DO 11 J=1,14
36 IF (ECHAR.EQ.NOS(J)) GOTO 12
37 11 CONTINUE
38 GOTO 1000
39 12 IF(J.LE.10) THEN
40 IF(J.EQ.10) J=0
41 IF(ADOT.EQ.0.0) THEN
42 ANO=10.0*ANO+J
43 ELSE
44 ANO=ANO+ADOT*J
45 ADOT=ADOT/10.0
46 ENDIF
47 ELSE
48 IF (J.EQ.11) ADOT=.I
49 IF (J.EQ.13) APLUS=-1.0
50 ENDIF
51 10 CONTINUE
52 GO TO 1000

C Return from the middle of the line
C
53 100 A=ANO*APLUS
54 N=I+1
55 IF(ECHAR.NE.,')') RETURN
56 DO 105 I=N,50
57 IF(ICHAR(LINE(I:I)).EQ.13) GOTO 110
58 IF(LINE(I:I).NE.,')') GOTO 103
59 105 CONTINUE
60 103 N=I
61 RETURN

C Return after carriage return
C
62 110 A=ANO*APLUS
63 N=-1
64 RETURN

C Error return
C
65 1000 PROMPT='BAD NUMBER'
66 N=-1
67 RETURN
68 END
SUBROUTINE TOOLT2(RX,RY,N)
DIMENSION FTMATB(6,6)

$INCLUDE (FTORQUE.INC)
$NOLIST

INTEGER FTMFLAG,FTDOIT,FTDONE,FSTART,FTMFLAG2,ROBDIF1,ROBDIF2
INTEGER FTGRIP,R2MSG,GRIPPD,STARTD,R1MSG,FCALC,AUTO,FTAUTO
INTEGER MVSKED,MVBAK,MVRAK
CHARACTER*40 PROMPT
CHARACTER*3 DISPT
LOGICAL*1 RESET,FTPLOT

PARAMETER (R2MSG=2,R1MSG=1)
PARAMETER (AUTO=34,GRIPPD=20,STARTD=65)
PARAMETER (FSTART=1,FTAUTO=2,FTGRIP=4,FTDONE=8)
PARAMETER (FTDOIT=16,NAVSW=32,MOVEDONE=64,FCALC=128)
PARAMETER (FTSTART=1,FTAUTOR=2,FTGRIPT=4,FTDONE=8)
PARAMETER (FTDOIT=16,NAVSW=32,MOVEDONE=64,FCALC=128)
PARAMETER (MVSKED=1,MVBAK=2,MVRAK=4)

COMMON/FTCOM/FORCE(14),FTBIAS(14),FTDIFF(14),FSIZE,FTSCALE,FTSIZE
COMMON/FTSUM/FTZSUM(6),FZSUM(6),FTSUM(6)
COMMON/XFORM/XGAIN,YGAIN,ZGAIN,ALength,FTMAT2(6,6),
COMMON/FTMAT2/FMTABX(6,6),FTSAME(6)
COMMON/XFORM/XTOWN1(3,3),XTOWN2(3,3),GTOOL2(6,6),
COMMON/FTMAT2/GTOOL1(3,3),GTOOL1(3,3)
COMMON/VECT/DX,DY,DZ,DX2,DY2,PHIZ2,DPHI
COMMON/VECT/DX,DY,DZ,DX2,DY2,PHIZ2,DPHI
COMMON/MNCTL/IOUTER,IFLAG,DISPT,FTMFLAG,FTMAT2
COMMON/NAVECT/BX,BY,BZ,RACKPT(4,2),ROBPT1(6),ROBPT2(6)
COMMON/MNCTL/IOUTER,IFLAG,DISPT,FTMFLAG,FTMAT2
C
IF (N.LE.0) THEN
IF (N.LT.0) THEN
PHIZ2=0.0
ELSE
PHIZ2=ATAN2(RX,RY)
DPHI=PHIZ2-PHIZ2
CALL LIMZ(DPHI,DPHI)
PHIZ2=PHIZ2+DPHI
ENDIF
ENDIF
C
CZ=COS(PHIZ2)
SZ=SIN(PHIZ2)
C
DO 20 I=1,6
DO 20 J=1,6
GTOOL2(I,J)=0.0
C
DO 30 I=1,2
GTOOL2(3*I-2,3*I-2)=CZ
GTOOL2(3*I-2,3*I-1)=SZ
GTOOL2(3*I-1,3*I-2)=-SZ
GTOOL2(3*I-1,3*I-1)=CZ
GTOOL2(3*I,3*I)=1.0
C
CONTINUE
DO 35 I=1,6
DO 35 J=1,6
FTMATB(I,J)=0.0
DO 35 K=1,6
FTMATB(I,J)=FTMATB(I,J)+FTMAT2(I,K)*GTOOL2(7-J,7-K)
CONTINUE
DO 45 I=1,6
DO 45 J=1,6
FTMSUM=0.0
DO 40 K=1,6
FTMSUM=FTMSUM+GTOOL2(I,K)*FTMATB(K,J)
CONTINUE
FTMATX(I,J)=FTMSUM
END
SUBROUTINE GAINM

$INCLUDE (FTORQUE.INC)

=1

INTEGER FTMFLAG, FTDMAIT, FTDONE, FSTART, FTMFLAG2, ROBDEF1, ROBDEF2
=1

INTEGER FTGRIP, R2MSG, GRIPPD, STARTD, R1MSG, FCALC, AUTO, FTAUTO
=1

INTEGER MVSKED, MVBAK, MVRAK
=1

CHARACTER*40 PROMPT
=1

CHARACTER*3 DISPT
=1

LOGICAL*1 RESET, FTPLOT
=1

PARAMETER (R2MSG=2, R1MSG=1)
=1

PARAMETER (AUTO=34, GRIPPD=20, STARTD=65)
=1

PARAMETER (FSTART=1, FTAUTO=2, FTGRIP=4, FTDONE=8)
=1

PARAMETER (FTDOIT=16, NAVSW=32, MOVEDONE=64, FCALC=128)
=1

PARAMETER (MVSKED=1, MVBAK=2, MVRAK=4)
=1

COMMON/FTCOM/FORCE(14), FTBIAS(14), FTDIF(14), FSACLE, TSCALE, TDZ
=1

COMMON/FTSUM/F1ZSUM(6), F2ZSUM(6), FSUM(6)
=1

COMMON/FTMAT/XGAIN, YGAIN, ZGAIN, ALENTH, FTMAT2(6, 6),
=1

COMMON/VECT/DX, DY, DZ, DX2, DY2, PHIZ2, DPHIM
=1

COMMON/MSCOM/PROMPT, RESET, FTPLOT(28)
=1

COMMON/MTN/OUTER, IFLAG, DISPT, FTMFLAG, FTMFLAG2
=1

COMMON/NAVEC/BX, BY, BZ, RACKPT(4, 2), ROBPT1(6), ROBPT2(6)
=1

COMMON/NAVEP/ROBDIF1(6), ROBDIF2(6), ROBWDIF1(6), ROBWDIF2(6),
=1

COMMON/NAVEPT/ROBDIF1(6), ROBDIF2(6), ROBWDIF1(6), ROBWDIF2(6),
=1

COMMON/NAVEP/ROBDIF1(6), ROBDIF2(6), ROBWDIF1(6), ROBWDIF2(6),
=1

COMMON/NAVEP/ROBDIF1(6), ROBDIF2(6), ROBWDIF1(6), ROBWDIF2(6),
=1

C

C Set the gain matrix for robot 2
C

FTMAT2(2,5)=YGAIN/20.0
=1

FTMAT2(5,2)=-YGAIN/80.0
=1

FTMAT2(1,6)=XGAIN
=1

FTMAT2(1,1)=XGAIN/ALENTH
=1

FTMAT2(6,6)=XGAIN/ALENTH
=1

FTMAT2(6,1)=2.0*YGAIN/(ALENTH*ALENTH)
=1

FTMAT2(3,4)=2.0*ZGAIN

RETURN
=1

END
SUBROUTINE LIMZ(AVAL, RATE)

C

IF(AVAL.GT.0.0) THEN
  IF(AVAL.GT.RATE) AVAL=RATE
  IF(AVAL.LT..0001) AVAL=0.0
ELSE
  IF(AVAL.LT.-RATE) AVAL=-RATE
  IF(AVAL.GT..0001) AVAL=0.0
ENDIF
RETURN
END
SUBROUTINE SUBZ(A,B,C)

X=ABS(A-B)
Y=A+.0005
Y=ABS(Y-B)
Z=A-.0005
Z=ABS(Z-B)

IF((X.LT..0001).OR.(Y.LT..0001).OR.(Z.LT..0001)) THEN
  C=0.0
ELSE
  C=A-B
ENDIF
RETURN
END
SUBROUTINE SUBZ(A, B, C)

X = ABS(A - B)
Y = A + .0005
Y = ABS(Y - B)
Z = A - .0005
Z = ABS(Z - B)

IF ((X < .0001) .OR. (Y < .0001) .OR. (Z < .0001)) THEN
  C = 0.0
ELSE
  C = A - B
ENDIF

RETURN
END